Contribution à la conception d'applications de pilotage des systèmes manufacturiers

Benoit Rohée

UNIVERSITE DE REIMS CHAMPAGNE ARDENNE

Thèse

Présentée pour obtenir le grade de

Docteur de l'Université de Reims Champagne-Ardenne

Spécialité
Spécialité : Génie Informatique, Automatique et Traitement du Signal

Par

Benoît ROHÉE

Contribution à la conception d'applications de pilotage des systèmes manufacturiers

Soutenue publiquement le 19 Décembre 2008, devant le jury composé de

Rapporteurs :		
	M. Serge Debernard	Professeur, Université de Valenciennes et du Hainaut-Cambrésis
	M. Benoît Eynard	Professeur, Université de Technologie de Compiègne
Examinateur :		
	M. Carlos Moreno	Professeur, Directeur Général de Sinovia SAS
Codirecteurs de Thèse :		
	Mme Véronique Carré-Ménétrier	Professeur, Université de Reims Champagne-Ardenne
	M. Bernard Riera	Professeur, Université de Reims Champagne-Ardenne
Encadrant :		
	M. Jean-Marc Roussel	Maître de Conférences, École Normale Supérieure de Cachan

Remerciements

Je souhaite remercier Monsieur Carlos Moreno, professeur et directeur général de SINOVIA SAS, qui m'a fait l'honneur d'exercer les fonctions de président du jury et d'examinateur de thèse pour le temps qu'il a consacré à étudier mon travail et pour sa sympathie.

Mes remerciements s'adressent également aux rapporteurs de mon mémoire de thèse Monsieur Serge DEBERNARD, professeur à l'Université de Valenciennes et du Hainaut-Cambrésis, et Monsieur Benoît EYNARD, professeur à l'UTC de Compiègne pour avoir participer à ce jury de thèse. Je leur exprime toute ma reconnaissance pour l'intérêt porté à ce travail.

J'aimerais remercier vivement, mes co-directeurs de thèse Madame Véronique CARRE-MENETRIER, directrice de l'UFR Sciences Exactes et naturelles, pour ses précieux conseils, sa disponibilité malgré son emploi du temps chargé et sa gaieté et Bernard RIERA pour tous les moments passés ensemble que ce soit pour travailler intellectuellement sur les travaux de recherche, de m'initier aux projets industriels ou aux travaux physiques de terrassement. Je leur exprime une grande reconnaissance pour m'avoir soutenu à achever la thèse. Je remercie Jean-Marc Roussel pour m'avoir donné des conseils avisés sur mon travail et pour m'avoir suivi malgré la distance.

Je souhaite remercier les personnes de l'Université de Reims Champagne Ardenne avec qui j'ai pu travailler et vivre quotidiennement et notamment François pour m'avoir initié à l'enseignement sous une forme différente à celle que j'ai pu voir précédemment, Pascale d'avoir réussi à partager le bureau pendant ces trois années malgré ma présence ainsi que toutes les personnes des départements d'enseignements dans lesquels j'ai travaillé, du laboratoire de recherche et des services techniques et administratifs de l'Université. Je remercie Alexandre, Doc, Fab, Laurent, François, Pascale et aux joueurs occasionnels pour leur ténacité à avoir essayé de m'apprendre à jouer au tarot.

Je souhaite enfin remercier toute ma famille de m'avoir soutenu lors de ces trois années difficiles malgré la distance et mon manque de présence dans ma région natale. Pour finir, un dernier mot à la personne qui partage le bureau (et un peu plus) avec moi en ce moment.

Je suis tombé totalement par hasard sur cette citation il y a très longtemps et je pense qu'elle convient très bien à ma vision de l'enseignement.

Le savoir n'est pas une vulgaire matière première. Il ne vient jamais à épuisement. Au contraire, il s'accroît toujours grâce à la diffusion des connaissances. -+- Daniel Boorstin -+-

Table des matières

Introduction Générale

L'objectif principal de la production industrielle est d'apporter une valeur ajoutée à un produit pour répondre aux besoins d'un client. Le monde de la production industrielle distingue, en fonction de la matière traitée, trois types de procédés de fabrication, appelés aussi Parties Opératives (PO) : batch, continu et manufacturier.

Les procédés de type batch également appelés discontinu ou encore par lots transforment, pendant une période de temps fixée, des quantités déterminées de matière en produit fini selon des procédures spécifiques (recettes). Ces systèmes, au contraire des systèmes continus, sont conçus pour faire face à des changements et donc des arrêts de production. Ces procédés se rencontrent dans la chimie fine, la pharmacie et l'agroalimentaire.

Dans les procédés continus (pétrochimie, centrale nucléaire, etc.), les flux de matière et/ou d'énergie traités sont continus. Les procédés de type continu ont donc une seule phase principale de production. Le volume de la production dépend essentiellement de la durée de cette phase. La matière première « entre » dans l'unité, et le produit fini en ressort sans discontinuité.

Dans les procédés manufacturiers, l'évolution du nombre de produits réalisés est discrète (nombre de voitures produites par exemple). L'industrie manufacturière regroupe des domaines variés comme la fabrication par assemblage ou par transformations successives que l'on retrouve par exemple dans la construction automobile et l'aéronautique. **Le travail de recherche présenté dans ce mémoire concerne principalement les systèmes manufacturiers.**

Quelque soit le procédé de production, le travail de l'automaticien consiste à proposer les méthodes et outils permettant de piloter (au sens large) des systèmes dynamiques, et plus généralement de prendre des décisions à leur égard. Les activités de conduite (i.e. le pilotage) comprennent entre autres, la planification, l'ordonnancement, la commande, la surveillance, le diagnostic, la supervision aussi bien en fonctionnement normal qu'en fonctionnement dégradé. La prise de décision nécessite l'acquisition et le traitement de signaux, d'images, et d'informations et repose sur l'observation et l'identification de l'état de la partie opérative. Le travail de l'automaticien requiert donc l'utilisation de modèles, qui représentent un point de vue particulier sur le système, le plus souvent pour un objectif applicatif donné. **La recherche que nous avons menée est une contribution à la conception et à l'utilisation de modèles comportementaux de la partie opérative pour les applications de pilotage des systèmes manufacturiers.**

L'automaticien classe les systèmes en fonction des variables d'état et de temps manipulées dans les modèles. En reprenant la classification proposée par [Flaus, 97], les variables d'état d'un système de production peuvent être :

- continues (évolution décrite sur une échelle de nombres réels, une température par exemple),
- discrètes (évolution décrite par un nombre entier, un nombre de pièces par exemple),
- symboliques (évolution décrite par un ensemble discret d'états, les différents états d'une vanne Tout Ou Rien (TOR) - fermée, ouverte, en défaut - par exemple).

Il est possible d'associer aux variables symboliques, une variable discrète numérique bornée. Toutefois, la variable est simplement énumérative et non structurée contrairement aux variables discrètes qui décrivent explicitement des situations.

L'écoulement du temps peut être représenté par une variable temporelle de plusieurs natures :

- continue (le système peut être alors décrit par un ensemble d'équations algébro-différentielles),
- discrète (les signaux décrivant l'évolution des variables du système sont échantillonnés afin de construire un modèle échantillonné),
- symbolique (le temps sert uniquement à définir une chronologie entre les événements).

Les grandeurs physiques réelles sont de type continu. Cependant, la connaissance, à tout instant de la valeur de ces variables d'états, n'est pas toujours possible ni utile et dépend du type d'application (commande, diagnostic, reconfiguration, …) et du type de procédé (continu, batch, manufacturier).

Les procédés continus sont plutôt perçus par l'automaticien au travers d'une représentation reposant le plus souvent sur des variables d'état continues et une variable temporelle, continue ou discrète. De ce fait, les outils de l'automaticien sont ceux de l'automatique continue et discrète. Les modèles utilisés reposent sur des équations algébro-différentielles continues ou échantillonnées.

Les procédés manufacturiers, quant à eux, sont principalement décrits par l'automaticien au moyen de modèles reposant sur des variables d'états discrètes (de type booléennes en particulier) et une variable temporelle de nature symbolique. En conséquence, les outils de l'automaticien dans ce cas, sont ceux de l'automatique événementielle et reposent principalement sur la théorie des Systèmes à Evénements Discrets (SED). Les automaticiens ont alors recours à des outils et méthodes de modélisation comme les automates à états, les Réseaux de Petri (RdP) ou encore le GRAphe Fonctionnel de Commande-Etape-Transition (GRAFCET). Ces modèles reposent sur une représentation graphique particulièrement adaptée à une description d'opérations séquentielles et parallèles, ponctuées d'événements prévisibles (arrivée en butée d'une action) ou aléatoires (appui par un opérateur sur un bouton d'urgence) dans un mode de fonctionnement donné (normal et/ou dégradé).

Les recherches menées en automatique, continue ou événementielle, qu'elles soient théoriques ou appliquées, nécessitent l'utilisation d'un modèle du système dynamique. Toutefois, il nous semble qu'il existe des différences importantes entre les automaticiens du continu et les automaticiens des SED. Dans le premier cas, les recherches, qu'elles soient fondamentales ou appliquées, reposent sur une utilisation explicite du modèle comportemental dynamique de la partie opérative. Les choses sont différentes dans les SED.

En pratique, le modèle de la partie opérative n'est pas forcément explicité pour mener à bien des applications, en particulier dans le domaine de la commande. En revanche, les travaux théoriques dans le domaine des SED reposent sur un modèle de la partie opérative qui peut sembler abstrait et éloigné de la réalité des systèmes manufacturiers.

Pourquoi existe-t-il ce fossé entre la théorie (SED) et la pratique (procédés manufacturiers) ? Est-il justifié ? Quelles sont les possibilités d'utilisation d'un modèle de partie opérative pour les applications de pilotage dans les procédés manufacturiers ? Comment modéliser un système manufacturier, quels modèles et quelle méthodologie de modélisation utiliser ? Le travail de recherche qui est présenté dans ce mémoire vise à apporter des éléments de réponse. La contribution que nous proposons, à la fois théorique, méthodologique et pratique, est développée dans ce mémoire au travers quatre chapitres.

Le premier chapitre propose un état de l'art sur l'utilisation des modèles de Partie Opérative dans les travaux en Systèmes à Evénements Discrets (SED). Il est montré l'importance de disposer de modèles performants de la partie opérative quelque soit le domaine d'application (commande, diagnostic, identification et supervision). Les travaux fondamentaux en commande des SED reposent essentiellement sur une modélisation de la partie opérative au moyen des automates à états. Le constat effectué montre qu'aucune démarche structurée n'est proposée pour construire le modèle à états dans le cas des systèmes manufacturiers.

Le deuxième chapitre s'intéresse donc plus spécifiquement à la modélisation des systèmes manufacturiers, avec un objectif de commande, et tente de faire le lien avec les travaux théoriques en particulier dans le domaine de la synthèse de la commande par SCT (Supervisory Control Theory, Ramadge et Wonham). Il est montré que le point de vue théorique sur la partie opérative est dans ce cas purement fonctionnel sans lien avec le comportement physique ayant lieu. L'application de ces travaux fondamentaux à la commande des systèmes manufacturiers nécessite d'utiliser un modèle de partie opérative qui ne bride pas les évolutions possibles de la commande. Une première contribution théorique est proposée dans ce cadre. Une propriété de commandabilité est énoncée en vue de s'assurer que le modèle de partie opérative, représenté sous la forme d'automates à états, ne contient aucune restriction de l'évolution de la commande fournissant une vision partielle du comportement de la partie opérative. Toutefois, le problème de l'obtention du modèle de partie opérative reste posé. La démarche de conception, par une étude purement fonctionnelle du comportement de la partie opérative permet de décrire formellement uniquement la perception qu'a le concepteur de ce comportement. En revanche, elle est très éloignée de la conception physique du système et des choix technologiques qui ont été faits. Il nous semble que le modèle de la partie opérative peut être obtenu alternativement par une étude comportementale physique et non plus fonctionnelle du système automatisé. Une telle approche permet d'obtenir des modèles différents au sens où par exemple la commande n'est plus appréhendée sous forme d'ordonnancement d'opérations élémentaires mais sur les sollicitations et les retours de signaux échangés entre la partie commande et la partie

opérative. De plus, ces modèles peuvent être utilisés en ligne pour des applications autres que la commande, comme par exemple le diagnostic et la supervision.

Par conséquent, une démarche de modélisation de la partie opérative basée sur l'étude des composants physiques à l'aide de différents outils est proposée dans le chapitre 3. Cette contribution méthodologique permet de représenter au moyen d'automates à états, la technologie physique des matériels constituant la partie opérative. La démarche proposée est illustrée au moyen de deux applications réelles, dans le domaine du diagnostic et de la supervision industrielle, qui exploitent une cellule flexible (CellFlex) que nous avons mise en œuvre au sein du CReSTIC de l'Université de Reims Champagne-Ardenne. Ces deux exemples permettent de montrer l'intérêt de la démarche de modélisation mais aussi ses limites, la perception purement discrète de la partie opérative restant restrictive.

Le chapitre 4 propose enfin d'étendre la démarche de modélisation pour représenter les aspects hybrides (continus et événementiels) de la partie opérative. Pour cela, les Réseaux de Petri hybrides sont utilisés. La démarche développée consiste à assembler trois modèles (chaînes d'action, actionneurs, capteurs) pour obtenir une modélisation de la chaîne fonctionnelle. Cela revient à établir les états discrets de capteurs en fonction des commandes tout en tenant compte des lois d'évolution continues des grandeurs physiques caractéristiques des systèmes réels. Deux applications pédagogiques sont proposées pour illustrer la simulation des chaînes fonctionnelles. Dans les deux cas, un moteur de calcul auteur « WINSIM » a été utilisé en lien avec un logiciel de type SCADA pour la visualisation La première concerne un axe numérique utilisé dans les établissements d'enseignement (banc Emericc). La seconde application concerne la simulation d'une porte de garage automatisée avec prise en compte des inerties. Dans cet exemple, nous montrons comment le modèle hybride obtenu peut être intégré dans des maquettes virtuelles qui peuvent être un outil intéressant lors de la formation aux automatismes.

Le mémoire se termine en proposant des perspectives de recherche. Celles-ci concernent d'une part l'extension de la démarche proposée par la prise en compte du produit, et d'autre part, l'utilisation du modèle de partie opérative à tous les niveaux de la pyramide CIM. Enfin, les modèles proposés dans ce mémoire devraient être dans les années à venir intégrés dans le logiciel de simulation d'environnements industriels (ITS PLC Professional Edition) dédié à la formation. Il est intéressant de constater qu'un travail initialement conduit pour des applications industrielles manufacturières a aussi débouché sur des applicatifs pédagogiques originaux.

Chapitre 1
Utilisation des modèles de partie opérative dans les travaux en SED

Historiquement, l'automatisation est apparue suite à la mécanisation des systèmes de production et représente désormais un vaste domaine de recherche comportant différents champs d'application. Des outils formels ont été développés pour explorer et exploiter ce domaine de recherche. Ces outils permettent de proposer des méthodologies pour élaborer une commande, pour vérifier sa cohérence, pour diagnostiquer des pannes au sein d'un système automatisé, etc. Désormais, la représentation du comportement d'un système automatisé avec ces outils formels apparaît comme une étape incontournable dans l'ensemble des concepts, outils et méthodes proposés dans la littérature. Cependant, les développements réalisés à l'aide de ces outils demeurent difficiles à transférer dans l'industrie.

Dans ce chapitre, une analyse bibliographique présente pourquoi la partie opérative nécessite d'être prise en compte, quels sont les travaux existants dans ce contexte, les difficultés et la problématique autour de la représentation par un modèle formel du comportement de la partie opérative.

D'abord global, notre travail se focalisera plus particulièrement sur l'implémentation d'une solution pratique destinée à piloter les systèmes automatisés de production manufacturiers sur les aspects de commande de bas-niveau. Les différentes démarches de conception de la commande sont présentées en mettant en avant pourquoi et comment prendre en compte le comportement de la partie opérative.

1) Contexte des systèmes automatisés : pourquoi l'automatisation, attentes et problématiques

a) La mécanisation

Les systèmes de production ont considérablement évolué au cours du temps. La mécanisation puis l'automatisation des systèmes ont eu pour effet de reproduire à grande échelle des séries de produits similaires. La mécanisation a tout d'abord fourni des outils permettant de reproduire les opérations manuelles dans des fonctions globales de transformation, de transport ou de stockage du produit. La Figure 1.1 illustre les étapes successives qui ont permis d'automatiser les systèmes de production. Les produits traités par un système mécanisé ont pour origine une situation brute et doivent atteindre une situation dans laquelle le produit acquiert une valeur ajoutée. Le produit peut prendre différentes formes suivant les domaines d'application : il peut s'agir de matières premières solides, liquides ou gazeuses, de produits ayant déjà eu une valeur ajoutée (produits conditionnés en quantité élémentaire tels que les granulats, les plaques, les tubes…), des produits manufacturés possédant une valeur ajoutée destinés aux clients finaux (produits de grande consommation, produits prêts à être utilisés…).

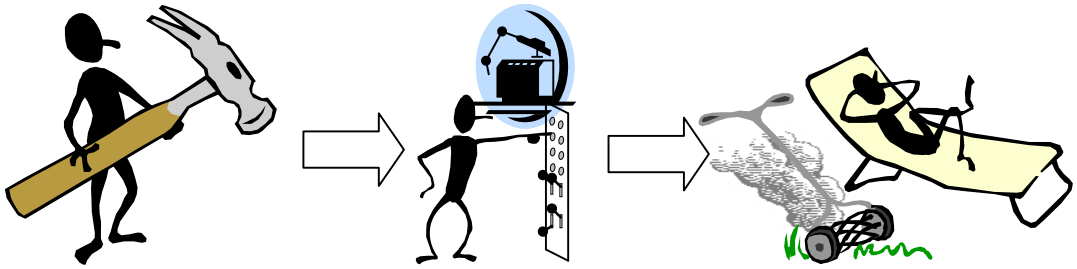


Figure 1.1 : évolutions des systèmes de production

Les domaines d'application utilisant la mécanisation comme solution technique sont très vastes. Chaque système mécanisé réalisant une fonction déterminée et connue, il est souvent nécessaire d'effectuer plusieurs opérations mécanisées successives pour obtenir une valeur ajoutée sur le produit. Ces différentes étapes peuvent être combinées ou synchronisées entre elles pour réduire la complexité. La synchronisation entre ces systèmes mécanisés peut prendre différentes solutions technologiques en fonction de nombreux paramètres liés au type de système de production impliqué, à l'énergie et aux technologies utilisées. Il s'agit d'évoquer ici l'utilisation de :

- La mécanique : Un système disposant de fonctionnalités synchronisées entre elles peut être automatisé avec des liens mécaniques. Tous les mécanismes à chaînes, câbles,

cames, courroies… permettent de synchroniser toutes les fonctions à partir d'une seule et même action. Dans cette classe de systèmes, on retrouve les systèmes à haute productivité tels que les systèmes d'embouteillages, de convoyage, etc.

- Les fluides : Les fluides hydrauliques ou pneumatiques peuvent aussi remplir des fonctions d'automatisme. Quelque soit le type d'énergie principal utilisé par le système, il est possible de modifier l'alimentation en énergie des actionneurs constituant le système mécanisé au cours de son fonctionnement pour qu'il réponde aux besoins attendus. Dans les systèmes incluant une cuve de liquide à niveau constant, la présence d'un flotteur permet, par exemple, de déclencher l'ouverture d'une vanne d'alimentation et la remise au niveau de remplissage de la cuve.
- Le courant électrique : Les commandes électriques sont de plus en plus présentes dans les systèmes automatisés. Cette énergie peut plus facilement être véhiculée et commutée pour réaliser les fonctionnalités auxquelles le système automatisé doit répondre. Les systèmes de relayage électromagnétiques peuvent réaliser la commande des systèmes d'éclairage temporisés, de régulation de température… Ces solutions sont utilisées sur des systèmes ayant un comportement logique relativement simple avec peu d'interactions entre les différentes fonctionnalités.

b) Évolution vers l'automatisation

L'automatisation constitue une seconde étape : en agençant les opérations mécanisées les unes avec les autres, le système de production devient autonome et son comportement systématique. Les interventions humaines n'ont pas lieu d'être dans l'utilisation continue du système. Les apports de l'automatisation par rapport à un système mécanisé sont multiples : l'automatisation permet d'optimiser la cadence, la qualité et le coût de production.

Avec l'évolution des technologies, de nouvelles solutions se sont développées avec des systèmes de commande beaucoup plus élaborés. Il ne s'agit plus de synchroniser quelques fonctions élémentaires les unes avec les autres mais de piloter l'ensemble du système avec un ensemble de matériels totalement dédiés à la commande. Il y a maintenant une dissociation complète entre la technologie utilisée pour la commande et l'énergie utilisée pour produire l'action sur la partie opérative. La commande est réalisée à l'aide de matériels dédiés et spécialisés. Deux évolutions technologiques majeures sont apparues sur la commande :

- Les composants électroniques : La miniaturisation des composants électriques a permis de répondre aux problèmes plus complexes tout en produisant des solutions moins coûteuses dans de nombreuses applications. L'électronique facilite la réalisation de systèmes automatisés complexes à l'aide de composants réalisant des systèmes de commande de base peu onéreux. Les portes logiques, les bascules, les circuits intégrés, les amplificateurs opérationnels… sont souvent utilisés dans de nombreux systèmes automatisés.
- L'informatique industrielle : Avec l'augmentation de la complexité des systèmes automatisés et du paramétrage nécessaire du fonctionnement, de nouvelles solutions

technologiques dérivées des solutions utilisées en informatique sont apparues. Les Automates Programmables Industriels (API) permettent une grande flexibilité dans les modifications de comportement à apporter au système. Cet équipement doit être configuré et programmé pour définir le comportement souhaité du système automatisé. Il apporte des atouts importants pour les systèmes comportant une grande complexité et fournit une facilité de mise en œuvre et une évolutivité lors des modifications de comportements à apporter à la commande. Il nécessite une console de programmation pour indiquer les actions que la partie opérative doit assurer.

c) Constituants des systèmes automatisés

Les systèmes automatisés sont constitués de deux parties ayant de fortes interactions entre elles. La Figure 1.2 illustre ce découpage. La partie opérative correspond à la partie mécanisée du système agrémentée de l'interface nécessaire à son pilotage par la partie commande. Les capteurs sont disposés sur la partie physique du système pour mesurer la situation courante du système et servent donc à l'observation de ses évolutions. Les capteurs adaptent l'information physique mesurée pour convertir cette information dans un signal adapté à la partie commande suivant le type d'énergie utilisée.

Pour que le système automatisé fonctionne, la partie commande doit lancer l'exécution d'opérations élémentaires. Les actionneurs en association avec les préactionneurs remplissent cette fonction. Les actionneurs sont alimentés par une source d'énergie et exécutent des opérations élémentaires en transformant l'énergie d'alimentation en énergie de puissance permettant l'exécution d'une action. Les préactionneurs permettent de faire le lien entre l'alimentation en énergie de puissance et les informations utilisées par la partie commande.

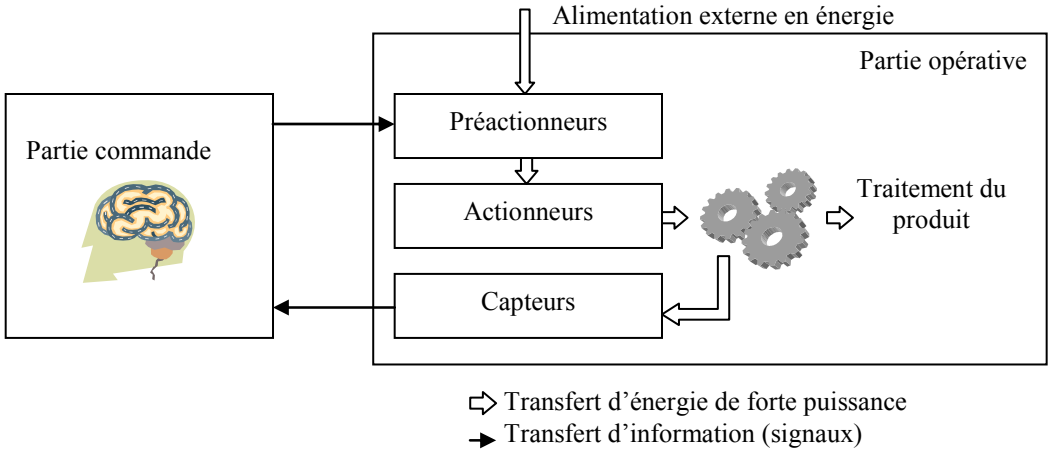


Figure 1.2 : découpage d'un système automatisé

L'ensemble du système automatisé effectue un traitement du produit. Le sous-ensemble de la partie opérative, qui exécute une opération élémentaire à partir du signal envoyé par la partie commande, correspond à une chaîne d'action. Le sous-ensemble mesurant une grandeur

physique sur la partie opérative jusqu'au signal reçu par la partie commande correspond à une chaîne d'acquisition.

d) La problématique actuelle

Aujourd'hui, la variabilité des produits est croissante et impose une plus grande flexibilité des moyens de production. Il ne s'agit plus de reproduire à l'infini un produit mais de prendre en compte la personnalisation du produit suivant les souhaits du client. Du point de vue de l'automatisation, ces changements ne sont pas sans conséquence : cette diversité des produits fabriqués et la complexité inhérente des moyens de production engendrent une grande complexité dans l'agencement des opérations. Le système automatisé doit donc être capable d'effectuer des opérations élémentaires paramétrées suivant les caractéristiques attendues du produit final.

L'organisation nécessaire dans le suivi individualisé de chaque produit implique également la mise en œuvre de technologies pour l'heure utilisées dans les domaines du traitement de l'information. Ces technologies sont liées à la traçabilité et au suivi en interaction avec l'automatisation.

Toute la problématique autour de l'automatisation des systèmes de production, de la diversification des produits et de la nécessité de tracer toutes les évolutions du produit fait l'objet de nombreux travaux de recherche selon des points de vue différents en fonction des objectifs à atteindre.

2) Systèmes manufacturiers et Systèmes à Evènements Discrets (SED)

L'automatisation des systèmes de production peut être abordée selon le point de vue de la régulation, de l'organisation de la charge de travail à répartir entre différentes machines ou le pilotage des lignes de fabrication de produits assemblés. Ces différents points de vue font l'objet d'axes de recherche souvent très différenciés.

Parmi l'ensemble des systèmes automatisés, nous nous intéressons, dans ce mémoire, plus particulièrement aux systèmes automatisés de ligne de production manufacturière pilotés par des API. Les API sont très souvent utilisés lorsque le système à piloter comporte de nombreuses entrées/sorties de type « tout ou rien ». Les systèmes manufacturiers contrôlent des séquences d'opérations élémentaires les unes après les autres. Ils utilisent généralement les API pour les capacités offertes de traitement des données, leurs simplicité de programmation... Cependant, suivant les applications et le domaine dans lequel elles sont mises en place, il existe de nombreuses difficultés à considérer, telles que le nombre important de capteurs et d'actionneurs à traiter, l'enchaînement complexe d'opérations, l'abondante diversité des produits fabriqués, l'importance de la sûreté de fonctionnement pour les opérateurs et le process, etc.

Les Systèmes à Evénements Discrets (SED) recouvrent plusieurs domaines d'applications tels que la robotique, la logistique, les systèmes de transport (aérien, ferroviaire, ...), les réseaux de communication, l'informatique, les circuits logiques et celui qui nous intéresse à savoir les systèmes de production manufacturière.

Ainsi, les nombreux concepts (techniques, théories, méthodes, outils, modèles et langages) élaborés pour les SED ont pour objectif d'améliorer la qualité et de maîtriser la complexité croissante de la conception et du développement de ces systèmes. Ils ne se limitent donc pas à la conception pure d'un modèle de commande mais s'intéressent également à la prise en charge de l'ensemble du comportement du système et des opérateurs humains en interaction avec la Partie Commande (PC) et la Partie Opérative (PO) du système automatisé. Par conséquent, les travaux sur la conception de la commande, sur le diagnostic, sur la surveillance vont s'intéresser respectivement à l'élaboration du modèle SED à implanter dans l'API, à la cohérence du comportement réel mesuré sur le système avec un modèle de comportement théorique et à la reconstitution d'informations simplifiées extraites du flot de données.

a) La commande

La norme [IEC-61508] définit les besoins à prendre en compte dans la réalisation d'une commande sûre de fonctionnement. Ce sont principalement les travaux de vérification/validation et de synthèse qui s'intéressent à l'obtention d'une commande sûre.

- **Les travaux de vérification/validation :**

L'objectif recherché dans ces travaux est de proposer un modèle de commande et de s'assurer, avant l'implantation dans le système réel et l'exploitation industrielle, que le modèle de commande vérifie des propriétés de sûreté et/ou de bon fonctionnement. La validation et la vérification prennent différentes acceptions suivant les auteurs.

- Dans le modèle de conception du cycle en V [Perrin, 2001] présenté sur la Figure 1.3, à chacune des phases de conception est associée une phase de test. La vérification peut être assimilée à l'analyse du modèle de conception dans chaque phase. Lorsque le modèle de conception répond à certains critères, la phase de conception est dite validée ce qui signifie qu'elle a atteint le niveau d'exigence demandé. La vérification a alors pour but de s'assurer de la conformité du modèle de la commande d'un point de vue sémantique et de ses propriétés intrinsèques. On s'assure ainsi que le modèle de commande proposé assure bien une fonctionnalité. La validation se place plus dans un contexte d'efficacité. Dans ce cadre, la validation a pour but de montrer que le modèle de commande proposé est bien cohérent vis-à-vis de la fonctionnalité souhaitée.

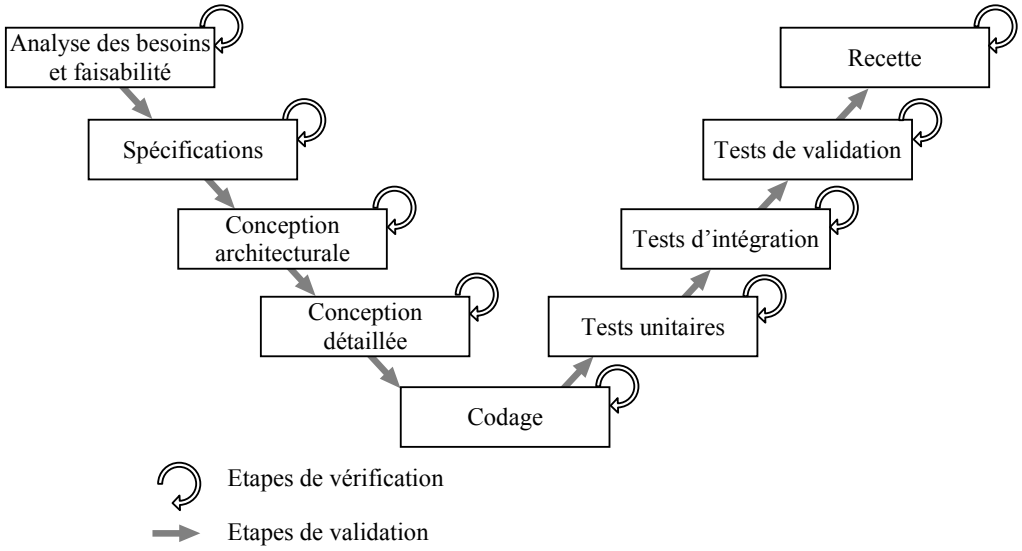


Figure 1.3 : cycle de conception en V

- La seconde variante consiste à différencier la validation de la vérification lorsqu'il y a prise en compte du comportement réel du système (Figure 1.4). La vérification ne s'intéresse qu'à la cohérence intrinsèque du modèle de commande. La validation détecte les comportements non souhaités et les réactions du système face à certaines sollicitations. Les travaux de vérification et de validation par les méthodes formelles (model checking et theorem proving) sont donc nombreux [Dimitrova, 2005], [Frey, 2000], [Frey, 2002], [Klein, 2003], [Raush, 1998], [Rossi, 2003], [Roussel, 1994], [Roussel, 2006].

  Le modèle de commande est proposé par le concepteur en fonction du cahier des charges informel préalablement établi décrivant le comportement que le système automatisé doit posséder. L'expert intervient pour construire un modèle de comportement du système automatisé soumis aux sollicitations de la commande. L'expert énonce également les propriétés formelles décrivant le comportement nécessaire du système pour être sûr de fonctionnement. La construction de ces modèles nécessite une bonne connaissance du processus de fabrication réalisé par le système automatisé.

  Dans ce cadre, la vérification est assimilée à l'évaluation du modèle de commande sur des critères indépendants du fonctionnement du système. La validation consiste alors à tester si les contraintes de fonctionnement décrites formellement par l'expert sont respectées.

Il existe différentes méthodes pour confronter le modèle de commande avec les spécifications du comportement du système définies par l'expert. Si le modèle de comportement du système défini par l'expert décrit toutes les situations successivement atteintes par le système, les outils de vérification tels que le model-checking ou le theorem-proving permettent de vérifier qu'aucune des situations atteintes par le système dans son utilisation ne violera une contrainte provenant des spécifications du comportement décrites par l'expert. Le modèle de

comportement du système piloté est obtenu en contraignant le modèle de comportement du système automatisé avec le modèle de commande proposé par le concepteur. La validation du comportement du modèle de commande initialement proposé est obtenue en mettant à l'épreuve le modèle de comportement commandé avec les spécifications comportementales décrites par l'expert (Figure 1.4).

Les difficultés liées à l'utilisation de ce type de méthodes résident dans l'explosion combinatoire de la description du comportement du système et dans la difficulté à prendre en compte le comportement du système dans l'écriture des propriétés formelles.

Les systèmes décrits par des modèles dynamiques représentant les lois physiques ne permettent pas d'explorer l'ensemble des situations atteignables par le système. Pour valider le comportement dans ce contexte, l'expert détermine des scénarios d'évolutions que le système doit être capable de respecter. Il s'agit d'une approche par simulation du comportement du système et d'analyse des évolutions. Les difficultés rencontrées avec cette approche sont l'exhaustivité des situations et les difficultés de modélisation. En effet, il ne s'agit dans cette méthode que d'analyser le comportement du système dans des contextes isolés suivant des lois d'évolution plus ou moins réalistes.

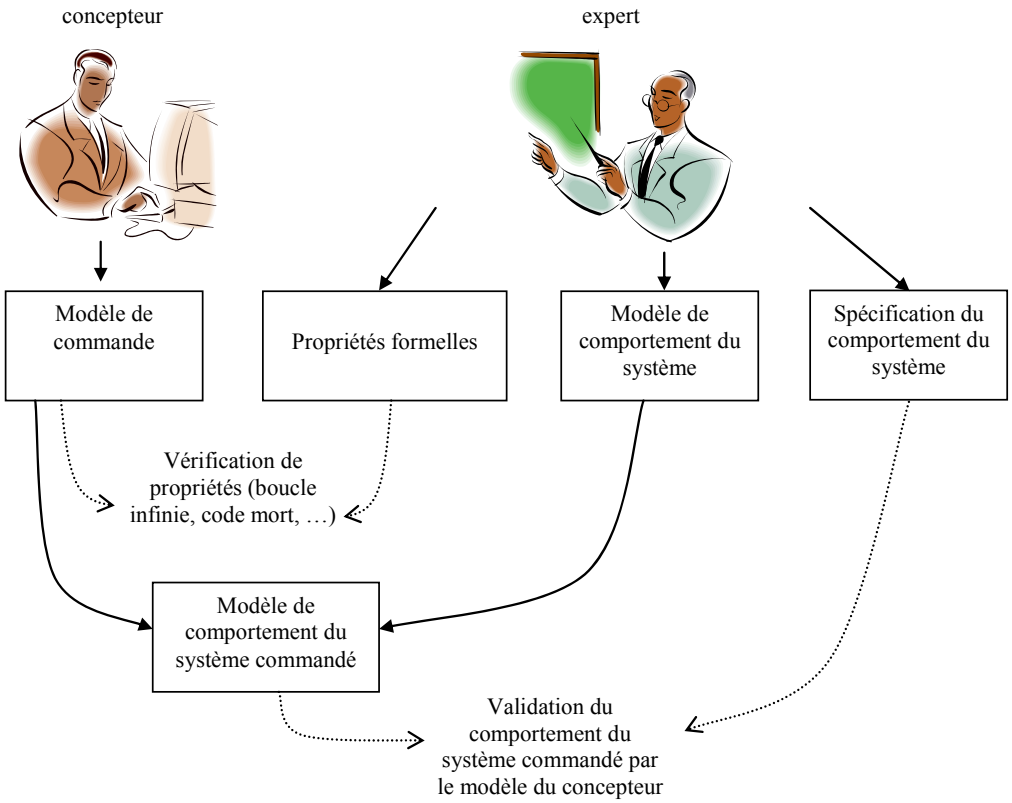


Figure 1.4 : validation de la commande par vérification

- **Les travaux de synthèse :**

Ces travaux permettent de déduire le comportement de la commande en réduisant le comportement complet du système avec des spécifications qui traduisent le comportement

souhaité et interdisent le comportement incorrect. Le modèle de commande obtenu est supposé sûr puisqu'il est construit à partir de la spécification du comportement attendue du système automatisé (Figure 1.5). La synthèse consiste donc à restreindre le comportement d'un système à l'aide de spécifications.

A partir du travail d'expertise réalisé, le modèle de comportement du système proposé par l'expert peut être réduit au seul comportement en accord avec les spécifications de comportement qu'il a défini. Le modèle obtenu est alors le modèle de comportement du système contraint par les spécifications mais il n'inclut pas toute la commande : des indéterminismes subsistent généralement dans la description effectuée par l'expert. Néanmoins, ce modèle de comportement contraint et non commandé peut être utilisé pour vérifier une commande réalisée par le concepteur ou pour en extraire un modèle de commande à partir de critères d'optimisation définis par le concepteur.

- ✓ Dans le premier cas de figure, les propriétés intrinsèques du modèle de commande proposé par le concepteur sont vérifiées pour tester s'il n'y a pas d'incohérence dans la définition du modèle de commande puis le modèle de commande est confronté au modèle de comportement contraint non commandé. Si le modèle de commande est inclus dans le modèle de comportement contraint non commandé, le modèle de commande proposé par le concepteur est sûr.
- ✓ La seconde possibilité consiste à demander au concepteur quels sont les critères d'optimisation à définir pour résoudre les indéterminismes présents dans le modèle de comportement contraint non commandé. Le modèle obtenu est un modèle de commande construit à partir du fonctionnement sûr du système et optimisé par les critères définis. Cette méthode n'est cependant pas sans défaut puisque l'élaboration du modèle de comportement complet du système est délicate à cause des proportions qu'elle peut prendre : les règles peuvent être difficiles à écrire et une erreur dans leur écriture produit une défaillance indécelable… De plus, le modèle obtenu peut comporter plusieurs solutions. Des critères d'optimisation doivent permettre de sélectionner un modèle de commande parmi les modèles satisfaisants les contraintes. Le modèle peut aussi être utilisé pour valider un modèle de commande proposé et ainsi tester si le modèle de commande répond bien aux contraintes de fonctionnement et respecte le fonctionnement intrinsèque du système. [Achour, 2004], [Carré-Ménétrier, 2002], [Makungu, 1999], [Ramadge, 1989], [Tajer, 2005], [Wonham, 1987], [Wonham, 2002], [Zaytoon, 1999].

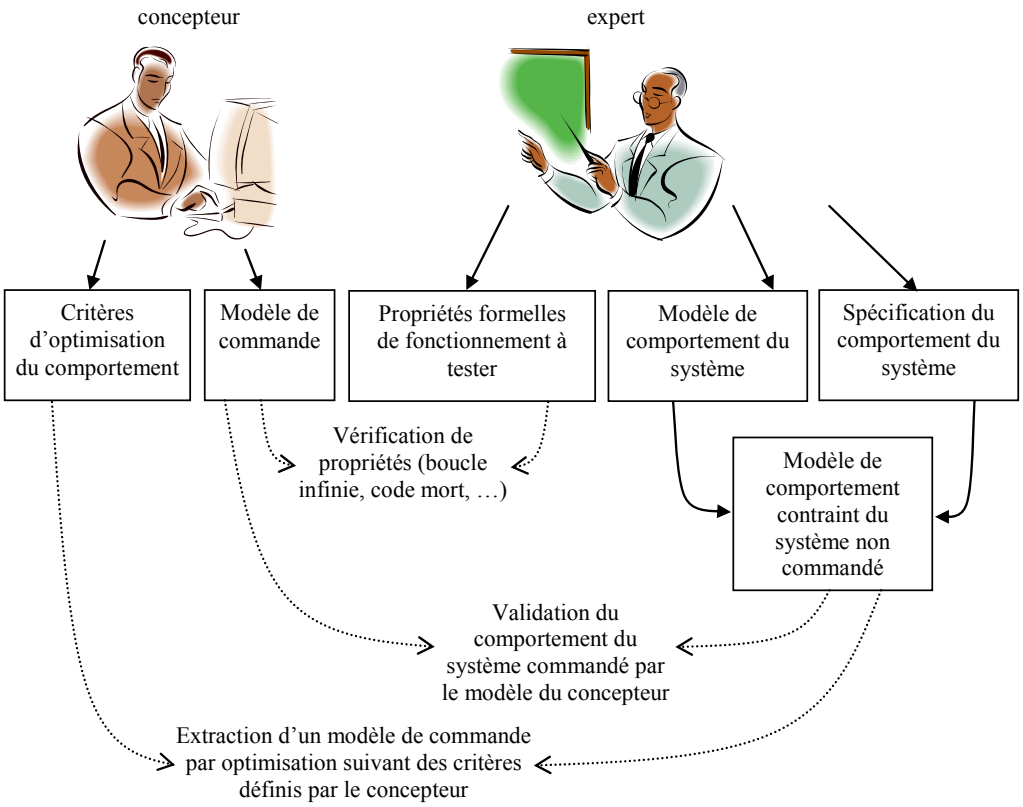


Figure 1.5 : démarche de validation ou de construction de la commande par synthèse

- **La simulation comportementale :**

  La simulation peut permettre de tester un modèle de commande sur un simulateur de partie opérative ayant pour but de réagir comme la partie opérative sans prendre les risques de son utilisation. Le modèle de partie opérative doit donc contenir tout le comportement de la partie opérative pour fournir une interprétation réaliste des résultats. L'outil Simac[1] propose de représenter le comportement de la partie opérative au travers un modèle sous la forme d'axes par émulation du comportement dynamique global du système. Chaque axe réalisant une fonction élémentaire est défini par les caractéristiques physiques (dimensions, poids), les équipements installés (capteurs), les mouvements autorisés et les technologies d'actionnement (monostable, bistable, …). Cet outil permet, après avoir modélisé l'ensemble de la partie opérative composant le système automatisé, de tester les modèles de commande avant l'implémentation dans l'API cible. Néanmoins, la difficulté de modélisation, notamment par le manque de méthodologie, ne facilite pas l'utilisation de cet outil. De plus, la modélisation de l'ensemble complet du comportement de la partie opérative nécessite un travail d'expertise important pour un résultat souvent mitigé à cause de la complexité du système automatisé.

[1] Simac (Prosyst) : progiciel de simulation de partie opérative (http://www.prosyst.fr)

b) Le diagnostic

Au-delà des méthodologies de conception de la commande proprement dite, d'autres méthodes ont été développées au niveau des systèmes automatisés de production pour améliorer la sûreté de fonctionnement.

Les applications de diagnostic s'intéressent plus particulièrement à l'étude de la cohérence des informations connues de la commande pour vérifier le bon fonctionnement du système ou à la détermination des causes et des conséquences d'une défaillance d'un élément du système. A partir des données connues de la partie commande et d'un modèle de comportement du système, la détection détermine les fautes pouvant avoir lieu. Le diagnostic détermine ensuite les causes probables de cette faute et les conséquences qu'elles peuvent avoir sur le système automatisé.

Ces travaux sont donc généralement constitués de deux étapes :

- La détection : il s'agit de détecter un comportement anormal du système à partir des seules informations connues par le système au travers des commandes et des capteurs. La détection est réalisée en comparant le comportement du système réel avec un modèle de comportement théorique. Le traitement des écarts permet d'identifier des comportements non souhaités dans le fonctionnement du système [Combacau, 2000a], [Combacau, 2000b].
- La recherche des causes et des conséquences d'une défaillance : il s'agit de déterminer pourquoi le système a eu un comportement défaillant. A partir de cette cause, on peut également déterminer quels sont les comportements résultants de la défaillance [Darkhovski, 2003].

Pour effectuer le diagnostic d'un système automatisé, un modèle du procédé est associé avec un module de diagnostic, souvent appelé diagnostiqueur, pour collecter les données du procédé et prendre une décision sur l'état de fonctionnement du procédé [Philippot, 2006] (Figure 1.6). Suivant la disponibilité des données sur le procédé, une structure de prise de décision est définie :

- La structure centralisée associe ainsi un modèle global du procédé à un seul diagnostiqueur global.
- La structure décentralisée utilise un modèle de procédé global avec des diagnostiqueurs locaux indépendants prenant des décisions locales à partir des données à disposition. Un coordinateur est chargé de collecter les décisions de chaque diagnostiqueur pour prendre la décision finale.
- La structure distribuée utilise un modèle de procédé représenté au travers de composants locaux. Chaque composant est observé par un diagnostiqueur local. Un protocole de communication entre diagnostiqueurs est défini pour leur permettre locaux d'échanger leurs décisions. Les diagnostiqueurs locaux prennent leur décision en fonction de leur observation et des décisions des autres diagnostiqueurs. Aucune

décision globale n'est construite par un coordinateur puisque chaque diagnostiqueur local détermine sa décision en fonction de toutes les informations à sa disposition.

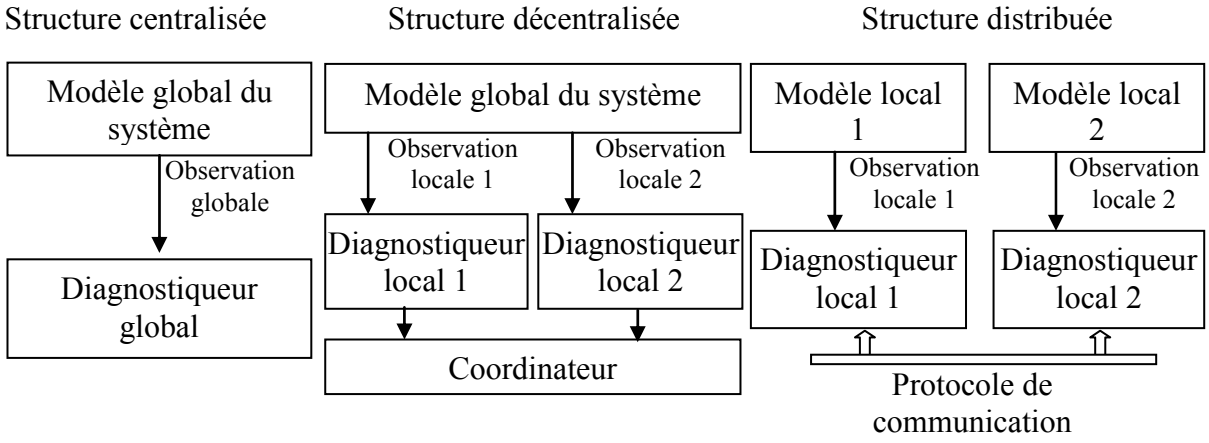


Figure 1.6 : structures de prises de décision utilisées en diagnostic

Différentes approches de construction du module de diagnostic sont possibles [Combacau, 2001] :

- L'approche intégrée insère le diagnostic dans la conception de la commande. Les décisions prises peuvent être utilisées dans la commande pour réagir rapidement aux aléas de fonctionnement.
- L'approche séparée dissocie totalement le module de diagnostic de la commande. Il n'y a donc aucune dépendance entre les fonctions de commande et de diagnostic, permettant ainsi d'utiliser des outils différents.
- L'approche mixte combine les deux approches. Elle consiste à utiliser des informations issues du modèle de commande pour construire le diagnostiqueur. Cette approche s'apparente à une supervision de la commande pour le diagnostic.

c) L'identification

L'identification des SED consiste en l'apprentissage progressif du comportement du système au cours de son fonctionnement. L'application enregistre les évolutions successives de la partie opérative. L'étude des scénarios enregistrés permet de générer un modèle représentant le comportement du système en cours de fonctionnement.

Le modèle obtenu peut être utilisé pour recréer un modèle de commande sur un système existant. Il peut être également un modèle de base pour les autres applications.

Les principaux problèmes rencontrés par ces travaux sont liés au problème d'exhaustivité du comportement du système et à la délimitation entre les comportements considérés comme répétables ou non.

Ce type d'application est particulièrement adapté pour le rétro-engineering consistant à recréer un modèle de commande au comportement similaire au modèle implanté dans le

système automatisé. [Bekrar, 2006], [Klein 2005], [Meda-Campaña, 1998], [Meda-Campaña, 2000], [Meda-Campaña, 2003]

d) La supervision industrielle

La supervision industrielle consiste à représenter pour un opérateur humain ou pour un système externe, une image du fonctionnement du système en vue de son pilotage à distance.

La supervision au sens « supervision industrielle » constitue l'interface de dialogue entre le système automatisé d'une part et l'opérateur surveillant le système d'autre part. Cette interface, généralement appelée Interface Homme Machine (IHM), permet la visualisation des données contenues dans le système de commande et l'acquisition des demandes de l'opérateur. Des difficultés liées à la présentation d'informations compréhensibles par un opérateur nécessitent de prendre en compte les capacités cognitives de chaque utilisateur et le flot de données fourni par la partie commande [Riera, 2003a].

La supervision industrielle peut aussi utiliser les informations pour l'historisation et l'exploitation de bases de données. L'intégration des données de production provenant du système de commande dans le système d'information global de l'entreprise fait également partie de travaux de recherche. La reconstruction d'informations exploitables passe nécessairement par une connaissance du fonctionnement du système automatisé. Un modèle de fonctionnement du système automatisé est nécessaire dans cette approche.

En fonction de l'objectif de l'application, des outils de modélisation et des hypothèses prises en compte pour la modélisation, le modèle de partie opérative prend différents niveaux d'abstraction comportant plus ou moins d'information de nature différente. Dans les niveaux les plus bas, le modèle prend en considération les évolutions des capteurs et des commandes d'actionneurs. Il s'agit là des informations de plus bas niveau qui sont identifiées. Ces informations peuvent toutefois être agrégées et ainsi représenter des situations dans lesquelles se trouve le système. Les mouvements, les positions… du système qui ne sont pas être directement mesurables avec un capteur ou extrait d'une commande sont représentables par le traitement des informations de plus bas niveau. Enfin, ces informations permettent de recomposer des fonctions élémentaires dans lesquelles le système est susceptible de se trouver. Les fonctions peuvent prendre plusieurs états préalablement déterminés tels qu' « en attente », « en cours », « terminé », « en panne »… Le comportement de ces fonctions souvent assez abstraites n'est généralement pas explicité et les liens avec les données réellement échangées avec le système ne sont pas représentés.

Le paradigme Computerized Integrated Manufacturing (CIM) représente les différents niveaux d'abstraction et de décision dans le fonctionnement d'un système de production totalement automatisé. Ce concept d'intégration d'une structure combinant commande et communication vis-à-vis des systèmes manufacturiers, a été initié par Dr. Joseph Harrington. Il s'est ensuite largement répandu dans les années 1980. Cependant, les évolutions des systèmes de production avec les cellules flexibles, les outils de conception/modélisation, l'amélioration des protocoles de communication ont modifié la définition originelle de CIM. Il s'agit désormais de l'intégration des nouvelles technologies dans des systèmes flexibles

pour réaliser les objectifs de l'entreprise. Le concept CIM intègre désormais la gestion d'entreprise telle que les fonctions logistique, approvisionnement, finance, relation clients / fournisseurs…

Le concept CIM est souvent représenté sous la forme d'une pyramide comportant quatre niveaux. A chaque niveau est associé un niveau de décision et une visibilité globale du système. Ce concept permet de représenter les niveaux hiérarchiques d'information au sein de l'entreprise. Il ne détaille pas quelles sont les données à instaurer à chaque niveau mais propose un cadre à chacun de ces niveaux. Le niveau 0, le plus bas de la pyramide, correspond aux capteurs/actionneurs. Le niveau 1 est associé à la cellule tandis que le niveau 2 correspond à l'atelier. Enfin le niveau 3, le niveau le plus élevé de la pyramide, est le niveau de gestion. Ce concept fait apparaître une hiérarchie très forte dans l'organisation et le traitement de l'information. D'un point de vue contrôle/commande, la mise en œuvre est réalisée par des technologies totalement identifiables : le niveau 1 est réalisé par l'automate de contrôle-commande ; le niveau 2 est mis en œuvre à l'aide des applications de supervision ; le niveau 3 utilise des solutions logicielles de gestion. Il convient aux systèmes centralisés dans lesquels les échanges d'information au sein d'un même niveau ne sont pas aisés.

A chacun de ces niveaux de représentation de l'information, il faut utiliser des outils de modélisation adaptés. Les outils représentant le comportement du système peuvent aussi prendre ces différents points de vue en fonction du niveau de représentation souhaitée par le modèle.

Les outils informatiques dans le domaine des technologies de l'information et de la communication permettent de stocker, traiter et diffuser les informations au travers les technologies Web telles que les bases de données SQL, le formatage XML, le standard de communication OPC, le réseau Ethernet et le protocole TCP/IP… [Zecevic, 1998]. Les architectures trois-tiers segmentent l'exécution d'une application en trois dispositifs aux fonctionnalités dédiées : un client exécute l'interface, un serveur est dédié au stockage de l'ensemble des données utilisées et un serveur de calcul est dédié au traitement de l'information en vue d'extraire les informations à diffuser sur les interfaces en fonction des données présentes dans le serveur de base de données [Qui, 2000]. Les logiciels intégrés de supervision industrielle tels que les SCADA (Supervisory Control And Data Acquisition) sont fortement imprégnés de ces technologies.

La principale difficulté de la supervision industrielle réside désormais plus dans la transformation des données en informations pertinentes que dans l'accès aux données. La Figure 1.7 adaptée de [Sheridan, 1988] et [Millot, 1998] illustre les échanges d'informations entre les différents niveaux de la pyramide.

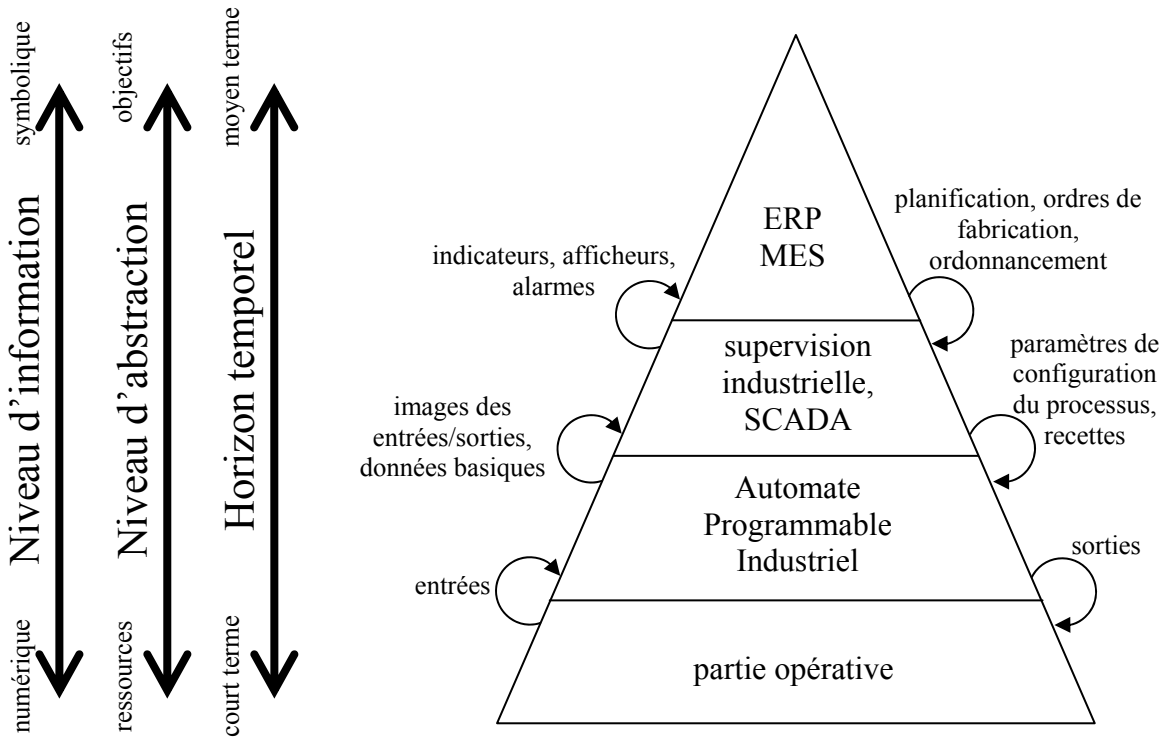


Figure 1.7 : illustration des échanges d'informations au sein du concept CIM

La supervision industrielle se place au croisement de plusieurs disciplines : l'automatique (continue et discrète) pour l'étude du système, les sciences cognitives pour prendre en compte le comportement de l'Homme et l'informatique pour le traitement. L'opérateur est informé de l'état du système et des disfonctionnements au travers d'Interfaces Homme-Machine (IHM). Les interfaces écologiques [Vicente, 1995], les interfaces à perception directe [Lambert, 1998] et l'affichage des données en masse [Boussoffara, 1998] sont différentes manières de présenter les données à l'opérateur avec pour objectif d'améliorer la conscience de la situation de la part de l'opérateur [Endsley, 1988]. Ces interfaces sont utilisées sur les processus continus. La principale difficulté réside dans l'extraction de l'état global du système à partir de l'observation de nombreuses variables analogiques réagissant à des boucles de régulation. Ces concepts peuvent être repris pour l'étude des systèmes à évènements discrets manufacturiers. Les systèmes à évènements discrets n'ont pas les mêmes caractéristiques : ils comportent un nombre important de variables booléennes dont le comportement est lié ensemble. L'apport du modèle de partie opérative est donc de générer des informations facilement interprétables à l'opérateur pour favoriser sa perception.

e) Etudes des systèmes manufacturiers et modélisation de la partie opérative

A travers la présentation précédente, il apparaît clairement que la prise en compte d'une modélisation du comportement du système physique est indispensable, même si cette modélisation n'est pas aisée à obtenir. Par exemple, il est tout à fait possible que le modèle théorique de la commande d'un système soit sans blocage après une simple procédure de vérification. Confrontée au système réel, il est également tout à fait possible que la commande engage la partie opérative dans une situation où elle ne pourra pas délivrer la réaction attendue par la dite commande. En fait, seule une réelle connaissance de l'état de la partie opérative et de son comportement permettra de traiter, dans ce cas précis, le problème de blocage.

Parmi tous les apports, certains travaux constituent des cadres formels puissants mais qui restent malheureusement difficilement utilisables sur des applications industrielles.

L'objectif de ces travaux de thèse est, par conséquent, de s'intéresser à une problématique permettant une exploitation des résultats à l'aide des outils couramment utilisés dans l'industrie. L'étude bibliographique présentée à la suite permet de montrer comment est, jusqu'à présent, appréhendée dans la littérature la modélisation de la partie opérative dans différents domaines des SED.

Nous nous focaliserons dans la suite de ce chapitre sur l'élaboration de la commande qui ne peut pas être sûre dans son fonctionnement sans prise en compte du comportement de la partie opérative. Ce n'est pas dans ce chapitre que seront abordés les modèles par eux-mêmes mais dans le chapitre suivant.

3) Etude bibliographique des travaux liés à la prise en compte de la partie opérative en conception de la commande

a) Problématiques de la commande

L'étude du comportement de la commande des systèmes automatisés fait l'objet de nombreux travaux. L'objectif initial est toujours le même : il s'agit de générer des ordres nécessaires à la partie opérative (émission des commandes) en fonction de compte-rendus (réception des informations capteurs). Le fonctionnement du système automatisé est totalement tributaire du comportement de la commande. La partie opérative peut avoir des comportements indésirables générant des dégâts matériels ou figer le système automatisé si le modèle de comportement de la commande n'est pas correctement établi.

L'élaboration du comportement de la commande reste néanmoins une opération particulièrement difficile à effectuer car le fonctionnement de la partie opérative est quant à lui totalement tributaire du comportement de la commande. Les contraintes auxquelles le fonctionnement de la partie opérative doit satisfaire, peuvent être sécuritaires ou

fonctionnelles. L'élaboration du comportement de la commande nécessite des phases de sécurisation et/ou d'optimisation du fonctionnement.

Il existe deux méthodes différentes dans l'élaboration de la commande. La preuve a posteriori permet de montrer qu'une solution proposée par le concepteur remplit les contraintes de fonctionnement. L'autre méthode d'élaboration du comportement de la commande est la construction a priori consistant à construire le modèle de commande à partir de la connaissance du comportement du système et à la restriction du comportement vis-à-vis des contraintes.

b) Vérification et validation

Comme nous l'avons vu, la validation par vérification consiste à demander au concepteur de proposer une solution de comportement de la commande pour ensuite s'assurer que la solution proposée ne viole pas les contraintes de fonctionnement. La validation permet donc de faire la preuve a posteriori que la solution proposée par le concepteur est cohérente avec le fonctionnement attendu. Les travaux utilisant cette approche se déroulent en deux temps. La vérification teste des propriétés intrinsèques du comportement de la commande notamment structurelles. La validation ajoute à la vérification le test de la cohérence du comportement de la commande vis-à-vis du fonctionnement de la partie opérative [Roussel, 1996], [Zaytoon, 1997].

Dans les travaux de validation, le modèle formel du comportement de la partie opérative soumis au comportement de la commande proposée par le concepteur est confronté à des spécifications décrivant les contraintes de fonctionnement. Le « model checking » et le « theorem proving » sont des outils permettant d'affirmer ou d'infirmer que le modèle de commande vérifie les spécifications intrinsèques. Cette démarche nécessite que le concepteur fournisse le modèle du comportement de la partie opérative restreint par le modèle de comportement de la commande qu'il a proposé. Ce modèle peut être obtenu en combinant un modèle de comportement de la commande avec un modèle de partie opérative complet. La réalisation du modèle de spécifications pose également problème lorsqu'il s'agit de formaliser l'idée du concepteur et de vérifier l'exhaustivité des spécifications.

[Machado, 2006] a montré les apports d'un modèle de partie opérative dans la vérification de la commande. L'utilisation d'un modèle de partie opérative permet en effet de prendre en compte le fonctionnement nominal du système automatisé. Sans l'utilisation du modèle de partie opérative, la vérification prend en considération toutes les situations possibles que pourrait prendre la partie opérative y compris des situations n'ayant aucun sens physique dans le cadre de l'utilisation sur un système réel.

Le modèle de partie opérative permet donc de réduire l'étude du comportement de la commande au comportement de fonctionnement nominal de la partie opérative défini dans le modèle. Suivant le type de propriété à tester, l'utilisation d'un modèle de partie opérative a des incidences sur le résultat. Lorsqu'il s'agit de montrer que la commande ne doit pas atteindre une situation dans laquelle des ordres ne doivent pas être émis simultanément (Figure 1.8, propriété A), le modèle de partie opérative doit être occulté. L'émission d'ordres

simultanés n'est pas dépendante de la situation dans laquelle peut se trouver le système. De plus, en cas de fonctionnement du système non pris en compte dans le modèle de partie opérative, il n'est plus possible de garantir la propriété si la vérification a été effectuée avec ce modèle de partie opérative. Le modèle de partie opérative restreint l'ensemble des états atteignables lors du fonctionnement du système et peut donc fausser la propriété. En revanche, lorsqu'il s'agit de vérifier qu'il est toujours possible d'atteindre une situation dans l'ensemble des états pouvant être atteints (Figure 1.8, propriété B), la validation de la propriété avec un modèle de partie opérative est plus restrictive. L'utilisation du modèle de partie opérative a pour effet de restreindre les évolutions futures du système automatisé aux seules évolutions cohérentes avec le fonctionnement du système automatisé. Dans ce cas, l'utilisation du modèle de partie opérative réduisant les situations atteignables aux seules situations cohérentes, est plus contraignante pour la vérification de la propriété. La recherche d'une situation future atteignable par le système évoquée par ce type de propriété nécessite de ne pas prendre en considération l'ensemble des évolutions cohérentes et incohérentes que pourrait avoir le système automatisé. Une telle propriété vérifiée sans le modèle de partie opérative ne l'est plus forcément avec la prise en compte de celui-ci.

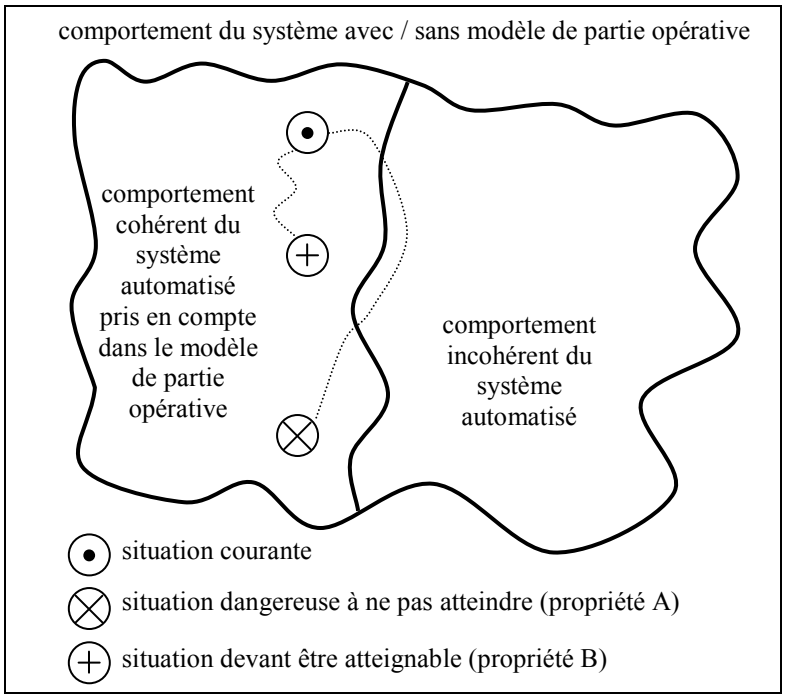


Figure 1.8 : vérification de propriétés sur le comportement d'un système automatisé

c) Synthèse

La construction a priori établit le modèle de comportement de la commande en se basant sur un modèle de partie opérative et des contraintes de fonctionnement. La construction par synthèse de la commande est basée sur cette méthode. La théorie de supervision fournit les outils mathématiques nécessaires à la synthèse [Ramadge, 1989]. La théorie de supervision,

contrairement à ce que son nom pourrait laisser supposer n'a pas de lien avec la supervision industrielle destinée au pilotage extérieur et au suivi informatique d'un procédé de fabrication. Elle consiste à élaborer un superviseur interdisant à un modèle de comportement du procédé certaines de ces évolutions. Dans ce cadre, le comportement du procédé est modélisé indépendamment du comportement de la commande. Le modèle de comportement du procédé représente toutes les évolutions que le procédé peut satisfaire. Les spécifications indiquent les restrictions de comportement que le superviseur va devoir satisfaire. Il s'agit de définir quelles sont les évolutions qui ne satisfont pas les contraintes imposées par l'expert. En couplant ces deux modèles, le superviseur de commande est généré et décrit le fonctionnement le plus large du système automatisé sous contraintes. Le superviseur interdit l'occurrence d'évènements qui peuvent être inhibés par la commande. Le superviseur comporte donc toutes les évolutions possibles d'un modèle de procédé ne violant pas les contraintes de manière à être commandé. Le superviseur ne décrit pas une loi de commande, puisqu'il n'impose pas aux événements d'occurrer et accepte les évolutions indéterministes.

Le comportement d'un modèle de commande peut être généré à partir du modèle du superviseur pour obtenir un comportement de la commande respectant les contraintes spécifiées en rendant déterministes les évolutions des évènements commandables. Une telle extraction peut être faite par optimisation du parcours que le modèle de commande doit satisfaire ou aléatoirement.

L'étude bibliographique de l'évolution de la synthèse de la commande des systèmes à évènements discrets montre que depuis les fondements de la théorie de supervision, les apports successifs tendent à prendre en compte la détermination et l'implantation du comportement d'un modèle de commande. Différents travaux de synthèse sont référencés sur le chronogramme présenté en Figure 1.9.

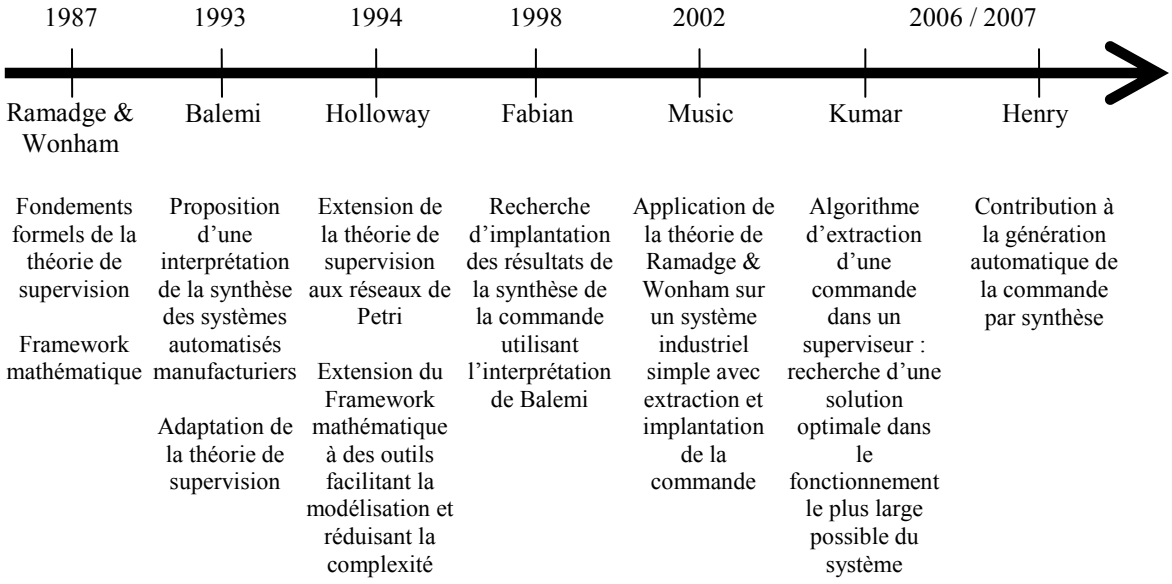


Figure 1.9 : historique des travaux de synthèse pour utilisation sur les systèmes réels

(i) Les fondements de la théorie de supervision

Depuis les fondements établis par Ramadge & Wonham [Ramadge, Wonham, 1987], de nombreux travaux ont été consacrés au développement de la théorie de supervision. Ces travaux formels considèrent que le processus est générateur d'évènements spontanés. Ces évènements sont classés en deux catégories. Pour les travaux relatifs à la théorie de supervision de Ramadge et Wonham, les évènements seront appelés contrôlables et non contrôlables. Comme illustré sur la Figure 1.10, tous les évènements sont générés par le modèle de processus. Les évènements contrôlables peuvent être inhibés par un superviseur tandis que les évènements non contrôlables sont toujours autorisés. Dans ce contexte, le superviseur n'est pas en mesure d'imposer l'occurrence d'un événement non contrôlable mais uniquement d'interdire l'occurrence de cet événement. [Ramadge, Wonham, 1987] fournit un cadre mathématique formel basé sur un outil appelé « théorie des langages » pour déterminer le comportement du superviseur. L'objectif est de sélectionner les autorisations d'évènements contrôlables permettant de respecter des contraintes liées au fonctionnement du système automatisé. Ces contraintes décrivent les situations ne devant pas se produire au cours du fonctionnement du système car elles correspondent à des situations dangereuses ou non souhaitées dans le cadre d'un fonctionnement nominal.

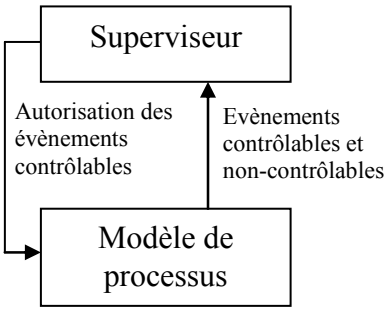


Figure 1.10 : synthèse de la commande selon Ramadge et Wonham

Les optimisations successives ont permis d'améliorer le concept et résoudre certains problèmes liés à l'explosion combinatoire ou bien encore certaines difficultés de modélisation. Les travaux utilisant les réseaux de Petri tels que [Holloway, Krogh, 1994], [Toguyeni, 2007] sont nombreux car ils permettent d'associer la synthèse avec un outil de modélisation simple d'utilisation et limitant l'explosion combinatoire. En revanche, les travaux liés aux applications pratiques de la synthèse de la commande des systèmes manufacturiers sont beaucoup moins nombreux. Les travaux de synthèse de Ramadge et Wonham sont souvent basés sur des modèles hauts niveaux utilisant le principe de « requêtes/réponses ». La partie opérative est vue comme un générateur d'événements dont certains peuvent être autorisés ou non par la commande. Cette vision peut être très utile pour les applications liées à l'ordonnancement de tâches mais ne peut pas s'appliquer sur des systèmes dans lesquels les valeurs des signaux dictent en permanence le comportement de la partie opérative. Le superviseur ne pouvant pas imposer des évolutions mais seulement en autoriser ou en interdire certaines, des problèmes de déterminisme se posent lorsque plusieurs

évolutions associées à des évènements contrôlables ou non sont simultanément possibles. Une nouvelle interprétation a donc été proposée pour être plus proche des problématiques rencontrées dans l'obtention d'un modèle de commande par la théorie de supervision dans le cadre des systèmes automatisés manufacturiers utilisant les API.

(ii) L'interprétation pour les systèmes automatisés de production

Dans [Balemi, 1992], [Balemi, 1993], une interprétation de la théorie de supervision est proposée. Celle-ci correspond mieux au problème d'élaboration de la commande des systèmes automatisés de production. Le principe consiste à étudier les valeurs des signaux échangés entre la partie commande et la partie opérative. La partie opérative n'est plus génératrice d'évènements mais subit les changements de signaux de la partie commande. Les termes « commandable » et « contrôlable » sont distingués en fonction de l'interprétation utilisée : le terme « contrôlable » est utilisé dans la synthèse de Ramadge et Wonham tandis que le terme « commandable » est utilisé dans l'interprétation de Balemi. Dans le cas de l'interprétation de Balemi et des changements de signaux, on utilisera les termes d'évènements commandables et non commandables. Le modèle de commande reçoit de la partie opérative des événements non commandables issus des capteurs et envoie des évènements commandables que la partie opérative doit exécuter (Figure 1.11).

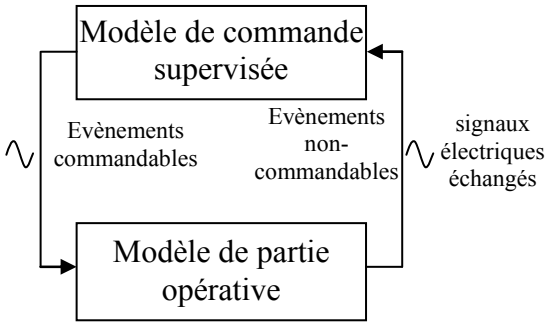


Figure 1.11 : interprétation de Balemi [Balemi, 1993]

Cette interprétation donne un autre sens aux modèles : le modèle de commande impose des évènements commandables au modèle de partie opérative. Les évènements non commandables correspondent aux réponses de la partie opérative à ces sollicitations. Le modèle de partie opérative n'est plus uniquement perçu comme un générateur d'événements spontanés pouvant être inhibés par le superviseur.

[Balemi, 1993] prend en considération les interactions entre le modèle de commande et le modèle de partie opérative. Ce concept aboutit de nouveau à un modèle de commande supervisée représentant l'ensemble des modèles de commande respectant les spécifications de fonctionnement de la partie opérative. Cependant, le modèle de partie opérative représente les évolutions des évènements non commandables conséquences des évènements commandables ayant pu avoir lieu. [Fabian, 1998] a utilisé cette interprétation pour la synthèse en vue d'implanter le résultat dans un API. Les problèmes rencontrés sont liés à la sémantique

utilisée par l'outil de modélisation lors de la synthèse de la commande. Des hypothèses sont proposées et formulées pour que le modèle de commande soit en mesure d'être implanté. Cependant, la construction du modèle de partie opérative reste intuitive et des erreurs de modélisation peuvent apparaître remettant en cause le résultat obtenu. Dans le cas d'un système complexe, les étapes qui précèdent l'application de ces travaux sont très difficiles à réaliser : la modélisation de la partie opérative à un niveau aussi proche de l'applicatif et la formalisation des contraintes deviennent particulièrement difficiles à obtenir et peuvent prendre des points de vue différents suivant le niveau de sûreté exigé et les modes de fonctionnement souhaités. L'écriture exhaustive des spécifications n'est pas simple à effectuer. De plus, l'implantation proposée dans ces travaux est le modèle de la commande supervisée sans que le modèle de comportement de la commande n'en soit extrait. Ces travaux sont néanmoins beaucoup plus proches de la problématique liée à l'utilisation d'API. Enfin, ces travaux ne font pas allusion à l'intégration d'un flux de production dans le modèle de partie opérative et ne représente pas l'écoulement du temps au cours du fonctionnement du système.

L'utilisation d'API comme applicatif nécessite néanmoins de s'intéresser plus particulièrement au problème d'implantation. Un des problèmes auquel la synthèse de la commande fait face est le type de résultat obtenu : en effet, la synthèse ne fournit pas le modèle de comportement de la commande à proprement parler mais un modèle de commande supervisée décrivant le comportement le plus large possible du modèle de commande sous contraintes. Le modèle de la commande supervisée inclut donc toutes les évolutions possibles ne pouvant pas violer les contraintes. Le modèle de comportement de la commande doit donc être extrait de ce modèle de commande supervisée. L'API utilise un moteur d'exécution pour effectuer les opérations décrites dans le code implanté. Différents moteurs d'exécution peuvent être utilisés suivant que l'API effectue une recherche de stabilité dans le programme de commande avant d'exécuter les ordres. Le traitement séquentiel cyclique du programme automate nécessite donc de prendre en compte l'ensemble du comportement du programme de commande implanté dans l'API et le moniteur d'exécution pour traduire le comportement de la partie commande.

(iii) La problématique industrielle

Dans un contexte industriel, la modélisation du comportement de la partie opérative d'un système manufacturier est très complexe et demande beaucoup plus de connaissances que la modélisation de l'ordonnancement d'actions élémentaires sous forme de requêtes/réponses. L'écriture des contraintes de fonctionnement demande, elle aussi, de nombreuses connaissances sur le système en question et peut prendre différents points de vue. Il s'agit, en effet, de déterminer le comportement de la partie opérative en fonction de toutes les séquences d'évolutions pouvant être suggérées par la commande. Le modèle de commande supervisée obtenu peut avoir plusieurs significations : les contraintes peuvent exprimer les comportements non souhaités pour des raisons de sécurité et le modèle de partie opérative représente le comportement non sécurisé du système. Le modèle de commande supervisée

obtenu est alors le modèle de partie opérative contraint par la sécurité et comporte toutes les évolutions non interdites.

Le modèle de partie opérative initial utilisé pour l'opération de synthèse peut tout aussi bien exclure toutes les situations interdites d'un point de vue sécurité et les contraintes peuvent être représentés par des spécifications décrivant le fonctionnement attendu du système en vue de produire. Le modèle de commande supervisée obtenu est, dans ce cas, l'ensemble des séquences aboutissant à l'élaboration du produit.

D'un point de vue plus général, le principe de l'opération de synthèse consiste à restreindre le comportement d'un système par des contraintes pour en extraire le comportement le plus large possible tout en respectant les contraintes. Pour appliquer ce concept aux systèmes manufacturiers, il est nécessaire de définir une sémantique cohérente avec les difficultés de modélisation et de formalisation rencontrées pour éviter les ambiguïtés.

Le modèle de partie opérative décrit les évolutions que la partie opérative est susceptible de suivre. Les contraintes de sécurité traduisent les comportements que la partie opérative ne doit pas pouvoir exécuter. Dans ce contexte, écrire les contraintes de sécurité correspond à décrire toutes les situations dont le comportement est dangereux ou incertain parce qu'il ne présente pas d'intérêt dans la finalité du système automatisé. Ces modèles formels expriment le comportement d'un point de vue déclaratif. Les contraintes liées aux conditions fonctionnelles traduisent, quant à elles, les évolutions que le système ne doit pas effectuer. Il s'agit de restreindre le comportement du système aux exigences exprimées par l'utilisateur pour son besoin telles que produire. Les modèles associés à ce comportement sont là encore déclaratifs. Il n'est pas toujours facile d'identifier si une contrainte est relative à la sécurité ou aux conditions fonctionnelles.

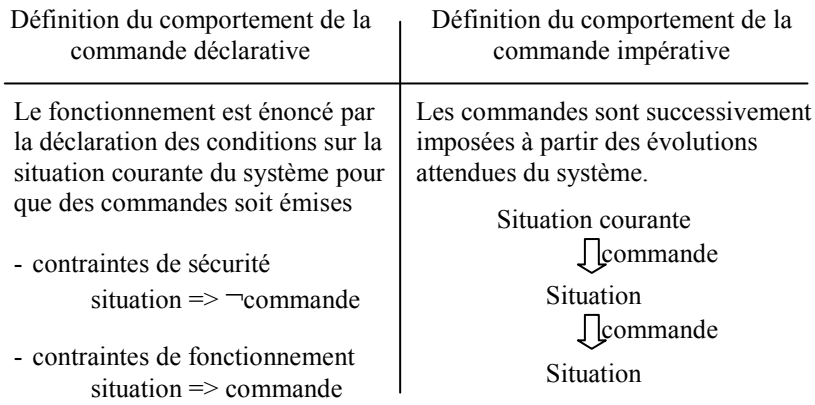


Figure 1.12 : définition des points de vue déclaratifs et impératifs

Dans les outils de programmation des systèmes automatisés, le comportement de la commande recherchée par le concepteur est bien souvent impératif (Figure 1.12) : dans ce contexte, il ne s'agit pas de rechercher quelle est la famille de modèles de commande susceptible de ne pas violer de contraintes ; on ne cherche pas un ensemble de solutions mais à imposer une solution particulière. Le modèle de commande supervisée obtenu peut aussi être utilisé pour la validation d'une commande proposée par le concepteur [Tajer, 2005]. Dans

ce cadre, il est indispensable que la sémantique utilisée par le modèle de commande supervisée et le modèle de commande proposé par le concepteur soient similaires. Il est également indispensable que le moniteur d'exécution utilisé dans les deux approches soit identique. La démarche consiste alors à tester si une proposition de commande est cohérente vis-à-vis d'un superviseur représentant le fonctionnement.

Une solution alternative à la validation d'une commande proposée par le concepteur est d'extraire le modèle de commande du modèle de commande supervisé. Des travaux récents se sont aussi intéressés à l'extraction de ce modèle.

(iv) L'extraction d'une commande dans un superviseur

Les travaux récents de Kumar sont consacrés à l'extraction de la commande dans le superviseur par optimisation. Kumar utilise le terme « directed control » pour désigner l'extraction d'un chemin optimisé dans le superviseur représentant le comportement le plus large possible admissible sous contraintes par le système automatisé. Dans [Huang, 2005], le modèle de comportement est extrait avec un algorithme destiné à restreindre le nombre d'évolutions successives. Ces travaux sont basés sur la théorie de Ramadge et Wonham sans utiliser l'interprétation de Balemi. A chaque occurrence d'événement, une transition est franchie. L'optimisation est donc réalisée en réduisant le nombre de transitions associées à des évènements contrôlables franchies pour atteindre un ou plusieurs états finaux indiqués comme étant marqués depuis un état initial. Le modèle de superviseur ayant subi l'optimisation est appelé directeur de commande. Les restrictions de comportement effectuées concernent uniquement les événements contrôlables. Les restrictions peuvent amener le modèle à ne jamais atteindre d'états marqués. Si une telle situation peut se produire, le directeur de commande est dit bloquant. Les auteurs démontrent une propriété sur la condition d'existence d'un directeur de commande non bloquant. Ensuite, un algorithme détermine comment calculer le directeur de commande. Il s'agit d'une boucle itérative utilisant la théorie des régions. A partir de l'ensemble des états marqués, une première région est formée incluant tous les états permettant d'atteindre un des états marqués sur occurrence d'évènements non contrôlables. La région est ensuite complétée par extension à l'ensemble des états permettant d'atteindre la région par occurrence d'un seul événement contrôlable. Les évènements contrôlables inutiles, c'est-à-dire qui n'ont pas d'intérêt pour agrandir la région, sont supprimés. En itérant sur cette extension, on finit par atteindre l'état initial en ayant conservé le minimum d'évènements contrôlables. En cas de mise en concurrence entre deux occurrences d'évènements contrôlables, le choix de l'événement à conserver est arbitraire. Le modèle obtenu est nettoyé de tous les états inatteignables et des transitions associées à des évènements ne pouvant plus occurrer. Dans [Huang, 2006], l'outil de modélisation est renforcé par l'apparition de valeurs associées au franchissement d'une transition. Un critère de minimisation de la valeur de franchissement des transitions permet de résoudre, en partie, les choix arbitraires entre les occurrences d'évènements contrôlables. Cette démarche intéressante reste néanmoins très théorique et ne permet pas jusqu'à présent d'applications dans la génération automatique de programme API sur les systèmes manufacturiers.

(v) Vers une applicabilité dans les systèmes automatisés de production

Dans [Music, 2002], l'approche se veut beaucoup plus proche des applications réelles. Le but visé est l'implantation d'une commande dans un API. Ces travaux utilisent les cadres formels préalablement établis pour proposer une démarche d'obtention de la commande par une approche modulaire de la théorie de supervision. Une méthode d'implantation est proposée pour convertir l'automate à états finis résultant de la démarche dans un formalisme exécutable par un API. Des hypothèses sont émises sur le fonctionnement global du système pour que la théorie de supervision puisse toujours être utilisée. En effet, des problèmes peuvent se poser quant à l'hypothèse forte de non-simultanéité des évènements. Le modèle obtenu permet l'implantation et la validation in situ de la démarche. Le problème d'implantation mis en avant dans ces travaux fait apparaître une problématique liée à la mise en pratique de la théorie de supervision dans le cadre de systèmes industriels.

d) Problèmes d'implantation

L'étude du comportement de la commande des systèmes automatisés nécessite de s'intéresser à l'implantation du modèle dans l'API. Les API exécutent du code de programmation suivant un moniteur d'exécution [Rohee, 2005] définissant les règles d'évolution temporelle de la commande. Les ordres et les comptes-rendus sont des signaux électriques. Ils représentent les échanges de valeurs discrètes entre la partie opérative et la partie commande. La réalisation du modèle de comportement de la commande est dépendante du programme de commande implanté et du moniteur d'exécution du programme. Un même programme de commande peut fournir un comportement différent en fonction du moniteur d'exécution. Le concepteur de la commande définit une solution de programme de commande en fonction de son interprétation du fonctionnement du moniteur d'exécution du programme mais aussi de l'interprétation du comportement de la partie opérative. Les travaux liés à l'implantation de la commande s'intéressent donc à la réalisation du programme de commande et à sa justification en tenant compte de la cible d'implantation et de son comportement.

Dans la définition de la solution du concepteur, le programme de commande implanté dans l'API est celui qu'il a défini. En revanche, le fonctionnement du moniteur d'exécution de l'API et le comportement de la partie opérative sont sujets à interprétation de la part du concepteur.

Très souvent, l'outil de modélisation utilisé par le concepteur influence son interprétation du moniteur d'exécution. L'utilisation de l'automate à états finis comme modèle de conception a des conséquences sur l'interprétation du moniteur d'exécution par le concepteur. En effet, des hypothèses de non-simultanéité d'évolutions doivent être formulées. La modélisation de l'écoulement du temps est notamment très dépendante de cette interprétation. Dans [Roussel, 2005], la modélisation sous la forme d'automates à états est agrémentée d'une forme de modélisation de l'API cible. Cette solution permet de faire apparaître une notion d'interprétation du moniteur d'exécution dans la synthèse de la commande. Il s'agit d'une

solution d'adaptation de l'outil de modélisation pour que l'interprétation du moniteur d'exécution soit plus proche de son fonctionnement réel. L'écoulement du temps lors du fonctionnement du système n'est pas toujours pris en compte dans le modèle de commande et de partie opérative. Dans le modèle de commande spécifié par des automates à états, le système de commande est supposé infiniment réactif.

Dans le fonctionnement réel d'un automate programmable industriel, le système de commande n'est pas infiniment réactif : l'exécution monotâche se déroule suivant un cycle dans lequel trois étapes sont exécutées (Figure 1.13). La première étape consiste en l'acquisition de toutes les entrées. Il s'agit d'une recopie instantanée de tous les états des entrées dans des registres internes de la mémoire de l'automate programmable industriel. Dans la deuxième étape, le programme implanté dans l'automate est traité. Le code programme que le concepteur a défini modifie les affectations des registres de la mémoire. Le temps pris pour cette étape est très dépendant du nombre d'instructions à exécuter et des performances de calcul du microprocesseur de l'automate programmable industriel. La dernière étape consiste à affecter les sorties de l'automate programmable industriel. Une recopie des registres mémoire est effectuée sur les sorties physiques de l'API.

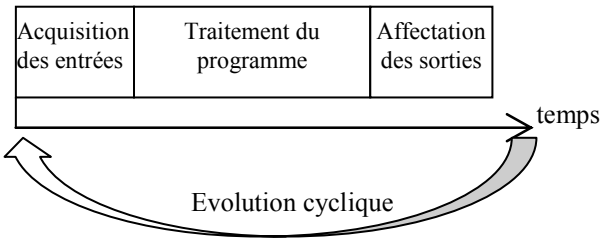


Figure 1.13 : évolutions cycliques de l'automate programmable industriel

L'évolution cyclique dans le temps du traitement du programme automate nécessite de prendre en compte un modèle du moniteur d'exécution du modèle dans les opérations de synthèse. Les travaux basés sur les outils de modélisation tels que les automates à états finis et toutes ses variantes ne tiennent, en général, pas compte du comportement de l'automate programmable industriel qui interprète le modèle de commande. Cette interprétation et la prise en compte du cycle automate et de la gestion du temps sont néanmoins primordiales pour appliquer ces travaux sur des cas industriels. En effet, le cycle automate fait apparaître des phénomènes de discrétisation dans le temps des signaux et de synchronisation des signaux les uns avec les autres. Les modèles basés sur les automates à états font l'hypothèse qu'il y a asynchronisme entre les évolutions et posent donc un problème de sémantique pour l'application de ces résultats sur les cas industriels. Le modèle de partie opérative n'est donc pas, à lui seul, l'élément suffisant pour appliquer les travaux de commande mais il faut également s'intéresser à l'environnement dans lequel il se place. L'implantation des travaux de synthèse de la commande fait l'objet de recherche mais les points de vue diffèrent selon les auteurs qui prennent en considération différents niveaux d'abstraction.

e) Méthodologie de modélisation initiée par Kumar

Suivant l'utilisation du modèle de partie opérative, la nature de l'outil de modélisation est différente. L'outil le plus communément utilisé dans les travaux de synthèse et de validation de la commande est l'automate à états finis. Les travaux successifs ont agrémenté cet outil de modélisation rudimentaire de fonctionnalités supplémentaires telles que la prise en compte du temps, l'ajout de fonction de pondération, le synchronisme ou bien encore les extensions vers les réseaux de Petri.

Comme nous l'avons déjà indiqué, l'obtention du modèle de la partie opérative peut s'avérer complexe pour le concepteur. Dans le cas des systèmes manufacturiers, une première approche consiste à étudier tous les états qui peuvent être atteints dans un modèle unique. Cette démarche aboutit à une explosion combinatoire car le nombre d'états augmente de manière exponentielle avec le nombre d'entrées/sorties du système modélisé. Il devient vite impossible de décrire sans méthodologie ce modèle unique.

* Dans [Chandra, 2001a], les auteurs proposent une méthode pour obtenir le modèle de partie opérative sous la forme d'un automate à états finis. La partie opérative est vue comme un bloc comportant des entrées contrôlables (commandes des actionneurs) ou non contrôlables (perturbation) ainsi que des sorties non contrôlables (capteurs). Les entrées et sorties non contrôlables peuvent être observables ou non observables vis-à-vis de la commande (Figure 1.14). La dénomination « entrées/sorties » est inversée par rapport à l'interprétation de Balemi car on se place du point de vue de la partie opérative et non pas du point de vue de la partie commande. Les seules informations auxquelles la partie commande peut avoir connaissance sont les entrées contrôlables et les sorties observables du modèle de partie opérative.

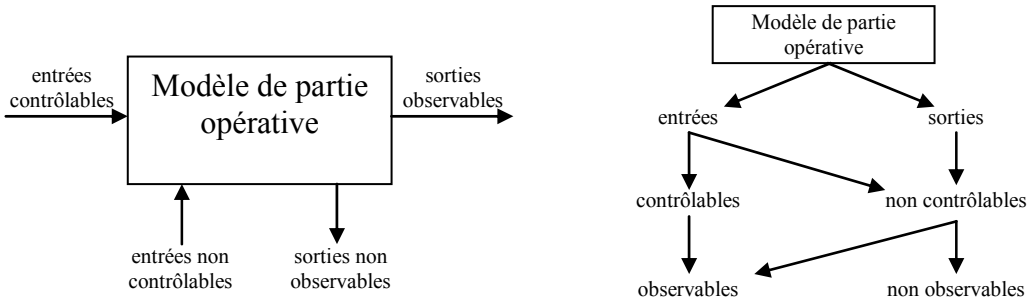


Figure 1.14 : représentation du modèle de partie opérative

La construction du modèle de partie opérative par cette méthode se veut modulaire : à partir d'un découpage du modèle en sous-modèles simplifiés et en assemblant ces différents sous-modèles, on peut obtenir le modèle complet de la partie opérative. La méthode proposée dans ces travaux est de décrire le fonctionnement du système par des assertions.

1) description de l'état initial du modèle de partie opérative
2) règles d'occurrence des évènements du modèle de partie opérative
3) relations de précédence

Tout d'abord, l'état initial des entrées/sorties du système sont définies telles qu'elles se trouvent lorsque le système est mis en fonctionnement. Cette description permet de faire coïncider le modèle de partie opérative avec la situation dans laquelle se trouve la partie opérative à la mise sous tension.

Ensuite, les règles d'occurrence des évènements non-contrôlables décrivent l'évolution de chaque sortie du modèle de partie opérative à partir des entrées de ce modèle. Pour que le système soit entièrement décrit, il faut que tous les évènements possibles des sorties non contrôlables soient décrits. Pour chaque sortie booléenne, il y a deux évènements possibles : le front montant et le front descendant. Les règles d'occurrence sont les relations déterminant l'occurrence d'un évènement contrôlable en fonction des entrées.

Pour un système comportant *m* sorties non contrôlables et *p* entrées contrôlables, il y a *2m* règles d'occurrence qui sont des fonctions des *p* entrées contrôlables et paramétrées par les entrées non contrôlables. Une sortie non contrôlable peut dépendre de plusieurs évènements d'entrées contrôlables. Des poids peuvent être affectés sur les événements occurrents pour donner des priorités en cas de contradiction (une entrée peut générer un événement mais une autre entrée peut empêcher cet événement si elles agissent sur le même élément de manière contradictoire : par exemple, l'ouverture d'une vanne d'évacuation peut empêcher un récipient de se remplir bien que le remplissage soit en cours ; dans ce cas, un poids est affecté au remplissage et à l'évacuation pour décider si l'occurrence de l'événement « récipient rempli » peut arriver). Les poids affectés ne peuvent être que positifs sur chaque évènement. C'est la comparaison entre les poids de deux évènements contraires qui permet de conclure sur l'évènement qui doit survenir. Des opérateurs ont été mis en place pour calculer les poids lorsqu'il y a des combinaisons d'entrées sur un même élément (par exemple, la mise en place de vannes en série ou en parallèle pour le remplissage d'une cuve). Il s'agit des opérateurs AND et OR.

Enfin, les relations de précédence entre les évènements non contrôlables finissent de décrire le modèle de partie opérative. Il s'agit de décrire le déroulement normal de la séquence d'événements non contrôlables (par exemple, si le récipient est initialement vide, il doit être partiellement rempli avant d'atteindre la fin de remplissage). Parmi les relations de précédence, il faut obligatoirement mettre des relations pour imposer qu'un front montant ne peut avoir lieu que si un front descendant a eu lieu auparavant. Il s'agit ici de décrire les séquences d'évènements possibles intrinsèques à une sortie non contrôlable.

Le modèle est construit à partir d'une démarche décrivant les étapes successives permettant d'établir le modèle de partie opérative.

Les étapes pour la construction du modèle à partir des informations fournies ci-dessus sont :

1) Créer un état initial à partir des conditions initiales

2) Construire un graphe qui, partant de l'état initial, contient tous les états accessibles par franchissement des transitions associées aux évènements contrôlables. On obtient un graphe comportant *2p* états pour *p* entrées contrôlables

3) Mise en place des transitions de bouclage sur les états pour les évènements non contrôlables

4) Regroupement des transitions de bouclages sur chaque état avec calcul des poids

5) Représentation des relations de précédence sous la forme d'automates à états

6) Faire le produit synchrone de l'automate à états obtenus à l'étape 4 avec les automates à états obtenus à l'étape 5

* Dans les publications [Chandra, 2001a] et [Chandra, 2001b], des exemples sur le remplissage d'une cuve sont présentés. Il s'agit d'exemples simples constructibles manuellement. Ces exemples permettent également de voir comment sont écrites les différentes assertions et l'utilisation des opérateurs pour calculer les poids. Un autre exemple est fourni dans ces articles pour un convoyeur. Cela montre que la méthode peut s'appliquer à des systèmes automatisés manufacturiers.

La méthode décrite permet d'obtenir un modèle de partie opérative de manière descriptive : à partir d'une description du fonctionnement du système, on obtient le modèle. Il est cependant nécessaire d'avoir une expertise du fonctionnement du système pour obtenir le modèle. De plus, si des erreurs ou oublis sont présents dans la description du fonctionnement du système, rien ne permet de déterminer ces erreurs et une solution sera trouvée par la méthode. Enfin, la description du fonctionnement du système par des assertions nécessite de prendre en considération l'ensemble du comportement du système pouvant avoir lieu.

* Dans [Chandra, 2002], l'auteur a modifié quelque peu la méthode : les relations de précédence sont remplacées par les règles d'occurrence. Chaque règle d'occurrence comporte trois fonctions : une fonction comportant un poids qui autorise l'occurrence de l'évènement, une fonction comportant un poids pour interdire l'occurrence de l'évènement et une fonction ne comportant pas de poids mais qui conditionne l'occurrence de l'évènement.

Chaque règle est donc écrite de la manière suivante :

$$\prod_{poids} (poids\_autorisant > poids\_interdisant)\, ET\, (condition\_sans\_poids) \rightarrow évènement$$

On obtient $2m+2p$ règles avec m le nombre de sorties non contrôlables et p le nombre d'entrées contrôlables du modèle de partie opérative. Ces règles correspondent à tous les évènements qui peuvent survenir sur les entrées contrôlables ou sur les sorties non contrôlables.

* [Huang, 2003] reprend la suite des travaux de [Chandra, 2002]. L'objectif est de faire apparaître des défaillances dans le fonctionnement du système et de les prendre en considération dans le modèle de partie opérative. L'auteur ajoute, dans les règles d'occurrence, des conditions pour les fautes. Les fautes prises en compte peuvent être le blocage d'une entrée contrôlable ou d'une sortie non contrôlable ou une faute du système ou d'un de ses composants. Pour cela, on ajoute des variables booléennes qui indiquent l'état de blocage d'un signal. Dans toutes les règles d'occurrence, on décrit de la même façon chaque

événement non contrôlable qui peut subvenir dans le déroulement sans faute ainsi que l'occurrence de l'évènement suite à une faute.

On écrit des règles supplémentaires pour décrire dans quels cas la faute peut survenir :

- ✓ signal au niveau bas ET évènement blocage du signal au niveau bas => faute

Pour les fautes dues au système ou à ses composants, il suffit que l'évènement causant la faute survienne pour que la faute soit présente.

- ✓ évènement causant une faute => faute

Toutes les règles d'occurrence sont alors modifiées de la manière suivante :

- ✓ Condition sans faute OU Condition avec faute => événement non contrôlable

* Dans [Huang, 2002], les fautes temporelles peuvent également être traitées. Pour cela, on met en place des variables temporelles discrètes qui sont initialisées à 0 et qui sont croissantes. On met en place des règles d'occurrence pour réinitialiser ces variables à 0. Dans les règles décrivant le fonctionnement du système, on ajoute les conditions temporelles en écrivant dans quel intervalle de temps peut survenir cet évènement.

- ✓ condition ET [0 < variable de temps < n] => évènement

Pour construire le modèle, il faut, dans un premier temps utiliser la méthode décrite précédemment pour obtenir le modèle sans l'aspect temporel puis ajouter sur les transitions les conditions de gardes temporelles ainsi que les réinitialisations des horloges.

L'utilisation de la modélisation de la partie opérative avec fautes permet d'affiner le modèle mais engendre un modèle de taille plus importante. Le modèle est, certes, plus complet mais l'explosion combinatoire va être atteinte plus rapidement. L'utilisation des fautes est assez difficile à mettre à place car elle nécessite que l'utilisateur sache quels éléments vont être mis en cause et oblige à écrire des conditions avec fautes pas toujours évidente à écrire. La recherche de fautes temporelles permet d'avoir un modèle beaucoup plus fin dans la description du système puisqu'on peut modéliser plus que l'ordre d'occurrence des évènements mais aussi les délais entre les occurrences d'évènements. Cependant, l'utilisation de cette modélisation se restreint à des systèmes de petite taille avec des discrétisations temporelles grossières (valeurs incrémentées par des horloges de temps limitées à un faible nombre de valeurs).

f) Utilisation de la méthodologie de modélisation initiée par Kumar pour les systèmes manufacturiers

La méthodologie de modélisation présentée peut être utilisée pour décrire le comportement des systèmes automatisés manufacturiers. Dans [Philippot, 2004], le but est de réaliser la synthèse de la commande des systèmes de production. Ces travaux permettent de présenter les

différents modèles pouvant être obtenus par la réduction de comportement de la partie opérative que propose la synthèse de la commande.

La méthodologie de modélisation utilisée par Kumar est reprise en utilisant l'interprétation de Balemi. Les contraintes utilisées dans la démarche de synthèse sont destinées à inhiber les actions pour assurer la sûreté (contraintes de sécurité) ainsi que pour vérifier le bon déroulement des actions dans l'ordre (contraintes de vivacité). Les contraintes ont pour but de restreindre les états atteints par le modèle de la partie opérative aux seuls états qui sont fonctionnels. En composant le produit synchrone des contraintes modélisées par des automates à états finis, on obtient le modèle des contraintes. On effectue la synthèse en composant le modèle de la parte opérative avec le modèle des contraintes. La méthodologie de modélisation de la partie opérative utilisée est celle de Chandra et Kumar car elle permet de modéliser de manière explicite la partie opérative indépendamment de la modélisation de la partie commande. Dans ce contexte, la partie opérative assimile tous les évènements commandables que la partie commande peut soumettre. En étendant cette méthode, il est possible de générer les modèles des contraintes. Les modèles de contraintes de sécurité ne comportent pas d'évènements non commandables, ils ne dépendent que des évènements commandables. Les relations de précédence sont donc les relations entre éléments commandables. Pour les contraintes de vivacité, les règles d'occurrences traduisent les conséquences des évènements non commandables sur les évènements commandables. On obtient alors le modèle des contraintes en faisant le produit synchrone des différents automates de contraintes de vivacité et de sécurité. Le modèle de comportement restreint obtenu par cette démarche comporte toutes les situations ne violant pas les contraintes de sécurité et de vivacité énoncées. Il ne s'agit pas d'un modèle de commande ou d'un modèle de partie opérative mais d'un modèle de comportement d'un système restreint à son fonctionnement nominal.

4) Conclusion

Dans ce chapitre, nous avons mis en évidence la nécessité d'avoir un modèle de partie opérative dans les différentes applications attachées à l'automatisation des systèmes de production. L'outil de modélisation généralement utilisé est l'automate à états finis avec différentes variantes pour s'adapter aux besoins propres. Bien que ce modèle de partie opérative soit à la base de tous les travaux, il existe peu de travaux de recherche pour faciliter sa réalisation. De plus, des problèmes liés à la taille du modèle et à l'applicabilité de ces propositions ne permettent pas d'exploiter ces travaux dans le monde industriel. Enfin, les modèles de partie opérative sont construits à partir d'une étude fonctionnelle du comportement du système modélisé. La conception du modèle de partie opérative à partir d'une description fonctionnelle ne permet pas d'exploiter facilement le modèle de partie opérative pour implanter le résultat dans l'API.

Le chapitre 2 va présenter les outils de modélisation rencontrés dans les travaux de recherche et comment la modélisation de la partie opérative est appréhendée par les auteurs.

Le chapitre 3 propose ensuite une approche différente de la réalisation du modèle de partie opérative. La construction du modèle de partie opérative va s'effectuer par une approche structurelle basée sur les aspects liés à la mécanisation dans la phase de conception de la partie opérative réelle. L'objectif est de prendre en compte le domaine d'utilisation et les technologies de mécanisation mise en œuvre lors de la construction de la partie opérative. Des exemples d'application montreront comment exploiter la modélisation de la partie opérative et les problèmes consécutifs.

Le chapitre 4 utilisera enfin proposera une extension de la démarche de modélisation de la partie opérative pour considérer l'aspect continu du comportement physique de la partie opérative et la perception discrète de la partie commande. Des applications en simulation de l'utilisation de ces modèles montrera l'intérêt de la vision hybride de la partie opérative.

Chapitre 2
Les outils de modélisation de la partie opérative

Les différents travaux de recherche concernant les systèmes automatisés manufacturiers nécessitent une représentation formelle du comportement de la partie opérative. La formalisation de cette modélisation passe nécessairement par l'utilisation d'outils de modélisation qui représentent, en général, le comportement du système automatisé par une succession de changements d'état. Les modèles obtenus sont en général focalisés sur le besoin d'information et sur le résultat à obtenir pour l'application de recherche. En effet, l'exploitation du modèle de partie opérative a de fortes incidences sur sa conception : construction de la commande, validation/vérification, diagnostic, etc. Cependant, le modèle de partie opérative peut aussi avoir pour objectif de représenter le comportement fonctionnel ou bien le comportement physique des différents constituants de la partie opérative.

Représenter le comportement d'un système automatisé de production par un modèle formel nécessite de définir une classification des modèles, le cadre de l'utilisation d'un modèle, les capacités de l'outil de modélisation mis en œuvre et ses limites. De plus, l'étude de la partie opérative est nécessairement associée à l'étude de la commande en interaction avec celle-ci. La modélisation de la partie opérative ne peut donc pas être totalement dissociée du modèle de la partie commande avec lequel il interagit. Il est nécessaire de définir quelles sont les limites d'interactions entre ces deux modèles pour les isoler. Enfin, l'utilisation du modèle peut se dérouler hors ligne ou en ligne. Dans le second aspect, la modélisation pose le problème des contraintes de temps réel et nécessite d'être cohérent vis-à-vis des matériels physiques constituant la partie opérative.

L'exploitation du modèle de partie opérative a de fortes incidences sur la conception de celui-ci. Le modèle de partie opérative peut avoir pour objectif de représenter le comportement fonctionnel d'un système ou bien le comportement physique des différents constituants de la partie opérative.

L'objectif de ce chapitre est d'étudier comment le comportement d'un système automatisé de production est généralement modélisé dans les travaux de recherche en lien avec l'obtention d'un modèle de commande. En conséquence, nous présentons, tout d'abord, une classification des modèles de partie opérative suivant la description choisie du système, puis l'influence de l'utilisation en ligne ou hors ligne des modèles et enfin les interactions entre le modèle de commande et le modèle de partie opérative.

Nous décrivons ensuite les outils de modélisation couramment utilisés et les modèles générés à l'aide de ces outils.

A partir des remarques formulées sur ces modèles, nous proposons une propriété formelle mettant à l'épreuve un modèle de partie opérative dans un contexte donné. L'objectif de cette propriété est de soumettre le modèle de partie opérative à une spécification donnée pour tester s'il ne manque pas d'évolutions de commande dans la formalisation du modèle de partie opérative.

1) Contexte de la modélisation de la partie opérative : outils et méthodes

a) Les outils rencontrés dans les applications industrielles

Dans le domaine de l'industrie, l'automatisme est utilisé pour piloter les moyens de production. L'objectif des équipements d'automatisme est de produire tout en assurant l'intégrité du moyen de production et la sécurité des personnes. Les plateformes d'implémentation sont souvent composées d'automates programmables industriels (API) notamment pour leur facilité d'intégration et pour leur robustesse de fonctionnement. L'utilisation de ces API nécessite des méthodes de programmation basées sur la standardisation des langages de programmation (norme IEC 61131-3). Les outils de spécification du comportement de la commande utilisent des langages tels que le Grafcet (norme IEC 60848). Bien que graphiquement ressemblant, le SFC de la norme IEC 61131-3 se place dans le cadre de l'implémentation et décrit la structure interne du code implanté dans l'API. Le Grafcet se place ainsi d'un point de vue externe au système d'exécution. L'interprétation de ce modèle n'est pas similaire au langage de programmation. Le Grafcet ne tient pas compte du moniteur d'exécution interne lié à l'utilisation d'un API. C'est un outil de description standard du fonctionnement attendu indépendant de la plateforme d'implémentation prévue contrairement au SFC. En conséquence, deux modèles rigoureusement semblables graphiquement ne sont pas interprétés de la même manière selon qu'il s'agisse d'une représentation en Grafcet ou d'une interprétation SFC.

D'autres langages de programmation issus de la norme IEC 61131-3 permettent de programmer l'API avec un codage de programmation différent. Le langage ST (Stuctural Text) est un langage de programmation textuel évolué proche des outils de programmation informatique classique. Le langage IL (Instruction List) est un langage bas niveau proche du code assembleur. Le LD (Ladder Diagram) permet de représenter graphiquement le comportement de la commande avec la représentation utilisée par les électriciens. Enfin, le langage FBD (Function Block Diagram) utilise les fonctions de programmation orientée objet pour générer des composants graphiques aux fonctionnalités paramétrables et interconnectables.

Pour les applications de supervision mises en place dans les systèmes industriels, des scripts sont généralement implémentés pour construire les informations nécessaires aux applications.

Le langage utilisé pour ces scripts n'est pas normalisé : il s'agit d'un langage littéral propre au fournisseur du logiciel de supervision dont les fonctionnalités apportées sont similaires.

Les applications de diagnostic n'utilisent pas d'outils spécifiques : elles sont implémentées dans le logiciel de supervision ou dans le programme de commande. Les outils utilisés sont donc ceux de la norme IEC 61131-3 et les scripts des logiciels de supervision.

La problématique qui se pose avec ces outils d'implémentation est qu'il n'existe pas de démarche pour définir, représenter et exploiter les modèles de partie opérative. La programmation associée à l'utilisation d'un modèle de partie opérative est donc empirique et très dépendante des habitudes du concepteur. La mise en œuvre d'un modèle de partie opérative peut totalement être masquée par la création de fonctionnalités basiques de reconstruction d'informations uniquement à partir des réflexions du concepteur.

b) Les outils de représentation utilisés dans les approches théoriques et fondamentales

La théorie des langages, les automates à états, les réseaux de Petri, l'algèbre Min-Max, les langages synchrones ou bien encore le Grafcet sont des outils couramment rencontrés dans la recherche fondamentale pour les applications d'élaboration de la commande sûre de fonctionnement, le diagnostic, l'identification…

La théorie des langages et les automates à états finis sont très souvent utilisés pour l'obtention d'un modèle de commande dans les approches théoriques. Néanmoins, ces outils ne permettent pas d'utiliser les modèles de commande élaborés directement dans les API car les langages utilisés ne font pas partie de la norme [IEC 61131-3].

Comme indiqué dans le premier chapitre, la formalisation du comportement du système automatisé sous la forme d'un modèle est rendue nécessaire pour l'application de ces travaux. Pour des raisons de cohérence, les auteurs sont bien souvent amenés à décrire le comportement du système automatisé de production à l'aide des mêmes outils. Cependant, ces outils restreignent la modélisation de la partie opérative en raison de leurs capacités d'abstraction.

La théorie de supervision initiée par Ramadge et Wonham utilise notamment les automates à états finis pour décrire le comportement de la partie opérative et le réduire à l'aide de contraintes de fonctionnement. Il existe des extensions de cette théorie utilisant les réseaux de Petri. Ce type d'applications exploite généralement les données du modèle de partie opérative hors-ligne : le système automatisé n'interagit pas directement en temps réel dans les algorithmes développés.

Le modèle de partie opérative a pour objectif de prendre en considération l'ensemble du comportement du système automatisé. Cette forme de simulation de comportement n'est toutefois pas déterministe car lorsque des actions ont lieu simultanément, il n'est pas toujours possible de savoir dans quel ordre les réactions de la partie opérative vont se produire.

Dans le cas d'applications fonctionnant en ligne, le modèle de partie opérative doit décrire le comportement futur du système automatisé à partir de la situation courante. Les algorithmes utilisés dans les applications en ligne et hors ligne ne répondent pas aux mêmes contraintes (Figure 2.1).

	Utilisation du modèle de partie opérative en ligne	**Utilisation du modèle de partie opérative hors ligne**
Construction du modèle	Recherche de la situation courante et construction des évolutions à venir	Simulation complète du comportement de la partie opérative
Problématique dans l'utilisation	Problématique liée à la réactivité entre le modèle et le système réel	Exploration et exploitation du modèle consommatrice de capacités informatiques
Incidences des lacunes	Risque importants lors de comportements imprévus dans le modèle	Non-perception des comportements imprévus
Niveau d'abstraction	Niveau d'abstraction proche de l'équipement	Niveau d'abstraction indépendant de l'équipement
Représentativité comportementale	Forte connexion avec le comportement réel	Faible représentativité du comportement concret du système

Figure 2.1 : comparaison des approches en ligne et hors ligne

L'utilisation d'un modèle de partie opérative dans des applications hors ligne permet de limiter les problèmes de réactivité des algorithmes au cours de l'exploitation. Le modèle de partie opérative utilisé hors ligne doit prendre en considération l'ensemble du comportement du système ce qui peut engendrer des problèmes d'explosion du volume de données à traiter ou des temps de calcul irréalistes étant données les capacités actuelles de calcul des équipements informatiques. A partir de ces constatations, il est indispensable d'étudier quelles sont les différentes possibilités de modélisation depuis l'utilisation des lois de la physique jusqu'à l'abstraction complète de la physique dans la représentation du comportement d'un système automatisé.

c) Les écarts entre le comportement réel d'un système automatisé de production et sa modélisation

Le comportement d'un système automatisé de production réel est régi par des relations physiques continues. Un ensemble de relations mathématiques constitué très souvent d'équations différentielles peut représenter partiellement le comportement physique du système sous la forme d'un ensemble de relations mathématiques. Les variables d'état issues de ce système d'équations différentielles sont alors utilisées pour les différentes applications. Cependant, l'exploitation d'un modèle représentant aussi finement le comportement réel n'est pas concevable dans des applications liées aux systèmes à évènements discrets. Néanmoins, les modèles sont à base d'équations mathématiques particulièrement intéressantes pour simuler virtuellement le comportement d'un système automatisé réel à travers une modélisation hybride discrète/continue. Il s'agit d'une forme de réalité virtuelle puisque le modèle de partie opérative peut être utilisé pour animer un modèle de commande implanté dans un API. Un tel modèle de partie opérative peut utiliser un moteur graphique créant alors pour l'utilisateur un espace virtuel dans lequel le système automatisé évolue de façon plus ou moins réaliste suivant la finesse du modèle. Le principe de base de cette modélisation est de représenter les évolutions qui auront lieu dans un temps infinitésiment petit à partir de la connaissance de la situation courante et des sollicitations de la commande. Le comportement pouvant avoir lieu dans le futur n'est a priori pas connu puisqu'il est construit au fur et à mesure de l'écoulement du temps et seul l'instant infinitésimal suivant est connu. La modélisation du comportement du système automatisé est déterministe puisque les variables d'état sont déterminées à partir l'ensemble de la connaissance du système à l'instant courant. Ce type de modèle ne peut donc pas permettre de construire un modèle de commande à partir de la connaissance a priori du comportement du système telle qu'elle est nécessaire pour la synthèse de la commande. En revanche, s'il s'agit de mettre à l'épreuve un modèle de commande sans prendre de risque avec un système réel, ce type d'approche est particulièrement intéressant pour la validation de la sûreté de fonctionnement.

La solution logicielle ITS PLC [ITS_PLC] [Riera, 2008], [Rohee, 2008a] de la société portugaise REALGAMES[2] est très représentative de l'exploitation de cette modélisation. Le principe est d'associer un moteur graphique de jeux vidéo en trois dimensions à une modélisation hybride du comportement du système. Les équations différentielles représentant les relations physiques continues permettent de prendre en compte les contacts entre les assemblages, l'accélération de la gravité, les mouvements des produits, … Cette simulation peut être pilotée par un API à l'aide d'une interface spécifique. Cet outil de simulation représente donc un environnement virtuel de production automatisé créé à partir d'un système réel grâce aux effets graphiques et sonores particulièrement bien adaptés pour la formation

[2] REALGAMES : http://www.realgames.pt

des personnes sur les outils de commande sans prendre de risque avec l'utilisation d'une partie opérative réelle.

La simulation de comportement d'un système automatisé n'est pas toujours synonyme de réalité virtuelle. Dans beaucoup d'applications de recherche, le comportement de la partie opérative est formalisé sous la forme d'un modèle utilisant uniquement les données nécessaires à l'application. La représentation d'un système automatisé peut donc prendre des formes diverses en fonction de l'application visée et de l'outil de modélisation nécessaire pour pouvoir s'interfacer avec l'application. La formalisation du modèle de partie opérative est donc effectuée à partir de l'observation du comportement réel du système sans utiliser la représentation par des équations différentielles. L'interprétation du comportement réel du système automatisé et la traduction sous la forme d'un modèle passent nécessairement par une discrétisation partielle ou totale dans l'espace et dans le temps générant une information partielle sur le comportement du système automatisé réel. Cette discrétisation réalise un filtrage des données issues du comportement global continu du système (Figure 2.2).

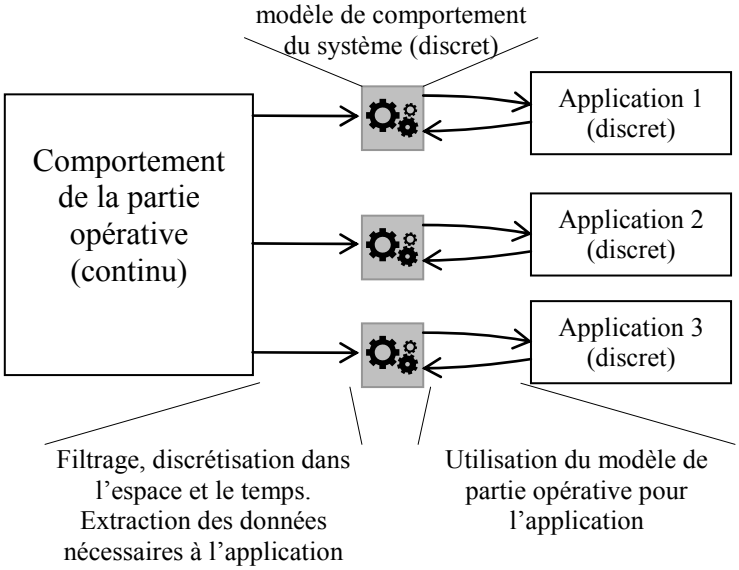


Figure 2.2 : interactions entre le comportement réel du système et la modélisation

Les modèles utilisant une telle représentation sont bien souvent utilisés hors ligne. Il ne s'agit que de donner une représentation des évolutions pouvant avoir lieu sur un système automatisé. Les évolutions sont énumérées successivement sans que le lien avec les évolutions mesurables sur le système réel ne soit défini. L'utilisation en ligne d'une telle modélisation nécessite de synchroniser le modèle avec les informations réellement mise à disposition par le système automatisé et de coupler l'ensemble du comportement du système automatisé avec le modèle de comportement.

Bien que tous issus du comportement d'un même système, les modèles utilisés dans les applications sont souvent différents. Le niveau de représentation des détails du système et les possibilités des outils de modélisation ne permettent généralement pas de prendre en compte

l'ensemble du comportement du système comme s'il avait été représenté par un modèle continu décrit sous la forme d'équations régissant les évolutions des grandeurs physiques. L'étude et l'interprétation du comportement réel du système permettent de proposer un modèle formel de comportement. Les informations disponibles dans ce modèle sont généralement décrites par des évènements représentant la détection d'une situation particulière du système de production telle que l'occurrence d'une panne ou d'une réparation, la prise en charge d'une pièce brute ou la fin de production d'un produit fini. Ces informations de haut niveau laisse la place à une interprétation importante sur la signification réelle ; le degré de précision de l'information n'est pas la priorité dans cette modélisation.

d) Illustration de la modélisation du vérin pour la simulation

Pour illustrer les différents points de vue sur la modélisation, un exemple de composant couramment utilisé dans les systèmes automatisés de production est utilisé. Il s'agit d'un vérin pneumatique avec un mouvement linéaire (Figure 2.3). Le système est commandé par un API recevant les informations des capteurs et pilotant les mouvements du vérin. Les éléments de pilotage et les capteurs ne sont pas particulièrement détaillés dans cet exemple. L'étude de ce système peut s'effectuer selon plusieurs points de vue.

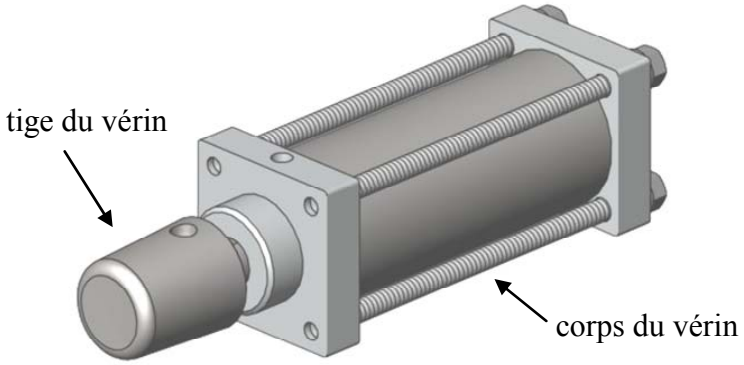


Figure 2.3 : exemple de vérin pneumatique

D'un point de vue continu, la simulation du comportement de ce vérin consiste à décrire les lois d'évolution au cours du temps. Des hypothèses peuvent être formulées pour simplifier la mise en équations de ce système. Les effets liés à l'inertie, aux frottements secs et visqueux, à la compressibilité de l'air, à la pression d'alimentation… peuvent être pris en compte dans des modèles fins. Un exemple de modèle de comportement continu basé sur une vitesse linéaire constante dépendant uniquement du mouvement demandé présenté Figure 2.4.

$$\begin{cases} \text{si } \textit{demande de sortir} \text{ alors } \dfrac{d(position)}{dt} = V \\ \text{si } \textit{demande de rentrer} \text{ alors } \dfrac{d(position)}{dt} = -V \end{cases}$$

Figure 2.4 : modèle continu du vérin représenté par un système d'équations différentielles

Pour élaborer le modèle, un repère est placé sur le corps du vérin pour déterminer la position de la tige par rapport au corps. La variable « *position* » correspond donc à la longueur d'extension du vérin. Une unité de mesure telle que le millimètre doit être choisie. La variation de la variable d'état « *position* » au cours du temps est décrite par le système d'équations différentielles de la Figure 2.4. Lorsque la demande « *sortir la tige du vérin* » est formulée, la vitesse de sortie de la tige est fixée à la valeur constante « V ». Lorsque la demande « *rentrer la tige du vérin* » est formulée, la vitesse de sortie de la tige est fixée à la valeur opposée à « V ». L'ensemble de ces deux relations simples permet une première approche de simulation des évolutions possibles du système à partir des demandes de mouvements formulées.

Cependant, les non-linéarités ne sont pas prises en compte dans cette modélisation. Le mouvement de la tige du vérin par rapport au corps est supposé non borné. Dans la réalité, le vérin n'a qu'une course limitée au cours de son mouvement et, arrivé en fin de course, le comportement réel du vérin ne peut pas assurer ce déplacement même si une demande de mouvement est formulée. Dans le cas du modèle proposé et si la variable d'état *position* n'est pas bornée, la variable *position* ne va pas adopter ce comportement. Le comportement du vérin à l'arrivée sur les butées de fin de course ne sera donc pas pris en compte dans cette modélisation.

Ce modèle nécessite également un éclaircissement de la situation initiale dans laquelle se trouve le vérin au démarrage du système. La situation initiale est associée à la position de la tige du vérin à t=0. Cette représentation permet néanmoins de représenter de façon primaire le comportement d'un vérin au cours du temps en fonction des mouvements qu'il effectue.

Le point de vue « Systèmes à Evénements Discrets » ne s'intéresse pas à ce type de modélisation car les informations contenues dans un tel modèle ne sont pas adaptées. De plus, cette représentation ne fait pas état des capteurs tout ou rien disposés sur le système réel. L'information contenue dans ce modèle n'est pas l'information réellement mise à disposition du système de commande. Il est donc nécessaire de prendre en compte le couplage entre le comportement physique du système automatisé et les informations réellement disponibles dans un API. La modélisation par des outils issus de l'étude des systèmes à évènements discrets se focalise plus particulièrement sur les successions de situations atteintes par le système au cours de son fonctionnement. La variable continue *position* et son étude de variation au cours du temps ne peuvent pas être conservées ainsi. Il s'agit plutôt de dénombrer les situations qui comportent un intérêt particulier dans le fonctionnement du système. Dans le

cas du vérin, on ne pourra pas, par exemple, distinguer uniquement si la tige du vérin est rentrée ou sortie. La représentation de ces modèles fait appel à des outils de modélisation très différents des équations différentielles. Cependant, avant de présenter des exemples de modèles de type discrets, les outils de modélisation nécessitent d'être plus détaillés.

2) Les outils de modélisation du comportement de la partie opérative dans le domaine des SED

Les outils de modélisation des systèmes à évènements discrets ont pour but de décrire le comportement d'un système automatisé par des évolutions liées à des changements de situations physiques de la partie opérative. Les modèles de comportement utilisés en SED ne construisent pas les données au fur et à mesure de l'écoulement du temps. Le modèle comporte ainsi toutes les données dès sa conception. L'atout majeur de ce type de représentation est de pouvoir représenter toutes les évolutions pouvant avoir lieu sur le système automatisé au cours du temps sans prendre en considération l'instant présent. L'exploitation hors ligne de ces modèles est donc possible. Le modèle de partie opérative n'est généralement pas déterministe puisqu'il doit prendre en considération toutes les évolutions pouvant avoir lieu au cours du fonctionnement du système automatisé et qu'il n'est pas toujours possible de connaître l'ordre dans lequel les évolutions vont avoir lieu lorsqu'aucune relation « cause - conséquence » ne peut être déterminée entre deux évolutions consécutives. Ce type de représentation est adapté pour déterminer le comportement de la commande à partir de la connaissance des évolutions susceptibles d'avoir lieu et des comportements non désirés. La synthèse de la commande exploite donc de tels modèles.

Pour décrire le comportement d'un système automatisé avec cette représentation, le principe général est d'énoncer les évolutions successives susceptibles de se produire. Dans le cas du vérin précédent, les relations de la Figure 2.5 permettent une représentation du comportement de ce vérin.

$$\begin{cases} \textit{Rentrer le vérin} \Rightarrow \textit{vérin rentré} \\ \textit{Sortir le vérin} \Rightarrow \textit{vérin sorti} \\ \textit{vérin rentré ET vérin sorti} = 0 \end{cases}$$

Figure 2.5 : modèle discret du vérin représenté par les successions d'évolutions pouvant avoir lieu

Dans cette représentation, la commande de rentrée ou de sortie du vérin conduit à la production d'une information sur la position du vérin : le vérin est rentré ou le vérin est sorti. Ce modèle ne tient pas compte de la situation initiale du système automatisé, le temps n'est pas pris en considération, le modèle ne fait uniquement qu'apparaître un séquencement des

évolutions les unes par rapport aux autres. Il est cependant possible d'extraire des informations qui n'étaient pas disponibles dans le modèle continu : par exemple, la première relation impose que la commande « *rentrer le vérin* » soit envoyée pour que le vérin rentre effectivement. Le modèle continu ne permet pas de déterminer directement quelles sont les commandes à envoyer pour que la variable d'état « *position* » puisse atteindre une valeur.

La représentation du comportement d'un système automatisé peut donc prendre des formes totalement différentes en fonction des informations contenues dans le modèle. Une représentation continue, prenant en considération la situation courante et les lois d'évolution physiques décrivant le comportement dynamique, est particulièrement appréciable lorsqu'il s'agit de simuler le comportement d'un système dans le but de représenter le système automatisé sous les aspects de réalité virtuelle. En revanche, lorsqu'il s'agit d'être exhaustif pour décrire toutes les évolutions que le système est susceptible de prendre, il est essentiel d'avoir la connaissance de tout le comportement du système quelques soient les circonstances. Dans ce cas de figure, l'abstraction est plus importante mais toutes les évolutions susceptibles d'avoir lieu sont connues sans aucune interprétation du modèle proposé. Les outils de modélisation et de représentation du modèle peuvent prendre des formes diverses en fonction de l'abstraction des informations et des contraintes liées à l'utilisation du modèle. Un tour d'horizon des outils de modélisation généralement utilisés pour représenter le comportement d'un système automatisé permet de connaître les données généralement prises en compte et les limitations inhérentes à l'utilisation de ces outils.

Les outils de modélisation comportant une abstraction du comportement peuvent utiliser deux visions différentes pour représenter le fonctionnement d'un tel système. Le fonctionnement peut être décrit par les évolutions successives ou par les situations successivement atteintes. Ces deux points de vue de description ne sont pas tout à fait similaires si on prend en compte la situation initiale dans lequel se trouve le système au démarrage. Dans le cas d'une description par les évolutions, la situation initiale peut ne pas être supposée connue. Elle peut être déterminée a posteriori après que les premières évolutions soient intervenues. Dans le cas d'une description par situations, la situation initiale peut supposée être connue. Avant qu'aucune évolution ne soit intervenue, des échanges d'informations sur la situation peuvent déjà renseigner sur la situation courante. Bien que partielle, cette prédétermination de la situation initiale permet de répondre partiellement à l'indéterminisme lié à la mise en fonctionnement nominal du système. Les outils de modélisation utilisent donc individuellement ou conjointement ces deux modes de description dans la représentation du système.

a) Description de la théorie du langage

La théorie des langages [Hopcrott, 1979] est un outil de modélisation basé sur le principe de description de l'ensemble des évolutions successives du comportement du système automatisé. Cette théorie utilise le vocabulaire et des analogies avec les méthodes utilisées dans la communication entre individus par l'écriture. Elle est basée sur l'assemblage de séquences de lettres issues d'un alphabet pour générer des mots. Cette théorie utilise donc un

vocabulaire particulier faisant appel au langage écrit. La théorie des langages est constituée de définitions des alphabets, des mots, ainsi que des opérateurs mathématiques. Tout d'abord, les lettres sont extraites d'un ensemble fini appelé alphabet.

L'alphabet décrit donc l'ensemble des lettres distinctes pouvant exister. Il n'existe pas de lettre vide.

$$\alpha, \beta, \gamma, \sigma, \tau ... \in \Sigma$$

Un mot est un assemblage de ces lettres. La séquence de lettres représentée par ce mot peut être finie ou infinie. Cette séquence de lettres peut également être vide. Il s'agit alors de la séquence vide notée ε. L'ensemble des mots pouvant exister à partir d'un alphabet Σ est noté Σ^* et inclut la séquence vide·

$$\Sigma := \{\sigma_1, ..., \sigma_n\},\ n \geq 1$$

$$\Sigma^* := \{\sigma_i \sigma ... \sigma_j\}, \sigma_i, \sigma_j, ... \in \varepsilon \cup \Sigma$$

b) Les opérateurs mathématiques utilisés dans la théorie des langages

Des opérateurs mathématiques permettent de travailler sur les mots. La concaténation de mots est une fonction permettant d'assembler deux mots distincts pour n'en former qu'un seul.

$$concatenation : \Sigma^* \times \Sigma^* \to \Sigma^*$$

$$concatenation(s,t) = st,\ s,t \in \Sigma^*$$

La notation * permet de décrire un ensemble de mot ayant la particularité d'avoir une succession de lettres ou d'ensembles de lettres.

$$st, ststst, ststst ... \in (st)^*,\ s,t \in \Sigma^*$$

La projection naturelle permet de sélectionner les lettres dans un mot. Toutes les lettres du mot initial qui ne font pas partie de l'alphabet de destination sont retirées.

$$projection : \Sigma^* \to T^*$$

$$projection(s\sigma) = projection(s)\, projection(\sigma),\ s \in \Sigma^*, \sigma \in \Sigma$$

$$projection(\sigma) = \begin{cases} \varepsilon \text{ si } \sigma \notin T \\ \sigma \text{ si } \sigma \in T \end{cases}$$

De nombreux autres opérateurs mathématiques ont été définis sur les langages mais ils ne sont pas détaillés ici.

c) Problématique de la théorie des langages dans la modélisation de comportement de systèmes automatisés

La théorie des langages peut être utilisée pour décrire les évolutions successives d'un système automatisé. Chaque évolution élémentaire du système peut être assimilée à une lettre. Pour décrire un ensemble d'évolutions, les lettres peuvent être assemblées sous forme de mots afin de pouvoir décrire toutes les évolutions possibles par un ensemble de mots. L'écriture exhaustive de ces ensembles de mots reste néanmoins peu évidente voire souvent impossible à réaliser à cause de la complexité du fonctionnement de la partie opérative. Il s'agit, de plus, uniquement de décrire les évolutions successives du système et il n'est pas possible d'avoir plus d'informations sur la situation courante dans laquelle se trouve un système après une succession d'évolutions car ce type de modélisation est uniquement basé sur le principe de description par les évolutions. La modélisation du comportement d'un système n'est donc pas facilement réalisable à l'aide de ce type d'outil car l'écriture d'ensemble de mots et la syntaxe utilisée ne sont pas simples à mettre en œuvre. La description du comportement d'un système automatisé uniquement par les évolutions successives n'est pas naturelle pour le concepteur et ne facilite pas l'obtention du modèle. Pour combler cette lacune, des outils permettent d'obtenir ce type de modèle à l'aide d'une description sous forme de situations. Les automates à états finis et les générateurs de langages permettent dans une certaine mesure d'obtenir des modèles sous forme de langage. Ces outils de modélisation graphiques sont beaucoup plus accessibles pour le concepteur. Néanmoins, ils ne permettent pas de se substituer à la théorie des langages car ils ne sont pas capables de représenter tous ces langages définis à partir de la définition originelle.

d) Définition des automates à états finis

Les automates à états finis ayant une fonction de transition partielle sont une solution pour définir un langage. Un automate à états finis est défini par un 5-uplet

$A := (Q, \Sigma, \delta, q_0, Q_m)$ avec

Q l'ensemble des états

Σ l'alphabet des évènements

$\delta : Q \times \Sigma \to Q$ la fonction de transition partielle de changement d'état sur occurrence d'un événement

q_0 l'état initial de l'automate à états

$Qm \subseteq Q$ l'ensemble des états marqués

L'état initial correspond au point de départ du générateur de langage. La fonction de transition définit l'association entre une évolution de l'état courant et l'occurrence d'un événement.

L'automate à états finis permet de générer des séquences d'évènements. Les évènements sont associés à des lettres de l'alphabet. Le langage généré est alors un ensemble de mots correspondant aux successions de lettres.

Le mot $s=\sigma_1\sigma_2\sigma_3$ représente la séquence d'évènements constituée de σ_1 puis σ_2 puis σ_3. Le mot vide ne comportant aucun évènement est nommé ε. Les états marqués sont des états qui doivent pouvoir être atteints par une succession d'occurrence d'évènements. L'ensemble des mots que l'alphabet Σ peut générer est appelé Σ^*. Le langage généré par un automate à états finis correspond à l'ensemble des mots que le modèle est susceptible de reconnaître. Le langage généré par un automate à états est inclus dans l'ensemble des mots pouvant être obtenu à partir de l'alphabet.

$$L(A) \subseteq \Sigma^*$$

L'automate à états finis est dit déterministe lorsqu'il n'y a pas de confusion possible dans la définition de la fonction de transition. Pour un état donné, il est déterministe lorsqu'il n'y a pas deux états différents qui peuvent être atteints pour la même occurrence d'un événement. L'occurrence simultanée d'évènements n'est pas supposée possible.

La fonction de transition $\delta : Q \times \Sigma \to Q$ est étendue vers $\delta : Q \times \Sigma^* \to Q$ en déclarant par récurrence $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$ avec $q \in Q, s \in \Sigma^*$ et $\sigma \in \Sigma$. Ainsi, l'état atteint après le franchissement du mot s puis σ est similaire à l'état atteint après le franchissement du mot $s\sigma$.

Le langage défini par un automate à états finis est l'ensemble des séquences d'événements permettant d'atteindre un état marqué.

$$L(A) = \left\{ s \in \Sigma^* \middle| \delta(q_0, s) \in Qm \right\}$$

e) Les générateurs de langage

Bien que l'écriture soit similaire, la définition d'un générateur de langage est différente de la définition des automates à états finis. Deux langages différents peuvent être extraits d'un générateur de langage.

$G := (X, \Sigma, \delta, x_0, X_m)$ avec

X l'ensemble d'états

Σ l'alphabet des évènements

$\delta : X \times \Sigma \to X$ la fonction de transition partielle de changement d'état sur occurrence d'un événement

x_0 l'état initial de l'automate à états

$Xm \subseteq X$ l'ensemble des états marqués

Le langage décrivant le comportement clos généré par *G* est l'ensemble des séquences d'évènements autorisés par le générateur. Le marquage des états n'est pas utilisé dans cette description.

$$L(G) = \left\{ s \in \Sigma^* \middle| \delta(x_0, s)! \right\}$$

Le langage marqué généré par *G* est décrit par une fonction comportant l'ensemble des séquences d'évènements tel que l'état final atteint soit un état marqué. Il s'agit de la même définition que le langage généré par un automate à états.

$$Lm(G) = \left\{ s \in \Sigma^* \middle| \delta(x_0, s) \in Xm \right\}$$

Le langage marqué défini par un générateur est inclus dans le langage du comportement clos. Toutes les séquences dans lesquelles l'état final atteint n'est pas marqué sont retirées.

$$Lm(G) \subseteq L(G)$$

L'ensemble d'états atteignables correspond aux états pouvant être atteints pour tous les mots de *L*(*G*).

$$X_r(G) = \left\{ x \in X \middle| \exists s \in \Sigma^*, \delta(x_0, s) = x \right\}$$

Un état co-atteignable est un état permettant d'atteindre un état marqué par un mot. L'ensemble des états co-atteignables est l'ensemble des états permettant d'atteindre un état marqué par une succession d'évènements.

$$X_{cr}(G) = \left\{ x \in X \middle| \exists s \in \Sigma^*, \delta(x, s) \in X_m \right\}$$

Un système est appelé parfait si tous les états pouvant être atteints depuis l'état initial permettent également d'atteindre un état marqué et si tous les états permettant d'atteindre un état marqué sont atteignables depuis l'état initial.

$$X_r(G) = Xcr(G)$$

Le préfixe d'un générateur de langage est l'ensemble des mots qui permettent d'atteindre un état marqué à partir de l'état initial.

$$\overline{L_m}(G) = \left\{ s \in \Sigma^* \middle| \exists s' \in \Sigma^*, \delta(x_0, ss') \in Q_m \right\}$$

Un générateur de langage est dit « non bloquant » si tous les états permettent d'atteindre un état marqué par une succession d'évènements. Cela signifie que tout état atteint par une succession d'évènements permet toujours de pouvoir atteindre un état marqué.

$$L_m(G) = \overline{L_m}(G)$$

Toutes ces notations sont très utilisées dans les applications à base de des systèmes à évènements discrets. Il est important de connaître ces opérateurs car ils seront pris en compte dans la modélisation. Les opérateurs de composition de générateurs d'automates sont également importants pour la modélisation car ils peuvent être utilisés pour rassembler des modèles partiels d'un système en vue de former un modèle de comportement global.

f) Les opérateurs mathématiques de composition de générateurs de langages

Les opérateurs mathématiques de type produit permettent de synchroniser les générateurs de langages entre eux. Ils sont notamment utilisés pour combiner des petits modèles décrits sous la forme de générateurs de langage en un modèle complexe difficilement réalisable directement.

Suivant les auteurs, différentes définitions des produits existent. Pour la suite de nos travaux, nous posons la même définition que celles définies dans [Akesson, 2003], [Akesson, 2007], [Supremica].

(i) Le produit synchrone

Le produit synchrone est utilisé pour représenter le comportement synchronisé de deux générateurs de langage. Dans cette composition, on ne prend pas en compte les évènements communs aux alphabets des deux automates et qui n'apparaissent pas en même temps depuis l'état courant.

Soient $G^A = \left\langle X^A, \Sigma^A, \delta^A, {x_0}^A, {X_m}^A \right\rangle$ et $G^B = \left\langle X^B, \Sigma^B, \delta^B, {x_0}^B, {X_m}^B \right\rangle$, le produit synchrone est noté $L(G^A)||L(G^B)$ et il est défini de la façon suivante :

$L(G_A) \| L(G_B) = L\left(\left\langle X^A \times X^B, \Sigma^A \cup \Sigma^B, \delta^{A\|B}, {x_0}^A \times {x_0}^B, {X_m}^A \times {X_m}^B \right\rangle \right)$ avec

$$\forall x^A \times x^B \in X^A \times X^B,\ \delta^{A\|B}\left(x^A \times x^B, \sigma\right) = \begin{cases} \delta^A(x^A, \sigma) \times \delta^B(x^B, \sigma) & si\ \sigma \in \Sigma(x^A) \cap \Sigma(x^B) \\ \delta^A(x^A, \sigma) \times x^B & si\ \sigma \in \Sigma(x^A) - \Sigma^B \\ x^A \times \delta^B(x^B, \sigma) & si\ \sigma \in \Sigma(x^B) - \Sigma^A \\ non\ défini & autrement \end{cases}$$

(ii) Le produit asynchrone

Le produit asynchrone permet de prendre en compte tous les évènements pouvant occurrer dans l'un ou l'autre des générateurs d'évènements.

Soient $G^A = \left\langle X^A, \Sigma^A, \delta^A, {x_0}^A, {X_m}^A \right\rangle$ et $G^B = \left\langle X^B, \Sigma^B, \delta^B, {x_0}^B, {X_m}^B \right\rangle$, le produit asynchrone est noté $L\left(G^A\right) + L\left(G^B\right)$ et il est défini de la façon suivante :

$L\left(G^A\right) + L\left(G^B\right) = L\left(\left\langle X^A \times X^B, \Sigma^A \cup \Sigma^B, \delta^{A+B}, {x_0}^A \times {x_0}^B, {X_m}^A \times {X_m}^B \right\rangle \right)$ avec

$$\delta^{A+B}\left(x^A \times x^B, \sigma\right) = \begin{cases} \delta^A(x^A, \sigma) \times \delta^B(x^B, \sigma) & si\ \sigma \in \Sigma(x^A) \cap \Sigma(x^B) \\ \delta^A(x^A, \sigma) \times x^B & si\ \sigma \in \Sigma(x^A) - \Sigma(x^B) \\ x^A \times \delta^B(x^B, \sigma) & si\ \sigma \in \Sigma(x^B) - \Sigma(x^A) \end{cases}$$

Lorsque les alphabets des langages sont indépendants, les produits synchrones et asynchrones sont équivalents et donnent le même résultat.

$$\Sigma^A \text{ I } \Sigma^B = \varnothing \Rightarrow L\left(G^A\right) + L\left(G^B\right) = L\left(G^A\right) \| L\left(G^B\right)$$

(iii) Le produit complet

Le produit complet permet de ne prendre en considération que les évènements communs aux alphabets des deux langages. La synchronisation entre les générateurs d'évènements est alors considérée comme totale.

Soient $G^A = \left\langle X^A, \Sigma^A, \delta^A, {x_0}^A, {X_m}^A \right\rangle$ et $G^B = \left\langle X^B, \Sigma^B, \delta^B, {x_0}^B, {X_m}^B \right\rangle$, le produit complet est noté $L\left(G^A\right) \text{I } L\left(G^B\right)$ et il est défini de la façon suivante :

$L\left(G^A\right) \text{I } L\left(G^B\right) = L\left(\left\langle X^A \times X^B, \Sigma^A \text{ I } \Sigma^B, \delta^{A \text{I } B}, {x_0}^A \times {x_0}^B, {X_m}^A \times {X_m}^B \right\rangle\right)$ avec

$$\delta^{A \text{I } B}\left(x^A \times x^B, \sigma\right) = \begin{cases} \delta^A(x^A, \sigma) \times \delta^B(x^B, \sigma) & si\ \sigma \in \Sigma(x^A) \cap \Sigma(x^B) \\ non\ défini & autrement \end{cases}$$

Toutes ces définitions sont extraites est adaptées de [Wonham, 2004].

3) Utilisation des automates à états finis et des générateurs de langages pour modéliser un système automatisé

En utilisant un automate à états finis ou des générateurs de langages pour décrire le comportement d'un système automatisé, les états peuvent prendre un sens physique en étant associés à la situation du système automatisé. La modélisation d'un système automatisé est souvent réalisée par les automates à états dans le cadre des systèmes à évènements discrets. En effet, l'aspect graphique des automates est beaucoup plus convivial que la présentation sous forme de langages, les langages restant un outil parfaitement adapté pour l'exploitation algorithmique.

Les travaux de synthèse et de vérification de la commande s'intéressent particulièrement à l'élaboration de ces algorithmes permettant d'exploiter les modèles de partie opérative décrit sous la forme d'automates à états finis. Les travaux s'intéressent pour la plupart à

l'amélioration des algorithmes mais la construction même du modèle de partie opérative reste empirique. Les modèles de partie opérative rencontrés dans les exemples traités sont, le plus souvent, proposés par les auteurs à partir de leur perception des fonctionnalités du système. La plupart de ces modèles ne tiennent donc pas compte des technologies employées pour réaliser les fonctionnalités et des limites liées à la modélisation.

a) Présentation des travaux de synthèse de la commande de Ramadge et Wonham

L'hypothèse de base utilisée pour appliquer ces travaux est que le procédé à piloter est un générateur spontané d'événements. Le procédé peut donc être modélisé par un générateur d'évènements $G := (X, \Sigma, \delta, x_0, X_m)$.

Les événements admissibles par G sont classés en deux catégories : les évènements contrôlables et les évènements non contrôlables.

$\Sigma = \Sigma_c \cup \Sigma_u$ avec $\Sigma_c \cap \Sigma_u = \emptyset$

On définit par $L(G) = \{ s \in \Sigma^* ;\ \delta(x_0, s)! \}$ le langage clos généré par G. On note $L(G)\backslash s$ le langage admissible commençant par le mot s $L(G)\backslash s := \{ t \in \Sigma^* ; st \in L(G) \}$. L'ensemble des évènements Σ est découpé en deux sous-ensembles d'évènements contrôlables Σ_c et non contrôlables Σ_u. Le langage marqué est l'ensemble des séquences de $L(G)$ dont l'état final atteint est marqué $L_m(G) = \{ s \in L(G) ; \delta(x_0, s) \in Xm \}$. Un superviseur $S : L(G) \to 2^{\Sigma - \Sigma u}$ est une fonction qui restreint l'occurrence d'évènements contrôlables. Le générateur G^S décrit le comportement de la partie opérative supervisée.

$\varepsilon \in L(G^S)$; $s\sigma \in L(G), s\sigma \in L(G^S)$ si et seulement si $\sigma \notin S(s)$; $L_m(G^S) := L(G^S) \cap L_m(G)$

b) Représentation graphique des automates à états finis

Pour la suite de ces travaux, les automates à états finis ne seront plus distinguer des générateurs d'évènements. Le terme automate à états finis sera exclusivement utilisé tout en employant la définition donnée aux générateurs d'évènements.

Les automates à états peuvent être graphiquement représentés par une structure. Les états sont représentés par des cercles. L'état initial est repéré par une flèche entrante. Les états marqués sont symbolisés par une flèche sortante. Les transitions associées à des évènements sont représentées par une flèche allant d'un état vers un autre état sur lequel est inscrit le nom de l'événement associé au franchissement. La flèche comporte un trait perpendiculaire pour repérer les transitions associées à des évènements contrôlables. Les flèches ne comportant pas

ce trait sont des transitions associées à des évènements non contrôlables. Un exemple de ce type de modélisation est présenté sur la Figure 2.6.

c) Problématique liée à l'utilisation de ces outils

Les outils de modélisation basés sur la théorie des langages permettent de représenter, sous certaines hypothèses, le comportement d'un système automatisé. Il existe de très nombreux exemples de modèles dans les travaux liés à la synthèse de la commande et l'exemple de la modélisation d'une machine par un modèle à trois états est couramment cité [Ramadge, 1987]. Les trois états représentent l'activité de la machine : dans l'état initial, la machine est au repos. Lorsqu'une tâche est associée à la machine (événement contrôlable α), un état de travail est actif. Lorsque ce travail est terminé, la machine revient dans son état de repos (événement non contrôlable β). Si une panne survient (événement non contrôlable λ), le modèle atteint un état de défaillance en attendant d'être réparé (événement contrôlable μ). Ce modèle peut être représenté graphiquement par la Figure 2.6.

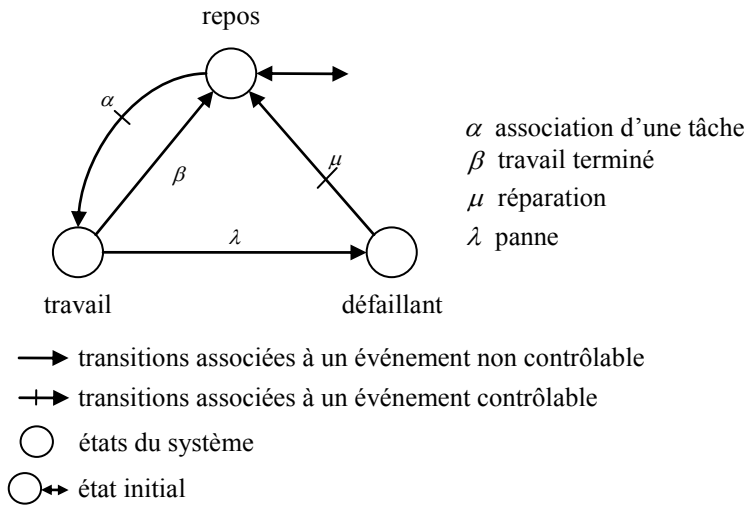


Figure 2.6 : représentation classique du comportement d'une machine par un automate à états finis

Un modèle de ce type peut représenter le comportement d'un système simplifié dans un contexte d'organisation de la production. La gestion de files d'attente, le kanban peuvent facilement être modélisés et simulés par des modèles de ce type pour les applications de la théorie de supervision. Cette représentation des systèmes nécessite une interprétation importante du comportement pour être mise à profit dans les applications d'automatisme. Les états du modèle correspondent bien à des situations du système mais à un niveau de détails très faible et un fort niveau d'abstraction. Quand l'interprétation de Balemi [Balemi, 1992] n'est pas utilisée, les événements ne sont pas rattachés à des évolutions de commande ou de capteurs. Les évènements contrôlables et non contrôlables ne peuvent être assimilés qu'à l'interprétation des changements d'états du système automatisé. Cette interprétation est

arbitraire et n'est pas facilement réaliste vis-à-vis de la problématique liée à l'utilisation d'un automate programmable industriel. De plus, la problématique liée à l'organisation de la production et celle liée l'automatisme se placent à des niveaux de représentation bien différents et ne traitent pas des même sujets. L'organisation de la production ne s'intéresse pas à l'exécution même de la tâche associée à une machine. L'automatisme se place à un niveau de modélisation beaucoup plus fin et prend en considération les constituants physiques du système automatisé et leurs interactions entre eux. La commande du système automatisé est alors vue comme l'étude du comportement du système au niveau des capteurs et des actionneurs. Les modèles couramment utilisés dans les travaux basés sur la théorie de Ramadge & Wonham sont plutôt situés dans la recherche des interdictions d'occurrence d'évènements contrôlables dans un contexte d'ordonnancement de la production. La spécification décrit les limites d'utilisation du système et la restriction ou les règles de comportement que le système doit suivre pour réaliser un fonctionnement spécifié. La spécification est également décrite sous la forme d'un automate à états finis. Les concepts de contrôlabilité d'un modèle sont particulièrement pris en compte puisqu'un état bloquant dans la solution génère un arrêt de la production ce qui est un élément déterminant dans l'organisation de la production.

Les différentes approches issues de cette théorie de supervision montrent bien cette problématique : les approches décentralisées considèrent que le système automatisé est constitué d'entités autonomes avec des relations d'échange d'informations entre elles. Chaque entité est modélisée indépendamment. L'approche décentralisée permet alors de combiner les modèles de comportement de chaque entité prise indépendamment. Les spécifications sont également combinées ensemble pour faciliter la description de comportement attendu par le système automatisé.

d) Modélisation du vérin par les SED

Ce que nous venons d'évoquer ne permet pas de représenter le comportement de vérin de la Figure 2.3. En effet, le vérin nécessite d'être modélisé sans prendre en considération le comportement de la commande qui le pilote, ce n'est pas un générateur spontané d'évènements et il ne peut pas seul effectuer des demandes de déplacements.

L'interprétation de Balemi est donc nécessaire pour modéliser le comportement de ce vérin. Cette interprétation nécessite de déterminer quelles sont les informations qui sont échangées entre le modèle de partie opérative du vérin et le modèle de commande supervisée. Les échanges d'informations correspondant aux entrées/sorties du modèle de partie opérative. Les messages reçus par le modèle de partie opérative provenant du modèle de commande supervisée sont alors assimilés aux évènements commandables. Les messages envoyés par le modèle de partie opérative vers le modèle de commande supervisée sont alors assimilés aux évènements non-commandables. Cette interprétation permet une vision plus réaliste des notions de commandes et de capteurs telles qu'elles sont perçues lors de l'utilisation d'automates programmables industriels. Cependant, cette interprétation se limite à l'envoi de requêtes et à la réception des réponses.

Les modèles de partie opérative présentés par Balemi utilisant l'interprétation peuvent être adaptés pour représenter le comportement du vérin. Dans [Balemi, 1992], un modèle de comportement d'une vanne est présenté. En faisant l'hypothèse que la vanne est actionnée par un vérin de technologie similaire à celui qui est présenté, il suffit d'adapter le vocabulaire de la vanne à celui du vérin. Le modèle est obtenu par l'assemblage de différents automates à états finis (Figure 2.7). Un premier modèle décrit le processus de fonctionnement qui correspond au comportement fondamental du système. Toutes les évolutions du système ne sont pas représentées. Les commandes générant une action ne sont pas représentées : seules les conséquences de ces actions sont représentées au travers les évènements non commandables. Les modèles suivants représentent le comportement des actions. Ces derniers modèles déterminent quelles sont les conditions nécessaires pour qu'une action soit effectuée et quelles sont les conséquences sur les évènements non commandables lorsque les actions sont exécutées.

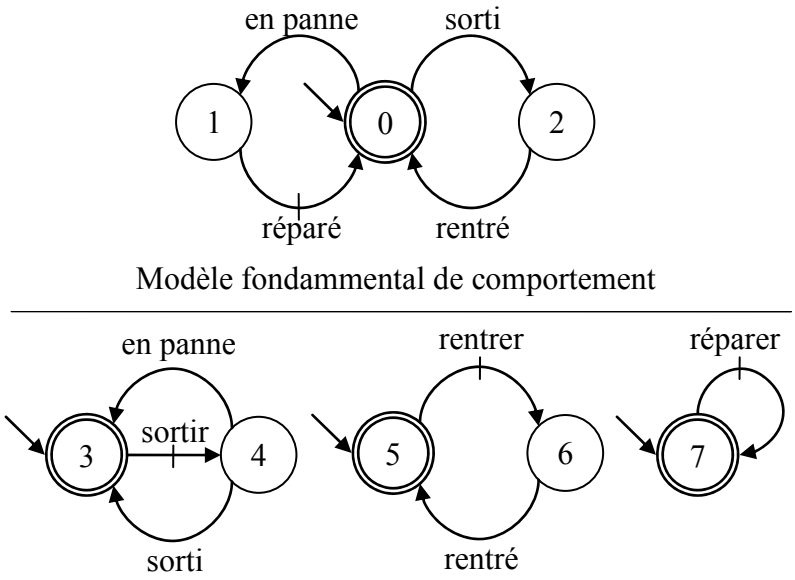


Figure 2.7 : Etapes de modélisation d'un vérin par [Balemi, 2002]

Le modèle fondamental de comportement décrit d'une manière générale comment le système doit fonctionner. Il ne s'agit que de définir les relations causes/conséquences dans le fonctionnement du système. Tel que ce modèle est conçu, le vérin est en position rentrée à la mise en route du système et le système n'est pas signalé en panne. Une panne ne peut être détectée que lorsque la tige du vérin est rentrée. Lorsqu'une panne arrive, il faut que la réparation soit exécutée pour remettre en fonction le système. L'état initial est indiqué comme étant un état marqué. Les modèles comportementaux des actions désignent quels sont les besoins pour qu'une action commandable puisse être exécutée. Ces modèles sont complémentaires du modèle fondamental de comportement.

L'assemblage de tous ces modèles à états finis est réalisé par une composition spécifique définie mathématiquement dans [Inan, 1989]. Le résultat de cette composition est représenté sur la Figure 2.8.

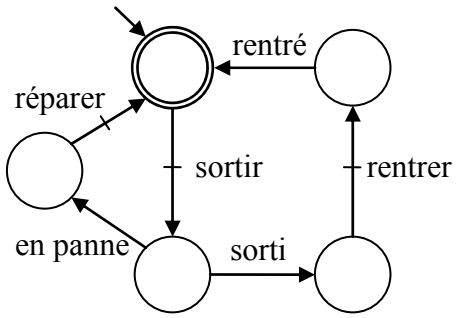


Figure 2.8 : assemblage du modèle de comportement du vérin

Le modèle obtenu représente le comportement complet du vérin. Le cycle de fonctionnement normal est constitué de la succession d'évènements commandables et non commandables {*sortir*, *sorti*, *rentrer*, *rentré*}. La panne ne peut être détectée qu'après l'occurrence de l'événement commandable « *sortir* ». Lorsque le système est en panne, il faut attendre que l'événement commandable « *réparer* » se présente pour que l'événement commandable « *sortir* » puisse de nouveau se produire.

Ce modèle peut être utilisé dans des applications mais la démarche reste très intuitive. Les échanges d'informations représentées par des évènements commandables et non commandables entre la partie opérative et la partie commande permet une forme de représentation du comportement du système automatisé. L'utilisation d'un automate programmable industriel est néanmoins essentiellement basée sur des échanges de signaux booléens. Les échanges d'évènements tels que présentés dans cette approche ne permettent pas de représenter directement ce mode d'échange car l'association des évolutions physiques et des évènements n'est pas déterminée.

e) Prise en compte des signaux booléens

Les signaux échangés entre la partie opérative et la partie commande sont majoritairement des signaux booléens. Pour représenter un comportement le plus proche possible du comportement effectif de la partie opérative, il faut étudier de près les échanges de signaux booléens. L'étude des signaux échangés entre la partie commande et la partie opérative nécessite d'étudier préalablement l'instrumentation du système automatisé. L'instrumentation est fortement liée aux technologies utilisées sur la partie opérative. Il ne s'agit pas seulement d'isoler des situations spécifiques de la partie opérative mais de tenir compte des capacités de mesure et de commande du système automatisé. L'implantation de capteurs et leurs emplacements vont permettre une vision partielle de l'état de la partie opérative du système automatisé. Les technologies de commande sont également très importantes : les actions ne sont exercées qu'en fonction des combinaisons de signaux de commande. Certaines

technologies limitent les actions pouvant être effectuées. L'arrêt en cours de mouvement pour un vérin n'est pas toujours possible suivant la technologie utilisée pour réaliser le pilotage de ce vérin. L'utilisation de modèles utilisant les signaux d'échange booléens nécessite donc de prendre en compte le comportement global du système mais aussi d'étudier au préalable les technologies mises en œuvre sur le système automatisé.

L'approche évènementielle proposée par l'interprétation de Balemi peut être conservée mais il est nécessaire d'associer aux évènements un lien avec les signaux booléens échangés entre la partie commande et la partie opérative. Le principe consiste à associer des évènements aux changements d'état d'un signal booléen. Chaque signal booléen pouvant prendre deux états, deux évènements peuvent être déterminés à partir de ces changements d'états. Les états des signaux booléens sont toujours associés à deux niveaux : haut niveau et bas niveau. Les changements de niveaux des variables booléennes sont appelés fronts. Le front montant ↑ correspond à un changement du signal booléen d'un niveau bas à un niveau haut. Le front descendant ↓ correspond à un changement d'un niveau haut à un niveau bas. Le modèle présenté sur la Figure 2.9 montre comment peuvent être représentés les évolutions d'un signal booléen. Les deux états correspondent aux valeurs du signal « *S* ». Les transitions représentent les évolutions de ces signaux par les fronts. La modélisation du comportement du système automatisé peut utiliser cette forme de représentation couplée à l'interprétation de Balemi. Les évènements ont alors un sens physique avec le comportement du système. La représentation de signaux booléens physiques permet de distinguer quels sont les signaux de commande et les signaux des capteurs uniquement à partir de ce qu'il représente. La distinction réalisée par la barre sur la transition entre les évènements commandables des évènements non commandables peut donc être suspendue dans la suite pour simplifier la représentation des modèles.

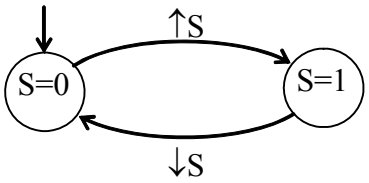


Figure 2.9 : modélisation des évolutions des signaux booléens

f) Conséquences sur la modélisation de la partie opérative

Ce type de représentation du comportement du système implique certaines contraintes dans la modélisation. Le comportement des signaux booléens représentés à l'aide de fronts a de fortes incidences dans la modélisation. Au cours du fonctionnement du système, chaque signal booléen évolue avec une alternance entre les fronts montants et les fronts descendants. La modélisation de la partie opérative à l'aide des automates à états finis comporte donc des propriétés particulières. Le modèle de partie opérative proposé par un concepteur est généré à

partir de ses connaissances du fonctionnement de façon empirique. Certaines propriétés peuvent être testées sur le modèle de partie opérative pour vérifier qu'il n'y a pas d'erreurs dans la construction. Dans les applications de génération de la commande, le but recherché est de concevoir une commande satisfaisant aux contraintes de fonctionnement imposées par le cahier des charges. Une des propriétés pouvant être mise en avant avec l'utilisation de modèle de partie opérative utilisant l'interprétation de Balemi est l'indépendance de la commande vis-à-vis des informations issues des capteurs.

g) Bilan

La modélisation de la partie opérative prend des aspects différents suivant le contexte dans lequel elle est utilisée. La modélisation utilisée dans la synthèse de Ramadge et Wonham décrit le comportement de la partie opérative à un niveau fonctionnel. Le procédé est perçu comme un générateur spontané d'évènements dont certains peuvent être inhibés. L'interprétation de Balemi apporte une vision plus proche des échanges réalisés entre la partie opérative et la partie commande par une représentation des commandes forcées par la partie commande. Néanmoins, la représentation de ces informations reste éloignée des signaux réels échangés entre l'API et la partie opérative. Parmi les critères nécessaires à la réalisation d'un modèle de partie opérative, il est nécessaire de s'assurer que le modèle comporte toutes les évolutions possibles des commandes car un modèle de partie opérative partiel ne permet pas l'exhaustivité des situations pouvant être atteintes par la partie opérative.

4) Propriété de commandabilité d'un modèle de partie opérative

Le modèle de partie opérative nécessite de ne pas brider les évolutions possibles de la partie commande. Le modèle de commande est donc libre de pouvoir émettre des évènements commandables quelle que soit la situation dans laquelle se trouve la partie opérative. Certains de ces évènements commandables peuvent entraîner des comportements dangereux du système automatisé mais ces comportements peuvent être déclarés et interceptés dans les spécifications comportementales du système automatisé. Un modèle de partie opérative peut donc être mis à l'épreuve avec une propriété de complétude vérifiant si les évènements commandables sont tous pris en compte depuis tous les états atteints par le modèle. Cette propriété vérifie donc que le modèle de comportement du système ne comporte aucune restriction d'évolution de la commande [Rohee, 2008b]. Il est nécessaire de décrire, sous la forme d'une spécification, le comportement libre des signaux commandables. Ce modèle est obtenu en représentant le comportement de chaque signal par un automate à états finis et en effectuant la composition étendue entre chacun de ces modèles. L'automate à états finis finalement obtenu est appelé modèle de spécification. L'état initial du modèle de spécification doit être cohérent vis-à-vis de l'état initial du modèle de partie opérative.

En supposant que le modèle de partie opérative, noté P, et le modèle de spécification noté S soient représentés sous la forme de deux automates à états, la propriété à vérifier revient à montrer que le langage de S réduit à l'alphabet de P est inclus dans le langage de P réduit à l'alphabet de S. L'équivalence suivante permet de vérifier cette condition de façon simple et formelle par des compositions d'automates et des équivalences de langages.

$P = \langle Q_P, \Sigma_P, \delta_P, q_{0_P} \rangle$

$S = \langle Q_S, \Sigma_S, \delta_S, q_{0S} \rangle$

Les compositions utilisées dans la propriété sont définies dans [Akesson, 2003]. Le logiciel Supremica utilise ces mêmes définitions pour effectuer les compositions.

$(L(S) + L(P)) \| L(S) = L(S) \| L(P)$ (équation 1)

a) Propriété de vérification de la complétude vis à vis de la commande

Soit $Am = \langle Q_{Am}, \Sigma_{Am}, \delta_{Am}, q_{0Am} \rangle$ l'automate à états décrivant la spécification de complétude, soit $P = \langle Q_P, \Sigma_P, \delta_P, q_{0P} \rangle$ l'automate à états décrivant un modèle de partie opérative, on souhaite vérifier que le modèle de partie opérative peut répondre à toutes les sollicitations de la spécification de complétude.

De façon formelle, la spécification de complétude peut s'écrire :

$\forall \sigma \in \Sigma_P \text{ I } \Sigma_{Am}$, *si* $\exists \delta_{Am}(q_{Am}, \sigma)$ *alors* $\exists \delta_P(q_P, \sigma)$

Cette caractéristique est vérifiée si et seulement si la propriété suivante est vérifiée :

sync(Am,P)≡sync(broad(Am,P),P)

b) Démonstration de la propriété.

Soient $Am = \langle Q_{Am}, \Sigma_{Am}, \delta_{Am}, q_{0Am} \rangle$ et $P = \langle Q_P, \Sigma_P, \delta_P, q_{0P} \rangle$, on veut démontrer que *sync(Am,P)≡sync(broad(Am,P),Am)* équivaut à dire que :

sync(sync(Am,P),Am)≡sync(broad(Am,P),Am) ou bien encore :

$\forall \sigma \in \Sigma_P \text{ I } \Sigma_{Am}$, *si* $\exists \delta_{Am}(q_{Am}, \sigma)$ *alors* $\exists \delta_P(q_P, \sigma)$.

Démontrer que $(L(P) + L(Am)) \| L(Am) = L(P) \| L(Am)$ revient à démontrer que $(L(P) + L(Am)) \| L(Am) = (L(P) \| L(Am)) \| L(Am)$ car l'opérateur || est associatif $(L(A) \| L(B)) \| L(C) = L(A) \| (L(B) \| L(C))$ et que $L(A) \| L(A) = L(A)$.

On va donc montrer que $\left(L(P)+L(Am)\right)\|L(Am)=L(P)\|L(Am)$ équivaut à dire que $\forall\sigma\in\Sigma_P \text{ I } \Sigma_{Am}$, *si* $\exists\delta_{Am}(q_{Am},\sigma)$ *alors* $\exists\delta_P(q_P,\sigma)$.

Soit $\langle q_P.q_{Am}\rangle$ un état de sync(P||Am),

En composant les 2 produits, on peut dire que :

$$\delta_{sync(broad(P,Am),Am)}(q_P.q_{Am}.q_{Am},\sigma)=$$

$$=\begin{cases}
\delta_P(q_P,\sigma).\delta_{Am}(q_{Am},\sigma).\delta_{Am}(q_{Am},\sigma) & si\ \exists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
\delta_P(q_P,\sigma).\delta_{Am}(q_{Am},\sigma).q_{Am} & si\ \exists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
\delta_P(q_P,\sigma).q_{Am}\quad .q_{Am} & si\ \exists\delta_P(q_P,\sigma)\ et\ \nexists\delta_{Am}(q_{Am},\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
\delta_P(q_P,\sigma).q_{Am}\quad .\delta_{Am}(q_{Am},\sigma) & si\ \exists\delta_P(q_P,\sigma)\ et\ \nexists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
q_P\quad .\delta_{Am}(q_{Am},\sigma).\delta_{Am}(q_{Am},\sigma) & si\ \nexists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
q_P\quad .\delta_{Am}(q_{Am},\sigma).q_{Am} & si\ \nexists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
q_P\quad .q_{Am}\quad .\delta_{Am}(q_{Am},\sigma) & si\ \nexists\delta_P(q_P,\sigma)\ et\ \nexists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
non\ défini & autres\ cas
\end{cases}$$

Soit, après simplification :

$$\delta_{sync(broad(P,Am),Am)}(q_P.q_{Am}.q_{Am},\sigma)=$$

$$=\begin{cases}
\delta_P(q_P,\sigma).\delta_{Am}(q_{Am},\sigma).\delta_{Am}(q_{Am},\sigma) & si\ \exists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
\delta_P(q_P,\sigma).q_{Am}\quad .q_{Am} & si\ \exists\delta_P(q_P,\sigma)\ et\ \nexists\delta_{Am}(q_{Am},\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
q_P\quad .\delta_{Am}(q_{Am},\sigma).q_{Am} & si\ \nexists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
q_P\quad .q_{Am}\quad .\delta_{Am}(q_{Am},\sigma) & si\ \nexists\delta_P(q_P,\sigma)\ et\ \nexists\delta_{Am}(q_{Am},\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
non\ défini & autres\ cas
\end{cases}$$

et

$$\delta_{sync(sync(P,Am),Am)}(q_P.q_{Am}.q_{Am},\sigma)=$$

$$=\begin{cases}
\delta_P(q_P,\sigma).\delta_{Am}(q_{Am},\sigma).\delta_{Am}(q_{Am},\sigma) & si\ \exists\delta_P(q_P,\sigma)\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
\delta_P(q_P,\sigma).q_{Am}\quad .q_{Am} & si\ \exists\delta_P(q_P,\sigma)\ et\ \sigma\notin\Sigma_{Am}\\
q_P\quad .\delta_{Am}(q_{Am},\sigma).\delta_{Am}(q_{Am},\sigma) & si\ \sigma\notin\Sigma_P\ et\ \exists\delta_{Am}(q_{Am},\sigma)\\
non\ défini & autres\ cas
\end{cases}$$

En écrivant la propriété,

$$sync(broad(P,Am),Am)=\left\langle Q_P\times Q_{Am}\times Q_{Am},\Sigma_P\cup\Sigma_{Am},\delta_{sync(broad(P,Am),Am)},q_{0P}.q_{0Am}.q_{0Am}\right\rangle$$

$$sync(sync(P,Am),Am)=\left\langle Q_P\times Q_{Am}\times Q_{Am},\Sigma_P\cup\Sigma_{Am},\delta_{sync(sync(P,Am),Am)},q_{0P}.q_{0Am}.q_{0Am}\right\rangle$$

Pour avoir $sync\left(sync\left(P,Am\right),Am\right) \equiv sync\left(broad\left(P,Am\right),Am\right)$, il faut et il suffit que $\delta_{sync\left(sync\left(P,Am\right),Am\right)} = \delta_{sync\left(broad\left(P,Am\right),Am\right)}$

Pour avoir $\delta_{sync\left(sync\left(P,Am\right),Am\right)} = \delta_{sync\left(broad\left(P,Am\right),Am\right)}$, il faut que :

$\forall \left\langle q_P.q_{Am}.q_{Am} \right\rangle \in Q_P \times Q_{Am} \times Q_{Am}$ et $\forall \sigma \in \Sigma_P \cup \Sigma_{Am}$,

$$\delta_{sync\left(sync\left(P,Am\right),Am\right)}\left(q_P.q_{Am}.q_{Am},\sigma\right) = \delta_{sync\left(broad\left(P,Am\right),Am\right)}\left(q_P.q_{Am}.q_{Am},\sigma\right).$$

En comparant les équations (1) et (2), cette condition est vérifiée si et seulement si $\forall \sigma \in \Sigma_P \cap \Sigma_{Am}$ *si* $\exists \delta_{Am}\left(q_{Am},\sigma\right)$ *alors* $\exists \delta_P\left(q_P,\sigma\right)$ CQFD

c) Illustration de la propriété

La propriété montre une relation de récurrence. Soient P et Am les générateurs de modèles de procédé et de la spécification de commandabilité. On appelle $\left\langle q_P,q_{Am}\right\rangle$ les états pris par ces deux modèles à un instant. A l'initialisation, $\left\langle q_P,q_{Am}\right\rangle = \left\langle q_{0P},q_{0Am}\right\rangle$, la relation de récurrence sur occurrence d'un événement σ est définie par

$$\delta\left(\left\langle q_P,q_{Am}\right\rangle,\sigma\right) = \begin{cases} \left\langle \delta\left(q_P,\sigma\right),q_{Am}\right\rangle & si\ \exists\delta\left(q_P,\sigma\right)\ et\ \nexists\delta\left(q_{Am},\sigma\right) \\ \left\langle \delta\left(q_P,\sigma\right),\delta\left(q_{Am},\sigma\right)\right\rangle & si\ \exists\delta\left(q_P,\sigma\right)\ et\ \exists\delta\left(q_{Am},\sigma\right) \\ \left\langle q_P,\delta\left(q_{Am},\sigma\right)\right\rangle & si\ \sigma\notin\Sigma_P\ et\ \exists\delta\left(q_{Am},\sigma\right) \end{cases}$$

La propriété est vérifiée si la définition de la fonction de transition est complète (le cas $\delta\left(\left\langle q_P,q_{Am}\right\rangle,\sigma\right) avec\ \sigma \in \Sigma_P\ et\ \exists\delta\left(q_{Am},\sigma\right)$ n'est jamais rencontré)

L'exemple présenté Figure 2.10 montre différents cas de figure rencontrés à l'occurrence d'un événement depuis un état atteint par les modèles *Am* et *P*. L'automate partiel *P* représente le modèle de partie opérative et l'automate partiel *Am* la spécification de commandabilité partielle. On étudie les événements tolérés par la propriété pour toutes les situations susceptibles de se produire depuis les états courants de *P* et *Am*.

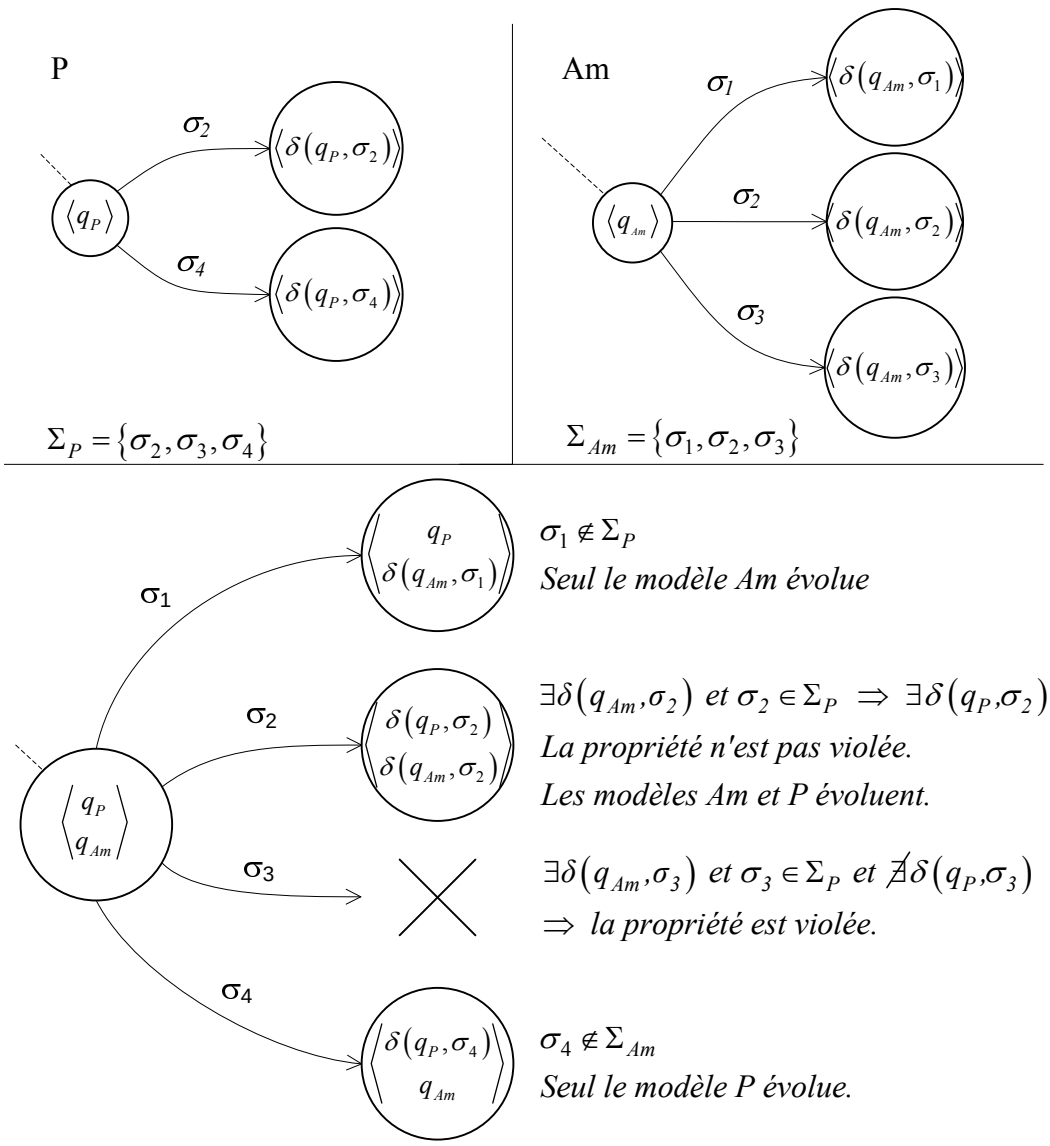


Figure 2.10 : étude de différents cas dans l'application de la propriété

Les automates *P* et *Am* ont respectivement atteint un état q_P et q_{Am} après une séquence d'événements donnée. Cinq événements différents sont acceptés par *P* ou *Am*. Les événements σ_1, σ_2, σ_3 sont des évènements contrôlables faisant l'objet de la vérification. L'évènement σ_4 n'est pas pris en compte dans la spécification car il s'agit d'un événement non contrôlable. L'objectif dans cet exemple est de s'assurer que toutes les évolutions de la spécification de commandabilité sont acceptées par le modèle de procédé. L'événement σ_1 admissible par la spécification de commandabilité *Am* est toléré car il n'est pas pris en compte par le modèle de procédé *P*. On fait l'hypothèse que le modèle de commandabilité peut comporter des événements qui ne sont pas pris en compte par le modèle de procédé. Cette hypothèse facilite la vérification du modèle de procédé partiel. L'événement σ_2 fait évoluer les deux modèles simultanément car les deux modèles sont réceptifs à cet événement. L'événement σ_3 n'est pas toléré par la propriété : l'événement est demandé par la spécification et devrait être pris en compte par le modèle de partie opérative car il est compris

dans l'alphabet de *P*. La propriété n'est pas vérifiée puisque l'événement σ_3 viole la propriété. L'événement σ_4 n'est pas testé par la spécification. Attention, la propriété n'est pas conçue pour détecter une surabondance d'événements dans le modèle de procédé.

d) Exemple d'utilisation de la propriété sur un exemple simple

Dans l'exemple de la Figure 2.11, considérons les modèles de procédé *P* et de spécification *S* suivants :

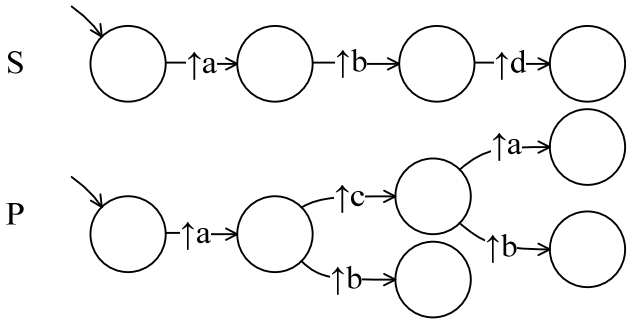


Figure 2.11 : illustration de la propriété sur un exemple abstrait

L'ensemble des mots du langage de S est

- {↑a, ↑a↑b, ↑a↑b↑d}

Si on réduit le langage de S à l'alphabet de P, on obtient

- {↑a, ↑a↑b}

L'ensemble des mots du langage de P est

- {↑a, ↑a↑b, ↑a ↑c, ↑a↑c↑a, ↑a↑c↑b}

Si on réduit le langage de P à l'alphabet de S, on obtient

- {↑a, ↑a↑b, ↑a↑a}

On vérifie bien que le langage de *S* réduit à l'alphabet de *P* est inclus dans le langage de *P* réduit à l'alphabet de *S*. La propriété doit donc être vérifiée

La composition *S*+*P* donne l'automate de la Figure 2.12.

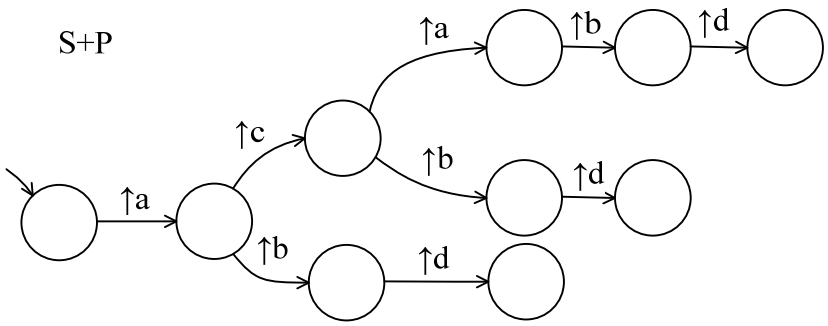


Figure 2.12 : automate résultant du produit étendu entre S et P

La propriété est vérifiée dans le cas de cet exemple car toutes les séquences d'évènements de S appartenant à l'alphabet de P sont admises par le langage de P réduit à l'alphabet de S (Figure 2.13) :

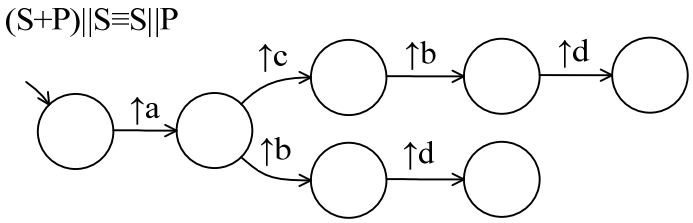


Figure 2.13 : modèle obtenu dans l'application de la propriété

La conclusion de cet exemple est que le modèle de procédé est compatible avec la spécification de commandabilité proposée. Il est à noter que cette propriété peut également être utilisée à d'autres fins. En effet, un modèle contraint correspondant à une restriction du comportement du procédé peut être mis à l'épreuve pour confronter avec le modèle de procédé.

5) Conclusion

La démarche de conception par l'étude fonctionnelle du comportement de la partie opérative permet de décrire formellement la perception du concepteur de ce comportement. En revanche, elle est très éloignée de la conception physique du système et des choix technologiques qui ont été faits. Les données échangées entre la partie opérative et la partie commande ne peuvent pas être exploitées pour utiliser le modèle de partie opérative dans des applications en ligne.

Le modèle de la partie opérative peut être obtenu alternativement à cette étude fonctionnelle du comportement du système automatisé. Basée sur la démarche de mise en œuvre des systèmes automatisés dans le secteur industriel, la modélisation comportementale de la partie opérative peut également se baser sur l'analyse physique des signaux échangés entre la partie

commande et la partie opérative. Il s'agit alors de modéliser le comportement du système automatisé au travers les entrées/sorties booléennes physiques de l'automate programmable industriel. Le modèle utilisant cette description n'a pas la même utilisation que les modèles basés sur l'étude fonctionnelle précédemment décrite : la commande n'est plus appréhendée sous forme d'ordonnancement d'opérations élémentaires mais sur les sollicitations et les retours de signaux échangés entre la partie commande et la partie opérative. L'implémentation des applications développées est facilitée par le faible niveau d'abstraction mais la modélisation de la partie opérative nécessite de prendre en compte des comportements temporels extrêmement difficiles à considérer à cause des phénomènes de synchronisation. L'étude des sous-ensembles mécaniques composant la partie opérative produit apporte une autre vision à la modélisation de la partie opérative. Cette démarche de modélisation n'est toutefois pas sans conséquence sur les outils de modélisation.

Pour pouvoir utiliser les outils de modélisation classiques et les différents travaux ayant pu être effectués, il est nécessaire de formuler des hypothèses sur les phénomènes ne pouvant pas être pris en considération. La démarche de modélisation de la partie opérative basée sur l'étude des composants physiques à l'aide de différents outils est présentée dans le chapitre 3.

Chapitre 3

Démarche de modélisation de la partie opérative et exploitation pour la modélisation discrète

La modélisation de la partie opérative peut être réalisée par une description fonctionnelle du comportement perçu par le concepteur mais elle suscite de nombreuses interrogations quant à l'obtention même du modèle. En effet, la toute première question à se poser concerne l'utilisation même du modèle. Est-il nécessaire pour élaborer la commande, pour valider ou vérifier la commande avant implantation, pour réaliser le diagnostic du système, pour suivre et visualiser les évolutions du système automatisé ? Peut-il être identifié par apprentissage progressif du comportement ? Le modèle obtenu est, par conséquent, différent suivant l'objectif recherché. Il est également évident que l'interprétation laissée au concepteur ne permet pas de concevoir un modèle de partie opérative unique. Enfin, l'utilisation du modèle de partie opérative dans des applications concrètes favorisant le transfert technologique sont difficiles à mettre en œuvre puisque l'approche fonctionnelle est déconnectée de la technologie mise en jeu dans le fonctionnement de la partie opérative. La Figure 3.1 présente le contexte de la modélisation de la partie opérative.

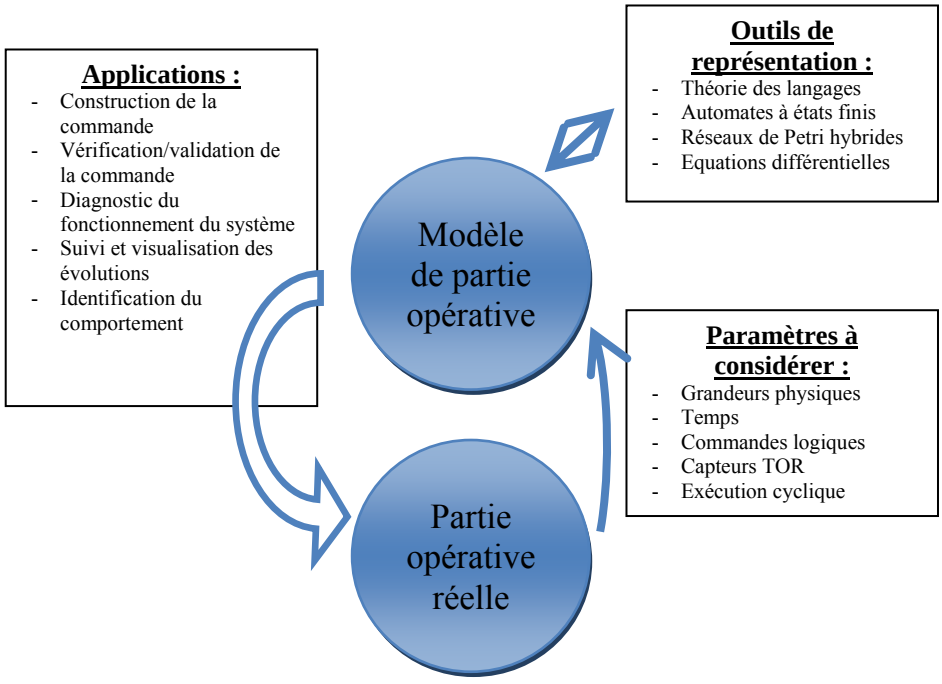


Figure 3.1 : contexte de la modélisation de la partie opérative

Pour palier ces interrogations, nous proposons une nouvelle démarche de modélisation [Rohee, 2006] de la partie opérative basée non plus sur une description fonctionnelle mais sur les composants physiques du système. Cette démarche s'appuie sur des outils de modélisation couramment utilisés en recherche pour décrire le comportement de la partie opérative à partir d'une étude des constituants physiques composant le système.

Le modèle de partie opérative est construit avec pour objectif de représenter les évolutions de la partie opérative telles que les perçoit la partie commande (Figure 3.2). La démarche de modélisation débute par une analyse de la partie opérative réelle. Deux types de modélisation sont alors possibles en fonction de l'utilisation du modèle : la modélisation discrète et la modélisation hybride. Ces modèles ont pour objectif de représenter le comportement de la partie opérative tel que la partie commande le perçoit. Certaines hypothèses nécessitent d'être formulées pour que l'interprétation du modèle avec les outils utilisés soit cohérente avec le comportement supposé.

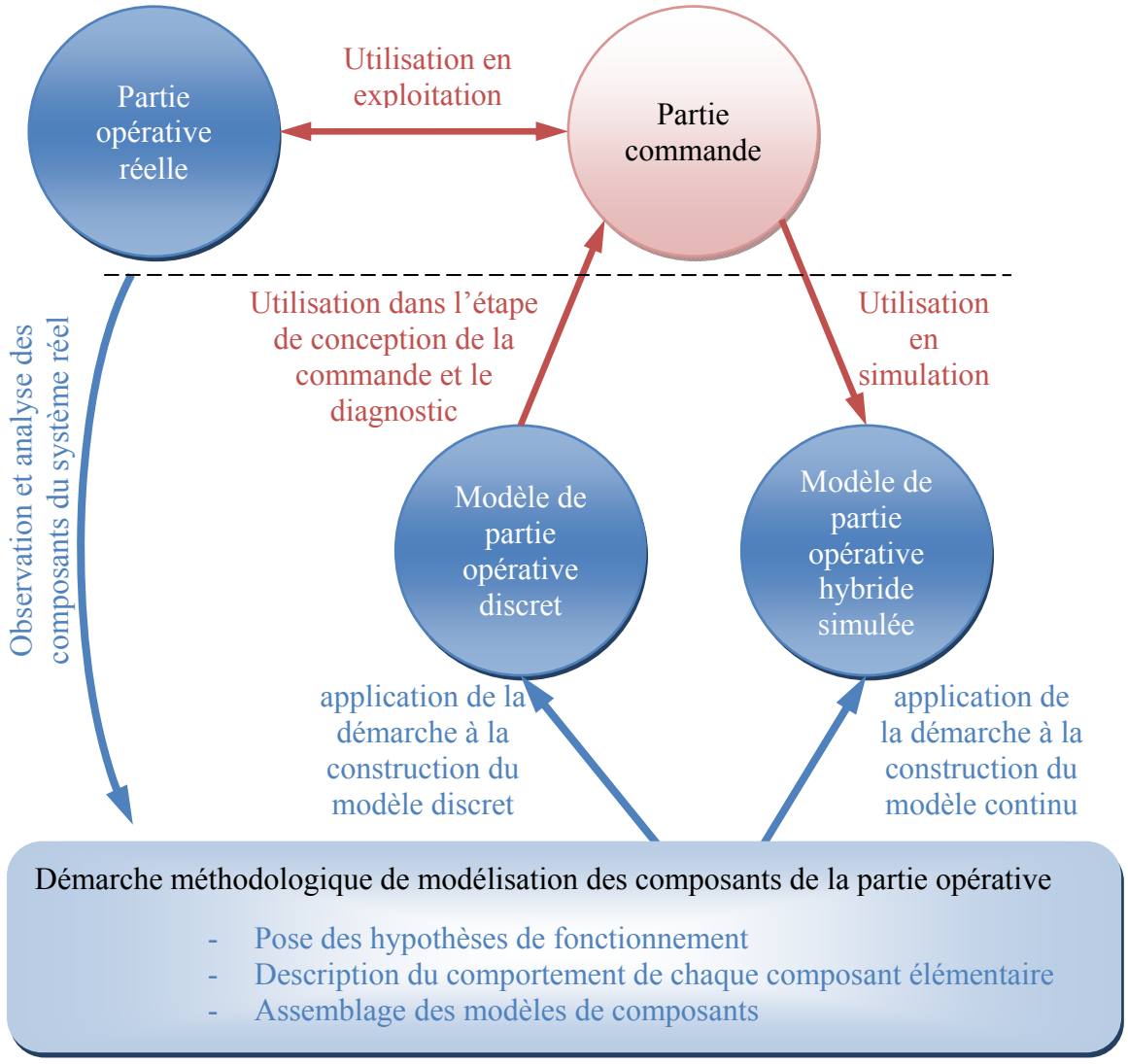


Figure 3.2 : positionnement de la modélisation de la partie opérative

Après la présentation de la démarche, nous présentons la méthodologie de modélisation de chaque sous-ensemble. Le chapitre est illustré par des exemples de composants génériques utilisés dans les systèmes automatisés. Deux implémentations sur système réel complètent le chapitre et montrent notamment l'apport de la démarche en terme de diagnostic et de supervision industrielle. Nous nous intéressons plus particulièrement dans ce chapitre à l'aspect discret de la modélisation, l'aspect hybride étant traité dans le chapitre suivant.

1. Etude de la partie opérative d'un système manufacturier

La modélisation intégrale de la partie opérative n'est pas une opération simple à réaliser. Une partie opérative est généralement composée de constituants physiques standards assemblés dans le but de réaliser les fonctionnalités visées par le système. L'utilisation de constituants standards permet notamment de capitaliser le travail de conception pour réduire le temps de réalisation et le coût de fabrication. Ce principe de découpage en éléments standards et de réutilisation du travail peut également être retenu pour la modélisation d'un système automatisé. Le découpage de la partie opérative peut alors être effectué différemment du découpage fonctionnel précédemment rencontré. Il ne s'agit alors plus de distinguer des fonctionnalités types dans une partie d'un système mais de créer le modèle de partie opérative en fonction des matériels la constituant. En étudiant les technologies utilisées par le système automatisé pour effectuer les actions sur le produit manufacturier, un découpage peut être proposé en fonction des constituants.

a. Les chaînes d'action et d'acquisition

La partie opérative est constituée de chaînes d'action permettant d'agir sur le produit physique à partir de l'ordre de commande envoyé par la partie commande et de chaînes d'acquisition permettant de mesurer les grandeurs physiques évoluant au cours du fonctionnement de la partie opérative. La Figure 3.3 présente le découpage de la partie opérative en différents éléments suivant leurs fonctionnalités.

La chaîne d'action est constituée de différents matériels répondant à des fonctions particulières : le préactionneur a pour but de canaliser l'énergie d'alimentation en fonction des signaux de commande transmis par la partie commande. A la sortie du préactionneur, l'énergie est conditionnée pour alimenter l'actionneur. L'actionneur convertit l'énergie fournie par le préactionneur en énergie d'action. L'effecteur permet d'appliquer l'énergie d'action au produit et d'effectuer l'opération demandée.

Les chaînes d'acquisition sont constituées de capteurs qui mesurent les grandeurs physiques nécessaires à la réalisation des actions. Les capteurs transcrivent les grandeurs physiques mesurées en signaux électriques compatibles avec les acquisitions de la partie commande.

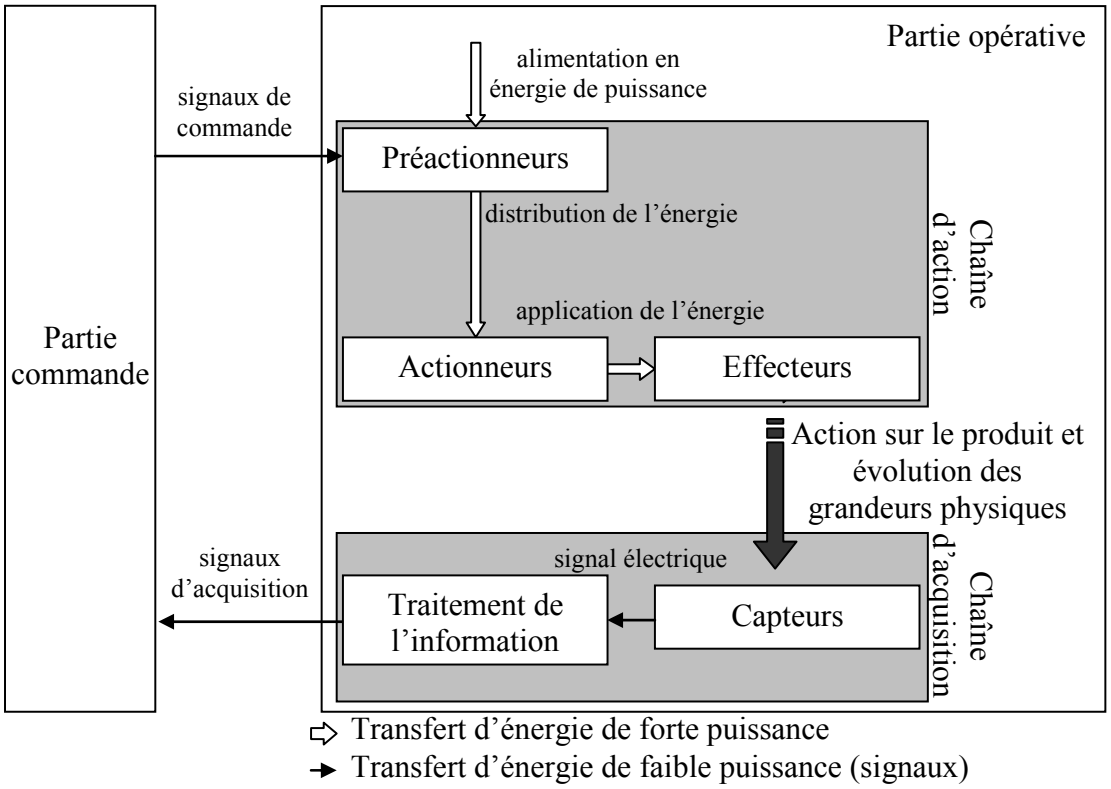


Figure 3.3 : constituants de la partie opérative

Pour palier les inconvénients évoqués précédemment d'une modélisation par description fonctionnelle du comportement perçu par le concepteur, nous proposons d'élaborer le modèle de la partie opérative en modélisant le comportement des chaînes d'action et d'acquisition.

Pour la chaîne d'action, des solutions technologiques sont généralement prédéfinies en fonction du type d'énergie mise en œuvre. Ainsi, les préactionneurs sont souvent associés aux distributeurs pneumatiques ou hydrauliques ou aux relais électriques à contacts. Les actionneurs correspondent à des vérins ou à des moteurs pneumatiques, hydrauliques ou électriques pour les actions mécaniques de déplacements ou de pression de contact.

Dans la chaîne d'acquisition, les capteurs transcrivent une grandeur physique en signal électrique nécessitant d'être traité pour que la partie commande puisse l'interpréter. Les détecteurs comportent à la fois la partie capteur et traitement de l'information pour fournir un signal Tout-Ou-Rien (TOR). Les détecteurs sont souvent utilisés pour indiquer la situation dans laquelle se trouvent les actionneurs ou les effecteurs par une information partielle. Les détecteurs déterminent également des informations sur le produit en cours de traitement sur la partie opérative.

b. Le niveau de détails et degré de précision à considérer dans la modélisation

La modélisation de la partie opérative peut donc passer par la représentation formelle du comportement de chacun des éléments énumérés précédemment. Cette modélisation prend en compte tous les matériels présents sur le système depuis le câblage de la partie commande jusqu'à l'action physique générée.

En fonction de l'utilisation du modèle de partie opérative, l'interprétation du fonctionnement des éléments est plus ou moins importante. Modéliser le système avec la finesse la plus importante possible n'est pas une bonne solution : les outils mathématiques utilisés dans les applications doivent pouvoir exploiter le modèle de partie opérative et les informations qu'il contient et l'exploitation du modèle de partie opérative avec les outils informatiques nécessite de définir un degré de précision adéquat pour ne pas compromettre le résultat final de l'application.

Le comportement hybride de la partie opérative d'un système automatisé est souvent représenté par un modèle discret simplifiant le traitement par les outils informatiques. La modélisation de la partie opérative par un modèle discret passe donc nécessairement par une restriction des informations disponibles dans le modèle. Ce niveau de restriction représente une part importante du travail de modélisation : les grandeurs physiques continues peuvent être échantillonnées, le modèle ne peut tenir compte que des informations discrètes issues des détecteurs…

c. Problématique de la prise en compte du temps

La modélisation du temps peut prendre des formes différentes suivant le comportement qui est spécifiquement étudié. Le temps joue un rôle important dans l'étude du comportement. Une partie opérative réelle évolue au cours du temps suivant un écoulement continu et régulier. Dans la modélisation des systèmes automatisés sous forme d'événements discrets, le temps est discrétisé : l'utilisation d'un API équipé d'un processeur implique une exécution cyclique du traitement du programme de commande. La modélisation du temps sous forme discrète peut prendre des formes diverses selon les besoins : la discrétisation périodique dans laquelle les informations contenues dans le modèle de partie opérative sont rafraichies à une date fixée, la discrétisation cyclique où les informations sont rafraichies pour chaque cycle de l'API. Dans le premier cas, la période de rafraichissement devra être déterminée en fonction des constantes de temps liées à l'API et aux actions de la partie opérative. Dans le second cas, une modélisation de l'API et du cycle d'exécution est indispensable. Le temps peut également être pris en compte plus succinctement en représentant uniquement le séquencement des évolutions successives. Le temps est donc particulièrement délicat à traiter dans la modélisation de la partie opérative notamment car des phénomènes de synchronisation sont

présents sur les entrées/sorties des API et des évolutions de la partie commande. Le temps peut également avoir des influences sur le temps de commutation des différents équipements composant la partie opérative. La modélisation du temps est nécessairement prise en compte dans la modélisation de la partie opérative mais elle peut être représentée implicitement ou explicitement.

2.La modélisation de la partie opérative par les automates à états finis

La modélisation par l'outil mathématique que sont les automates à états finis, passe nécessairement par l'interprétation et l'association entre les possibilités formelles de l'outil et le sens que l'on veut donner au modèle vis-à-vis du comportement observé. Les automates à états, dans leur définition originelle, ne comportent que des états et des transitions et sont destinés à générer des langages. Dans ces conditions, la modélisation de la partie opérative avec cet outil est considérée comme une énumération des situations que la partie opérative est susceptible d'atteindre et une association entre les évènements et les évolutions de la partie opérative.

a. Hypothèses de modélisation

Dans une telle modélisation, il n'est pas possible de faire intervenir le temps autrement que par l'ordre séquentiel des évolutions. Ce dernier point pose problème lorsque des évolutions simultanées de la partie opérative sont susceptibles de survenir. L'outil de modélisation n'est pas capable d'assimiler un tel comportement. Les travaux d'élaboration de la commande tels que la synthèse et la vérification utilise également l'automate à états finis pour décrire le comportement de la commande. L'interaction entre ces deux modèles est naturelle et se fait par synchronisation. Les délais entre les évolutions ne peuvent pas être pris en considération dans ce type de modélisation. Comme évoqués dans le chapitre précédent, les modèles représentés par ces outils sont souvent attachés au comportement fonctionnel du système. Le modèle de partie opérative ne prend alors pas en considération l'instrumentation présente sur le système. Les évènements sont associés à des évolutions perceptibles par observation du comportement mais le lien avec les signaux échangés entre les API et la partie opérative n'est pas établi. Nous proposons dans la suite du document, une approche de modélisation reposant sur l'analyse comportementale des constituants physiques de la partie opérative.

b. Contexte de la modélisation par les automates à états finis

La modélisation comportementale de la partie opérative par les automates à états finis nécessite de définir quelles sont les interactions entre les données disponibles et l'outil de

modélisation. Dans le cas de systèmes réels et en prenant un faible niveau d'abstraction, les données sont échangées entre la partie opérative et la partie commande au travers les entrées/sorties TOR. Cet ensemble de données booléennes sont associées avec les évènements des automates à états finis. Les évènements correspondant à des évolutions, il est nécessaire de prendre en considération non pas uniquement les signaux échangés mais leurs évolutions. Pour chaque signal booléen échangé, on associe deux événements correspondant aux changements d'un niveau du signal à l'autre niveau. Chaque signal est réalisé sur la partie opérative par une tension électrique associé à un seuil. Ainsi, le front montant décrit le passage du niveau bas du signal au niveau haut du seuil et le front descendant un passage du niveau haut au niveau bas. Le modèle de la partie opérative représente alors une représentation formelle des évolutions de ces signaux. Dans un fonctionnement normal du système, il est possible de prévoir comment les signaux issus des capteurs vont réagir face aux commandes qui ont été envoyées. Le fonctionnement normal du système suppose notamment que l'énergie permettant aux préactionneurs d'alimenter les actionneurs soit disponible, que le câblage du système et le positionnement des actionneurs et des capteurs soient corrects, qu'il n'y a pas de blocages ou de mouvements contraires exercés sur l'actionneur.

La Figure 3.4 illustre comment le comportement de la partie opérative peut être modélisé par un modèle à états. Les signaux échangés entre l'API et la partie opérative sont des signaux électriques dont la tension varie dans le temps pour échanger les données (Figure 3.4a). Ces signaux sont nommés « *Ei* » pour les capteurs connectés aux entrées de la partie commande et « *Si* » pour les commandes reliées aux sorties de la partie commande. Ces signaux sont continus et contiennent du bruit de fonctionnement. Les seuils de changement de niveau logique permettent de déterminer une donnée booléenne contenue dans chaque signal continu. L'ensemble de ces données booléennes peut être représenté sur un graphe temporel d'évolution au cours du temps (Figure 3.4b). Ce graphe ne peut pas représenter l'ensemble des évolutions susceptibles de se produire. Le comportement représenté sur ce graphe est un exemple de comportement susceptible de se produire sur le système en cours de fonctionnement. Cela signifie que pour une telle sollicitation de l'API, le comportement de la partie opérative est connu. Ce comportement peut également être représenté sous la forme d'un mot provenant d'un langage. Un événement représente un changement d'une variable booléenne. Les changements de valeur d'une variable booléenne sont représentés par une flèche (↑ ou ↓). La flèche montante ↑ représente le front montant d'une variable booléenne passant du niveau bas au niveau haut. La flèche descendante ↓ représente également le front descendant d'une variable booléenne passant du niveau haut au niveau bas. L'ordre d'occurrence de ces évènements les uns après les autres peut être pris en compte pour l'établissement du modèle. L'évolution continue du temps n'est pas conservée. Seul l'ordre d'occurrence des évènements est pris en compte, les délais entre occurrence d'évènements ne pouvant être pris en compte dans ce modèle. Cette séquence d'évènements peut être représentée par un mot (Figure 3.4c).

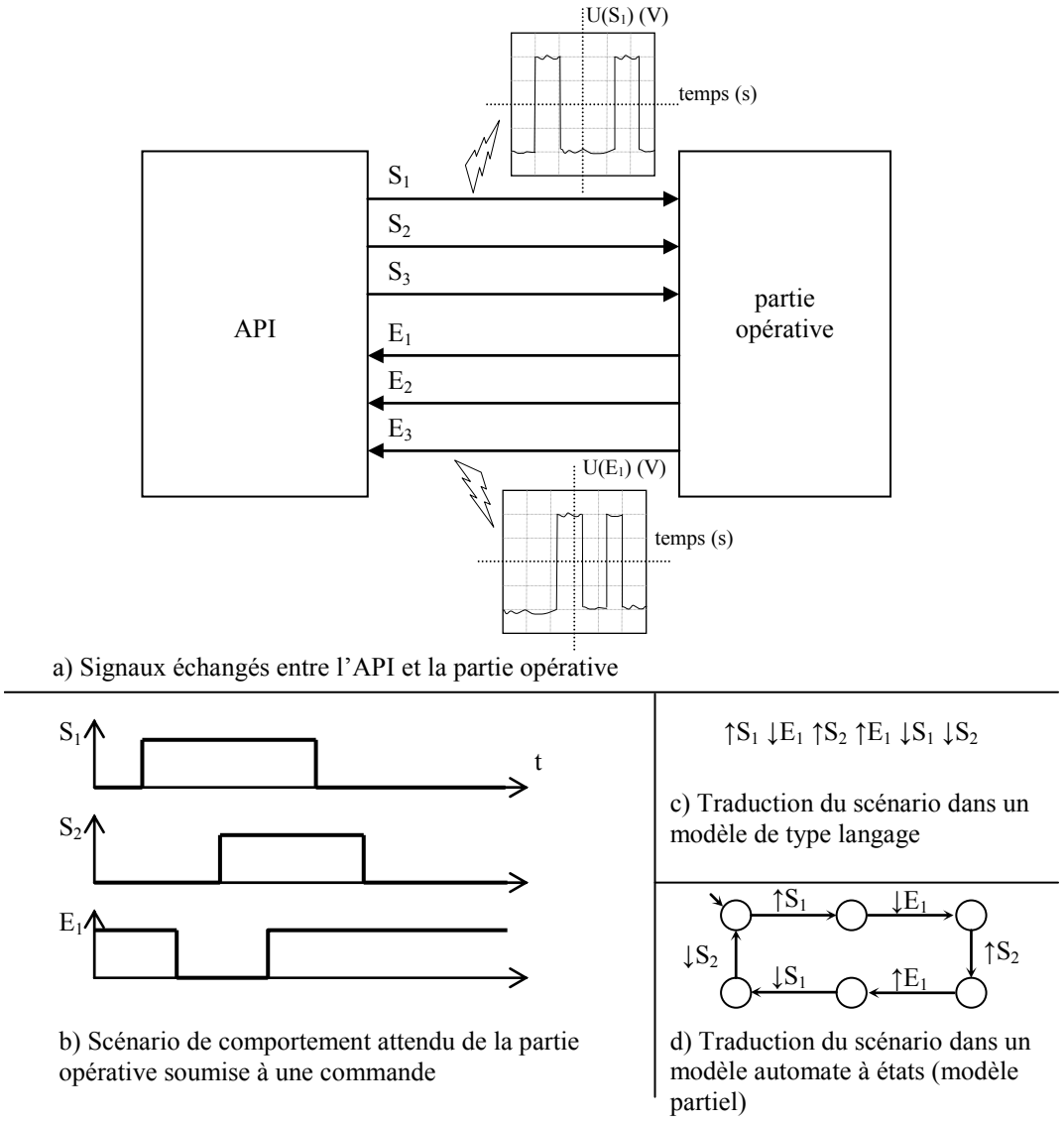


Figure 3.4 : modélisation comportementale de la partie opérative par un automate à états

Le modèle de partie opérative représente l'ensemble des séquences d'événements susceptibles de se produire sur le système. Les automates à états finis permettent de générer un langage décrivant l'ensemble des séquences d'évènements susceptibles de se produire. L'ensemble des évènements $\sum$ inclut tous les fronts des variables d'entrées/sorties de l'API ($\Sigma = \{\{Ei\} \cup \{Si\}\}$). La Figure 3.4d représente un automate à états finis $A = \{Q, \Sigma, \delta, q_0\}$ dans lequel le mot admissible par le système au cours de son fonctionnement, est admissible. L'ensemble d'états Q permet d'énumérer toutes les situations que le modèle est capable de distinguer. L'ensemble des transitions δ autorise l'occurrence des évènements acceptables depuis les états considérés. L'état q_0 permet de décrire l'état initial décrivant la situation de la partie opérative à la mise en fonctionnement. Ce modèle ne représente que le fonctionnement

partiel associé au scénario qui a été présenté sur la Figure 3.4. Ce modèle fait également l'hypothèse que la situation atteinte à la fin du scénario permet de le reproduire à l'infini. Par cette illustration, nous avons montré que la partie opérative était susceptible d'être représentée par un modèle sous la forme d'un automate à états finis. Des conditions doivent néanmoins être appliquées puisque ce modèle comporte des limitations notamment du point de vue de la gestion temporelle.

La représentation d'un système automatisé par les automates à états finis permet d'utiliser les résultats des travaux de recherche et les applications associées basés sur cet outil de modélisation. Dans ce contexte, nous proposons de modéliser la partie opérative de manière modulaire en fonction des constituants de la partie opérative. Il s'agit à terme de proposer un catalogue de constituants pouvant être mis en œuvre sur une partie opérative.

La distinction des chaînes d'action et d'acquisition de la partie opérative peut être reprise dans la démarche de modélisation de la partie opérative en exploitant la construction de la partie opérative réelle à partir de composants mécaniques standards. Les composants technologiques que sont les préactionneurs, les actionneurs et les capteurs peuvent ainsi être modélisés individuellement avant d'être assemblés pour réaliser méthodiquement le modèle de la partie opérative. La démarche de modélisation proposée consiste donc à modéliser la partie opérative par une décomposition en chaînes fonctionnelles.

3. Démarche de modélisation des chaînes fonctionnelles par les automates à états finis

Nous avons vu que la chaîne fonctionnelle était généralement composée d'une chaîne d'action et d'une chaîne d'acquisition. La modélisation de la chaîne d'action a alors pour but de déterminer l'action qui sera la conséquence des signaux de commande auxquels la partie opérative est soumise au sein de la chaîne fonctionnelle. La partie opérative réagit à ces signaux de commande par une action dont les effets sont mesurés par la chaîne d'acquisition. La modélisation de la chaîne d'acquisition consiste alors à déterminer le comportement des capteurs positionnés sur la partie opérative lorsque celle-ci est soumise à l'action.

Les chaînes d'action et d'acquisition sont conçues, pour la partie opérative réelle, avec des composants standards. La modélisation reprend cette structuration en adoptant une modélisation des composants standards dans des modèles de comportement (Figure 3.5). L'assemblage de ces deux modèles permet d'obtenir le modèle de partie opérative de la chaine fonctionnelle décrivant le comportement possible des capteurs en fonction des commandes.

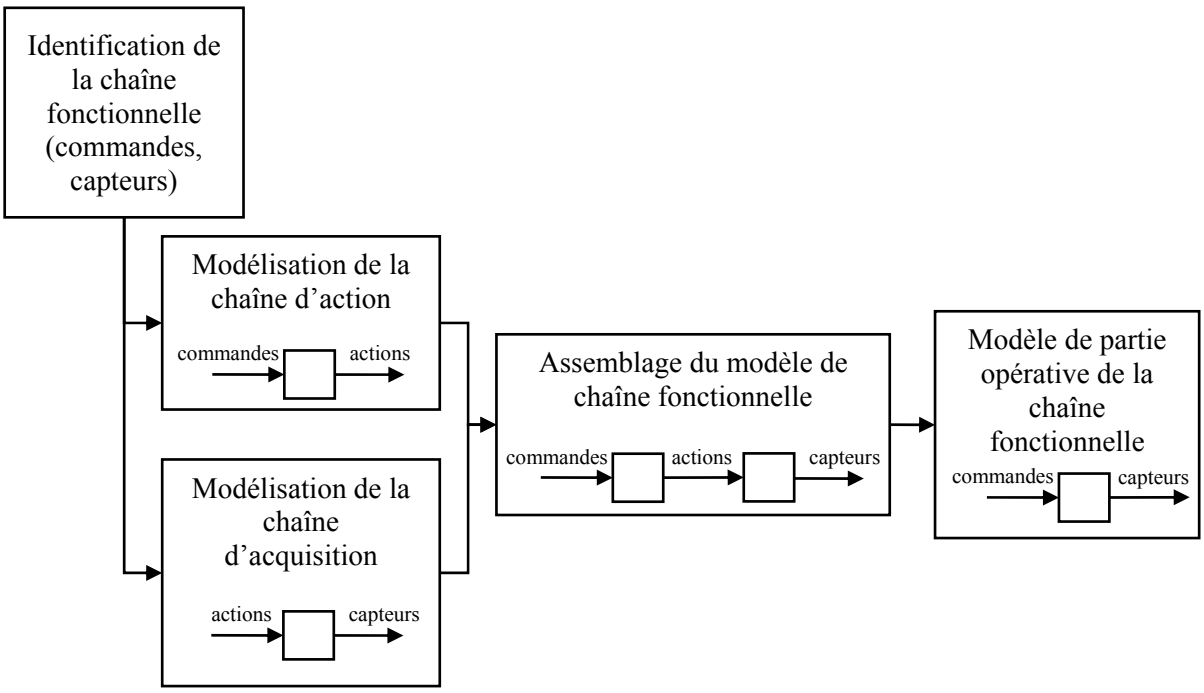


Figure 3.5 : étapes de la démarche de modélisation d'une chaîne fonctionnelle

a. Modélisation de la chaîne d'action

D'un point de vue technologique, la chaîne d'action est réalisée par le préactionneur. Les préactionneurs fournissent l'énergie à l'actionneur en fonction des signaux de commande reçus. Pour modéliser ce type de constituant, on peut associer des modes de distribution d'énergie à toutes les combinaisons possibles des signaux de commande. La technologie de pilotage permettant de générer un mode de distribution d'énergie, a une forte incidence sur le modèle du préactionneur.

Dans le cas d'un préactionneur monostable, un mode de fourniture d'énergie est associé à chaque état pris par les signaux de commande. On obtient alors deux modes de fourniture d'énergie. Le modèle de comportement obtenu est représenté sur la Figure 3.6 avec l'hypothèse que dans l'état initial, la commande est au niveau bas.

Le temps n'étant pas pris en compte, le temps de commutation du préactionneur est supposé nul. Les évolutions intermédiaires entre le passage d'un mode à l'autre ne peuvent pas être prises en compte dans la modélisation.

Le modèle de comportement du préactionneur suppose que le préactionneur soit alimenté en énergie. Certains préactionneurs nécessitent d'avoir une alimentation en énergie de puissance destinée à alimenter l'actionneur et une alimentation en pilotage pour que la commutation entre les différents modes de distribution dans le fonctionnement interne du préactionneur soit assurée. Les électro-distributeurs pneumatiques à commande indirecte par exemple, sont composés de plusieurs étages.

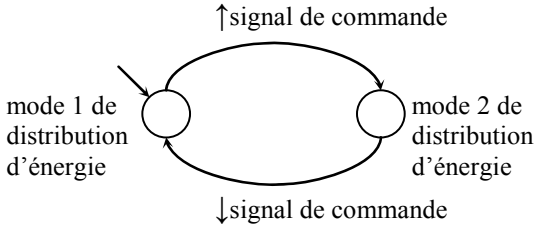


Figure 3.6 : modélisation d'un préactionneur avec une commande monostable

Les préactionneurs ayant une commande monostable peuvent utiliser le modèle de la Figure 3.6 Les relais électriques monostables, les distributeurs pneumatiques ou hydrauliques monostables… sont des préactionneurs pouvant être ainsi représentés. Cependant ce modèle s'avère très simplifié par rapport à la variété de préactionneurs existants dans les catalogues des constructeurs. En effet, les paramètres sur les délais de commutation, le dimensionnement des équipements en fonction des puissances mises en jeu, les paramètres liés aux informations de commande… ne sont pas considérés dans cette proposition de modélisation. Toutefois, ces hypothèses permettent de se placer à un niveau de modélisation suffisamment faible pour que l'applicabilité des résultats soit réalisable moyennant ces hypothèses mais également suffisamment élevé pour utiliser les outils de modélisation classiquement utilisés que sont par exemple les automates à états finis.

Dans le cas d'un préactionneur bistable, deux commandes permettent de piloter le préactionneur. Le préactionneur bistable peut avoir un nombre plus important plus de modes de distribution d'énergie que le préactionneur monostable car les combinaisons réalisables avec les commandes sont plus nombreuses. Un mode de distribution d'énergie ne peut pas toujours être associé à chaque combinaison des commandes : certains préactionneurs possèdent une fonction de mémorisation similaire aux fonctions bascules en logique câblée. On peut donc associer une fonctionnalité pour chacune des combinaisons possibles des commandes mais chaque fonctionnalité peut être un mode de distribution d'énergie mais aussi la mémorisation du mode de distribution d'énergie. Enfin, certains préactionneurs ont un comportement non fonctionnel lorsqu'ils sont soumis à une combinaison des commandes. Ces combinaisons ne sont pas utilisées dans le fonctionnement normal et cohérent du système. Ces combinaisons peuvent néanmoins être prises en considération pour certaines applications de validation de la commande ou de diagnostic par exemple. En général, les commandes utilisées sur les préactionneurs bistables ont des fonctions opposées sur l'actionneur qui va être alimenté en énergie. Certains sont équipés de système de verrouillage mécanique pour inhiber les commandes contradictoires. Par exemple, les moteurs asynchrones triphasés pouvant tourner dans les deux sens de rotation sont pilotés par des contacteurs électriques qui ne doivent pas être simultanément fermés pour sous peine de provoquer des courts-circuits

D'un point de vue technologique, les préactionneurs utilisant une commande bistable sont nombreux et de multiples variantes existent en fonction des modes de distribution d'énergie. Les préactionneurs bistables peuvent être mis en œuvre avec des relais électriques, des

distributeurs pneumatiques ou hydrauliques. L'accouplement de deux préactionneurs monostables pour alimenter un même actionneur permet également de réaliser la fonction de préactionneur bistable. Les préactionneurs bistables sont utilisés pour fournir l'énergie à des moteurs à deux sens de rotation et aux vérins pneumatiques ou électriques comportant une position intermédiaire sur leur course nominale. Tous les actionneurs devant avoir trois modes de distribution d'énergie sont pilotés par des préactionneurs bistables. Certains actionneurs comportant deux modes de distribution d'énergie utilisent des préactionneurs bistables. Les raisons de ce choix dépendent principalement des habitudes du concepteur du système automatisé. Des exemples de modèles de préactionneurs bistables sous la forme d'automates à états finis seront détaillés dans la suite dans les modèles de chaînes fonctionnelles standards.

b. Modélisation de la chaîne d'acquisition

La chaine d'acquisition est technologiquement réalisée par l'actionneur et le positionnement des capteurs sur celui-ci. Les actionneurs évoluent en fonction de la distribution d'énergie réalisée en amont par les préactionneurs. Les actions et leurs effets sont mesurés au travers des détecteurs disposés sur la partie opérative. La modélisation des actionneurs et des détecteurs consiste à exprimer formellement le comportement des détecteurs lors du fonctionnement du système. Dans le cadre d'un fonctionnement normal, certaines évolutions peuvent être prévues à l'avance. Des évolutions imprévues permettent ainsi de détecter un fonctionnement incohérent du système. Le modèle de partie opérative décrit donc le comportement du système dans son fonctionnement normal. Par exemple, dans le cas d'un vérin doté de détecteurs en fin de course, le passage d'une extrémité à l'autre de la course du piston entraîne une situation intermédiaire dans laquelle les deux détecteurs sont au niveau bas. Le modèle de partie opérative réfutera la configuration telle que les deux détecteurs soient au niveau haut au même instant car il ne comporte pas de situation similaire.

Une fois le modèle du préactionneur et le modèle d'actionneur perçu au travers des capteurs, ces modèles sont assemblés pour forme le modèle de la chaîne fonctionnelle (Figure 3.5).

c. Association entre le modèle de préactionneurs et le modèle d'actionneur

Pour modéliser le comportement de l'actionneur et des détecteurs, on peut s'intéresser à l'action qui est en train de se produire. L'action n'est généralement pas directement mesurable sur le système. En effet, un mouvement de déplacement par exemple n'est pas souvent détecté par un capteur. Il est cependant possible, en posant les bonnes hypothèses, de supposer la tendance de l'action générée par l'alimentation en énergie (Figure 3.6). Le préactionneur alimente l'actionneur en énergie pour lui permettre de générer une action. Si on suppose que l'alimentation en énergie est suffisante pour que l'action soit générée, alors le modèle de préactionneur qui permet de déterminer le mode d'alimentation en énergie de l'actionneur détermine également l'action qui est générée. Dans le cas de vérins en cours de déplacement,

le mouvement peut supposer être connu à partir du mode d'alimentation en énergie. Pour cela, il est nécessaire de supposer que la connaissance de la configuration des commandes est suffisante pour savoir quel mouvement est effectué. L'action n'est cependant pas instantanément lancée lors des changements de commande dans la réalité. Il faut nécessairement supposer que le temps de commutation du préactionneur soit nul et qu'il n'y a pas d'effets d'inertie de l'actionneur entre les passages d'un mode d'alimentation en énergie à un autre. Avec cette hypothèse, les évolutions des détecteurs mesurant la course du vérin dépendent des commandes qui sont envoyées et l'association peut être réalisée. Ainsi, pour un vérin dont la course est mesurée par deux détecteurs de fin de course, la connaissance du mouvement permet de supposer quel sera le prochain changement de niveau des détecteurs.

Si l'hypothèse ne peut pas être appliquée pour des raisons telles qu'une inertie trop importante de l'actionneur par rapport aux temps couramment perçu sur le système entre les évolutions de détecteurs, alors le modèle de partie opérative n'est pas en mesure de corréler les modes d'alimentation de l'actionneur avec la restriction des changements de niveau des détecteurs.

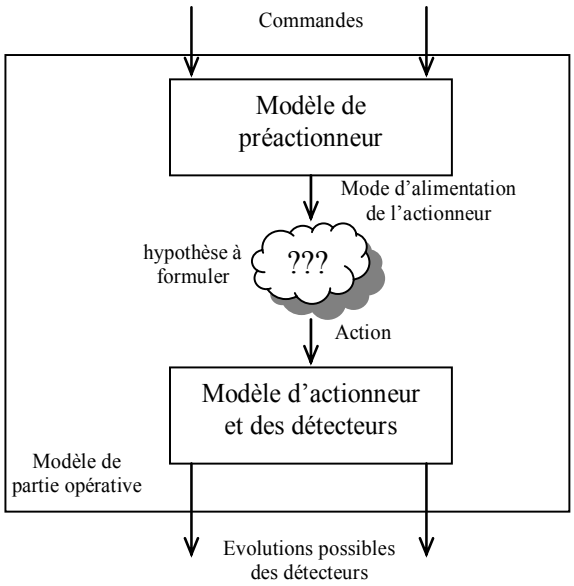


Figure 3.7 : structuration du modèle de partie opérative par automates à états finis

d. Bilan de la démarche de modélisation retenue

La démarche de modélisation présentée permet d'obtenir le modèle de partie opérative d'une chaîne fonctionnelle à partir de la constitution de la partie opérative réelle. Pour que la démarche de modélisation s'applique, certaines hypothèses sont nécessaires. Les constantes de temps mises en jeu dans la partie opérative réelle ne doivent pas être trop faibles vis-à-vis du temps de cycle de la commande pour que les contraintes de temps réel soient respectées. Le fonctionnement de la partie opérative suppose qu'aucune action extérieure n'interfère avec

l'action générée par chaque chaine fonctionnelle. L'absence d'alimentation en énergie de la partie opérative n'est pas prise en compte dans la modélisation.

Selon les paramètres physiques du préactionneur et de l'actionneur, l'action déterminée par la modélisation du préactionneur n'est pas toujours exploitable pour restreindre le comportement de l'actionneur. Si l'hypothèse de corrélation ne peut pas être formulée, le modèle d'actionneur doit pouvoir prendre en compte toutes les évolutions logiques possibles des capteurs quelque soit les signaux de commande.

Pour illustrer cette démarche de modélisation, le paragraphe suivant propose cinq exemples de modèles de chaînes fonctionnelles utilisant différentes technologies.

4.Exemple de modèles de partie opérative décrit par les automates à états finis

Suivant les matériels utilisés pour former une chaîne fonctionnelle, le modèle de partie opérative sera différent en prenant en compte les spécificités de la partie opérative. Cinq chaînes fonctionnelles sont présentées et modélisées avec notre démarche.

a. Modélisation du vérin avec un distributeur monostable et un détecteur de fin de course

La première modélisation d'une chaîne fonctionnelle est le système constitué d'un vérin pneumatique alimenté par un distributeur monostable et équipé d'un détecteur de fin de course (Figure 3.8). L'actionneur est représenté par le vérin, le préactionneur par le distributeur et les détecteurs par le capteur TOR, « *a* ». La commande « *SORTIR* » permet de piloter la chaîne fonctionnelle.

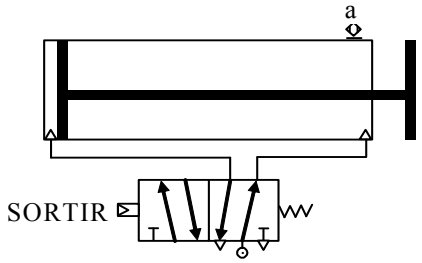


Figure 3.8 : chaîne fonctionnelle « vérin pneumatique avec distributeur monostable et un détecteur fin de course »

Du point de vue du comportement de la chaîne fonctionnelle, l'utilisation d'un vérin simple effet associé à un distributeur 3/2 et l'utilisation d'un vérin double effet et d'un distributeur

5/2 n'a pas d'incidence sur le résultat de la modélisation. Il s'agit de deux manières différentes de réaliser physiquement les modes d'alimentation en énergie. Des modifications de comportement sont observées dans le cas de fonctionnement non nominal notamment en cas d'absence d'alimentation en énergie.

Le distributeur utilisé pour effectuer l'alimentation du vérin comporte deux modes d'alimentation en air comprimé. Ces modes d'alimentation sont également appelés états du distributeur (Figure 3.9). L'état 1 correspond à une alimentation du vérin en énergie pneumatique en vue de rentrer la tige dans le corps. L'état 2 correspond à une alimentation du vérin en énergie pneumatique en vue de sortir la tige du corps du vérin.

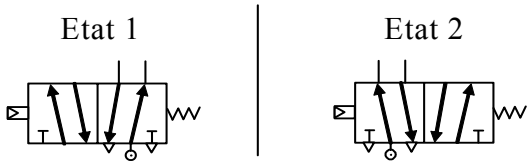


Figure 3.9 : positions possibles du tiroir du distributeur

Modéliser le comportement du préactionneur revient à établir l'état du distributeur en fonction des commandes. Bien que très simple dans ce cas de figure, on peut réaliser une table de vérité suivante (Figure 3.10) pour déterminer quel est l'état pris par le distributeur en fonction de l'état de la commande « *SORTIR* ».

SORTIR	Etat du distributeur
0	Etat 1
1	Etat 2

Figure 3.10 : table de vérité du distributeur monostable en fonction de la commande

A partir de cette table de vérité, il est possible de construire un automate à états décrivant tous les cas de figure différents des entrées et du préactionneur. L'état initial est fixé par la commande émise à la mise en marche du système. Le front montant de la commande « *SORTIR* » génère un changement d'état. Le préactionneur change par la même occasion de mode d'alimentation en énergie. Un front descendant sur la commande permet de revenir dans l'état initial et de reprendre le mode d'alimentation associé.

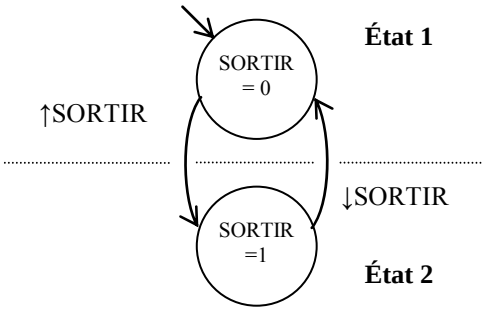


Figure 3.11 : modèle du distributeur monostable

Pour déterminer le modèle d'actionneur et des capteurs, il faut tout d'abord s'intéresser aux informations qui vont être délivrées par ces capteurs. En effet, les capteurs disposés sur la partie opérative du système permettent de donner une vision partielle de l'actionneur et de la situation courante. Pour modéliser les évolutions des capteurs lors du fonctionnement, il faut préalablement s'intéresser aux comportements attendus des capteurs suivant les actions exercées par l'actionneur. Il faut lister toutes les situations distinguables possibles avec les capteurs : il ne suffit pas seulement de lister les configurations possibles d'états de capteurs mais il faut tenir compte des évolutions. Des situations ayant la même configuration d'états de capteurs peuvent néanmoins être distinguées si l'historique des évolutions des capteurs peut permettre de les distinguer.

Dans l'exemple de la Figure 3.8, le capteur est utilisé pour tester si la tige du vérin est en position sortie. Le capteur est positionné de façon à ce qu'il passe au niveau haut lorsque le piston du vérin arrive à l'extrémité sortie. Lorsque la tige du vérin n'est plus située à l'extrémité de la course du vérin, le capteur retrouve au niveau bas. Ce comportement peut être représenté dans un tableau (Figure 3.12).

situation	**niveau du capteur**
sortie	a=1
non sortie	a=0

Figure 3.12 : situations distinguables du vérin équipé d'un capteur

Les évolutions susceptibles de se produire lorsqu'une action a lieu sur l'actionneur sont présentées sur la Figure 3.13.

Action de sortie du vérin :
non sorti → sorti

Action de rentrée du vérin :
sorti → non sorti

Figure 3.13 : évolutions des situations lors des actions du vérin

Ces évolutions possibles de situations peuvent être représentées par les évolutions possibles des états de capteurs sous une structure proche d'un automate à états finis (Figure 3.14)

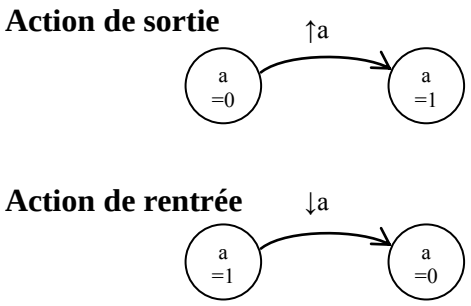


Figure 3.14 : évolution des capteurs en fonction des actions

En fonction des hypothèses retenues, les états du préactionneur peuvent être corrélés avec les actions du modèle d'actionneur/capteur. En supposant que cette corrélation peut être effectuée parce que les inerties lors des déplacements de la tige du vérin sont peu importantes et que les temps de commutation du distributeur sont négligeables vis-à-vis des temps de réponse de la partie opérative, le modèle de partie opérative de la chaîne fonctionnelle est construit en agrémentant le modèle de préactionneurs des séquences d'évolution de capteurs.

Nous précisons maintenant **l'hypothèse de corrélation entre le modèle d'actionneur et le modèle de capteurs** : si la durée entre les évolutions des commandes et l'action physique générée par ces évolutions est négligeable par rapport aux constantes de temps mises en œuvre sur le procédé de fabrication, alors l'action évaluée à l'aide des commandes peut être exploitée pour connaître les évolutions possibles des capteurs.

La construction du modèle nécessite de connaître la configuration des capteurs à l'initialisation du système pour qu'il y ait adéquation entre l'état réel de la partie opérative et du modèle. Le modèle de partie opérative ne représentant que les évolutions du système, il faut utiliser les informations contenues dans les états découlant des évènements pour connaître quel est l'état de la séquence d'évolution initiale.

Dans le cas du modèle de la chaîne fonctionnelle, on admet que le vérin est rentré à l'initialisation. Le modèle obtenu est présenté sur la Figure 3.15.

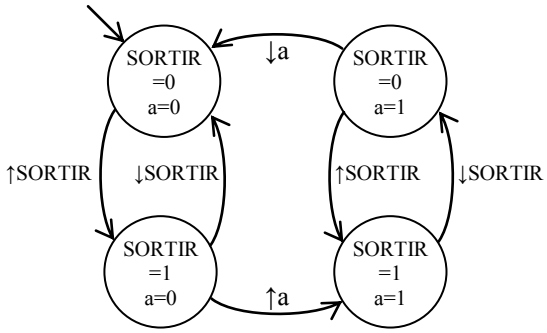


Figure 3.15 : modèle de partie opérative de la chaîne fonctionnelle avec l'hypothèse de corrélation entre l'état du préactionneur et l'action

Le modèle comporte des annotations dans les états décrivant les niveaux des capteurs et des commandes. Ces annotations peuvent être retirées pour une utilisation mathématique des automates à états finis. Cela permet, entre autre, d'exploiter les résultats de l'ensemble des travaux de recherche sur les SED avec ce modèle.

Cependant, l'hypothèse de corrélation entre le modèle d'actionneur et le modèle des capteurs n'est pas toujours justifiable. Les inerties de l'actionneur ou les temps de commutation du préactionneur induisent des délais entre les changements de niveaux de la commande et la réaction de la partie opérative. Si tous ces paramètres ne permettent pas de retenir l'hypothèse, il n'est pas possible d'utiliser les modes d'alimentation de l'actionneur pour restreindre les évolutions des capteurs. Le modèle de partie opérative de la chaîne fonctionnelle comporte alors toutes les évolutions possibles des capteurs et distingue les modes d'alimentation entre les états. Il n'y a par contre, aucun recoupement effectué entre les modes d'alimentation et les évolutions possibles des capteurs. Les chaînes d'action ne peuvent pas être différenciées dans le modèle de partie opérative. Il y a donc fusion des chaînes d'action dans un modèle de actionneur/capteurs unique (Figure 3.16).

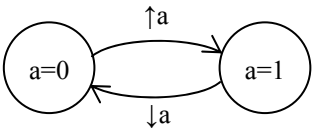


Figure 3.16 : modèle d'actionneur/capteurs sans l'hypothèse

Le modèle de la chaîne fonctionnelle n'utilisant pas l'hypothèse de corrélation entre le modèle d'actionneur et le modèle de capteurs, il est réalisé par une composition du modèle de préactionneur et du modèle d'actionneur/capteurs. Ce modèle comporte la même structure que le modèle prenant en compte l'hypothèse mais des transitions sont ajoutées au modèle. Il s'agit des transitions associées à des évolutions de capteurs allant à contre courant des niveaux de commande envoyés par la partie commande (Figure 3.17). Le modèle comporte toutes les évolutions possibles du capteur « *a* » quelque soit l'état des commandes : lorsque le

capteur est au niveau haut, le front descendant du capteur est pris en compte et lorsque le capteur est au niveau bas, le front montant est également considéré.

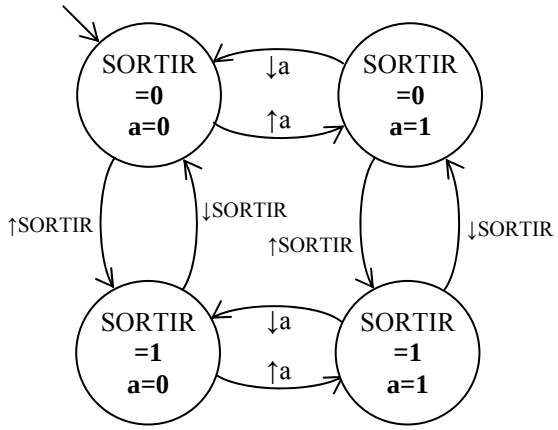


Figure 3.17 : modèle de partie opérative de la chaîne fonctionnelle sans hypothèse

Ce modèle de partie opérative prend ainsi en compte toutes les évolutions possibles des commandes et des capteurs. Les seules restrictions que comporte ce modèle, concernent la non-simultanéité des évolutions et l'alternance entre les fronts montants et descendants. Tous les changements de niveaux de capteur et de commande sont tolérés indépendamment de l'ordre dans lequel ils interviennent. Sans hypothèse, le modèle de partie opérative ne restreint donc pas le comportement du système vis-à-vis de l'ensemble des évolutions possibles. Ce comportement n'est toutefois pas généralisable et les modèles présentés dans les figures suivantes vont illustrer ce propos. Les annotations dans les états peuvent également être retirées pour les applications futures.

Ce premier exemple de chaîne fonctionnelle illustre la technique de modélisation de la partie opérative à l'aide de l'outil que sont les automates à états finis. D'autres chaînes fonctionnelles sont présentées à la suite pour souligner les atouts mais aussi les limites des modèles obtenus.

b. Modélisation du vérin avec un distributeur bistable et deux détecteurs

La chaîne fonctionnelle présentée dans ce paragraphe est composée d'un vérin pneumatique alimenté par un distributeur à deux positions bistables et de deux détecteurs de fin de course. En reprenant la démarche de modélisation précédemment évoquée, chaque composant physique est associé à un élément de la chaîne fonctionnelle. L'actionneur est représenté par le vérin pneumatique double effet, le préactionneur par le distributeur bistable et les détecteurs par les capteurs TOR « *a* » et « *b* ». Les commandes « *RENTRER* » et « *SORTIR* » permettent de piloter cette chaîne fonctionnelle.

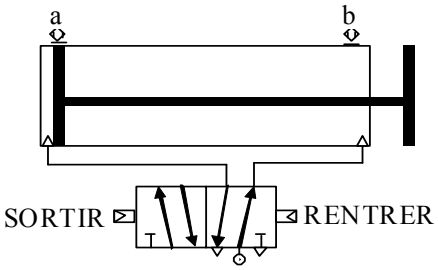


Figure 3.18 : chaîne fonctionnelle « vérin pneumatique avec distributeur bistable et deux capteurs fin de course »

Le distributeur utilisé pour effectuer l'alimentation du vérin comporte deux modes d'alimentation en air comprimé en fonction de la configuration des commandes (Figure 3.19). L'état 1 correspond à une alimentation du vérin en énergie pneumatique en vue de rentrer la tige dans le corps. L'état 2 correspond à une alimentation du vérin en énergie pneumatique en vue de sortir la tige dans le corps.

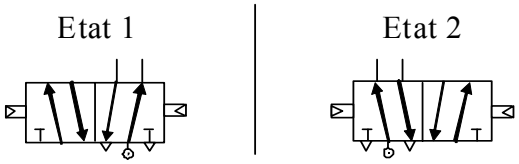


Figure 3.19 : positions possibles du tiroir du distributeur

La modélisation du comportement de ce préactionneur peut être assimilée à la définition d'une loi de comportement de l'état du distributeur en fonction des commandes. Dans le cas de ce préactionneur, la table de vérité facilite la représentation. Elle est construite en énumérant l'état occupé par le préactionneur en fonction de toutes les configurations pouvant être prises par les commandes (Figure 3.20). Les cas de figure dans lesquels une seule des commandes est à son niveau haut, sont simples à déterminer. On admet que le distributeur prend l'état que le niveau haut de la commande lui demande de prendre. En revanche, les situations dans lesquelles les commandes sont au niveau bas, sont plus difficiles à prendre en compte. Les distributeurs bistables classiques tel que celui présenté ici, permettent au distributeur de conserver l'état précédent avant que la dernière commande n'atteigne le niveau bas. Lorsque les deux commandes sont simultanément au niveau haut, il y a contradiction dans le pilotage de la chaîne fonctionnelle.

RENTRER	SORTIR	Etat du distributeur
0	0	Etat précédent
0	1	Etat 2
1	0	Etat 1
1	1	Etat précédent

Figure 3.20 : table de vérité du distributeur bistable en fonction des commandes

Cette table de vérité facilite la réalisation du modèle de comportement du distributeur par un automate à états finis. Le lien qui a été proposé dans la table de vérité de cet exemple peut être vérifié en étudiant les documentations techniques émanant des fabricants ou en réalisant des tests sur les matériels à proprement parler.

Pour construire le modèle sous la forme d'un automate à états finis, il faut tout d'abord énumérer toutes les configurations possibles des commandes et de l'état du distributeur. Six configurations sont possibles : lorsque les commandes sont simultanément au niveau bas ou au niveau haut, le préactionneur peut prendre l'un des deux états. Lorsqu'une seule des commandes est au niveau haut, le préactionneur est dans l'état fixé par la commande qui est au niveau haut.

Ces six configurations permettent de créer six états différents. Les transitions permettent de représenter les évolutions des commandes. Depuis les états dans lesquels les commandes sont simultanément au niveau bas ou haut, l'évolution d'une commande permet d'atteindre l'état unique auquel correspondent les niveaux de commande. Depuis un des états dans lequel une seule commande est au niveau haut, plusieurs états sont susceptibles d'accepter une évolution dans laquelle les commandes sont au même niveau. Le choix de l'état de destination est réalisé à l'aide de l'état du préactionneur qui doit être conservé lorsque les commandes sont au même niveau.

L'état choisi pour initialiser le modèle en fonction des commandes émises à la mise en marche du système correspond au fait que les commandes soient au niveau bas. L'état initial du préactionneur est supposé connu et être dans le mode d'alimentation 1 (Figure 3.21).

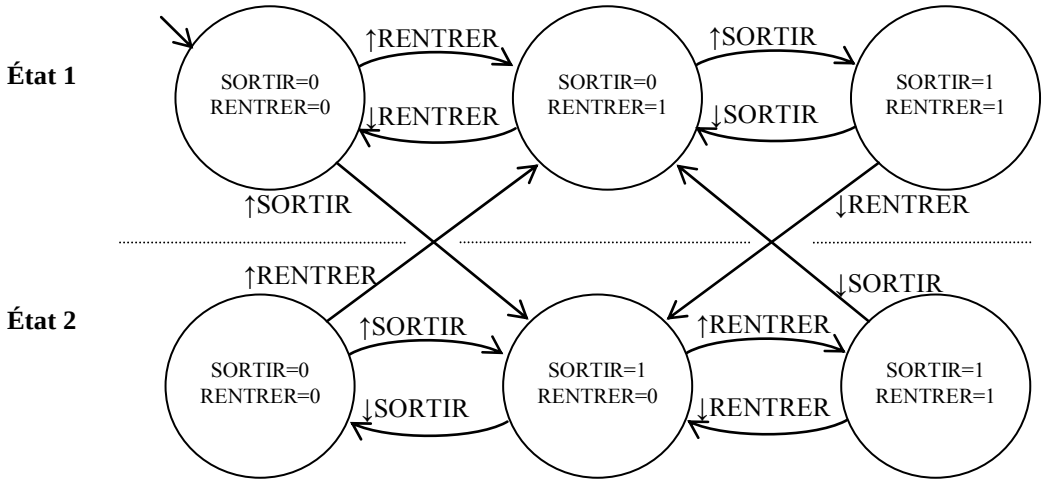


Figure 3.21 : modèle de distributeur bistable

Pour modéliser le comportement de l'actionneur et des capteurs, on détermine quelles sont les situations de l'actionneur pouvant être distinguées. Les deux capteurs TOR disposés sur la course de l'actionneur permettent d'identifier les situations où la tige du vérin est totalement rentrée, sortie et en position intermédiaire. Les situations sont dites distinguables si les capteurs disposés sur l'actionneur permettent par un changement de niveau de distinguer où se situe l'actionneur. Chacune de ces situations sont représentées dans un tableau avec le niveau des capteurs TOR (Figure 3.22). Pour chacune de ces situations, il n'existe qu'une seule configuration possible des niveaux de capteurs. Ainsi, pour la situation rentrée, seul le capteur « *a* » est au niveau haut. Pour la situation intermédiaire, en analysant le positionnement des capteurs, les capteurs sont simultanément au niveau bas lors du passage par la situation intermédiaire. Pour la situation sortie, seul le capteur « *b* » est au niveau haut.

situation	**niveau du capteur**
rentrée	a=1, b=0
intermédiaire	a=0, b=0
sortie	a=0, b=1

Figure 3.22 : situations distinguables du vérin équipé de deux capteurs de fin de course

Les évolutions de l'actionneur d'une situation à une autre suivent une logique en fonction de l'action se produisant sur le système (Figure 3.23). L'action de sortie de la tige du vérin engendre une séquence d'évolutions possible sur le système. L'action de sortie de la tige du vérin permet de passer de la situation rentrée puis intermédiaire à sortie. Dans le cas de l'action de rentrée de la tige du vérin, l'actionneur passe successivement de la situation sortie puis la situation intermédiaire et la situation rentrée.

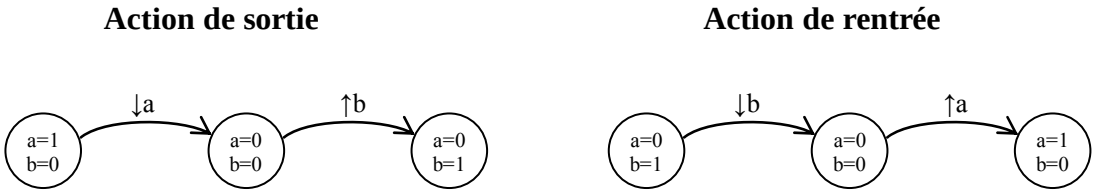


Figure 3.23 : évolution des capteurs en fonction des actions

De manière identique à la chaîne fonctionnelle présentée auparavant, les modes d'alimentation en énergie déterminés dans le modèle de préactionneur peuvent être corrélés avec les actions du modèle d'actionneur/capteurs sous des hypothèses identiques. En supposant négligeables les inerties de l'actionneur et le temps de commutation du distributeur entre les différents modes d'alimentation en énergie vis-à-vis des constantes de temps mises en jeu dans le fonctionnement de l'actionneur, on peut être amené à supposer que les capteurs ne vont pas suivre une évolution selon l'action générée par l'état du préactionneur. La construction du modèle est donc réalisée en n'acceptant les évolutions des capteurs uniquement si le modèle d'actionneur/capteurs tolère l'évolution en fonction du mode d'alimentation courant du préactionneur.

La Figure 3.24 donne une représentation graphique du modèle de partie opérative de cette chaîne fonctionnelle. Ce modèle combine le modèle de préactionneur, décrivant les modes d'alimentation de l'actionneur en fonction des niveaux des commandes et le modèle d'actionneur/capteurs décrivant les évolutions possibles des capteurs en fonction de l'action se produisant dans chacun des états du modèle. Chaque état du modèle comporte ainsi les niveaux des commandes et des capteurs. Le mode d'alimentation de l'actionneur est implicitement connu pour chacun des états du modèle à partir du modèle de préactionneur. En effet, le modèle de partie opérative est construit en dupliquant trois fois le modèle de préactionneur, une fois pour chaque situation de l'actionneur.

L'état initial est choisi en fonction des niveaux des commandes et des capteurs lors de la mise en service du système. Ce choix est fait arbitrairement ici, à partir des considérations liées au fonctionnement du système au cours de son utilisation. Les évolutions des capteurs tiennent compte de l'hypothèse associant le mode d'alimentation courant de l'actionneur avec l'action suivi par l'actionneur.

Le comportement transcrit par le modèle laisse apparaître la mémorisation de l'action suivie par l'actionneur lorsque les deux commandes sont au même niveau. Le modèle prend en compte les commandes contradictoires pouvant avoir lieu même si cette situation ne fait pas partie d'un usage normal de la partie opérative.

Le comportement des capteurs est limité au comportement normal de fonctionnement. Les évolutions ne sont pas toutes présentes dans le modèle car certaines évolutions de capteurs sont susceptibles de se produire en cas de défaillance du système automatisé ou si les limites du modèle sont dépassées. Une configuration dans laquelle les capteurs « *a* » et « *b* » sont en même temps au niveau haut signifie qu'une évolution imprévue a eu lieu sur le système automatisé. Ce type d'information peut être exploité dans les applications de diagnostic.

D’autres évolutions telles que les changements simultanés de capteurs (activation d’un capteur de fin de course en même temps que la désactivation du capteur situé à l’autre extrémité) signale que la limite du domaine de validité du modèle est atteinte dans le contexte dans lequel il est utilisé.

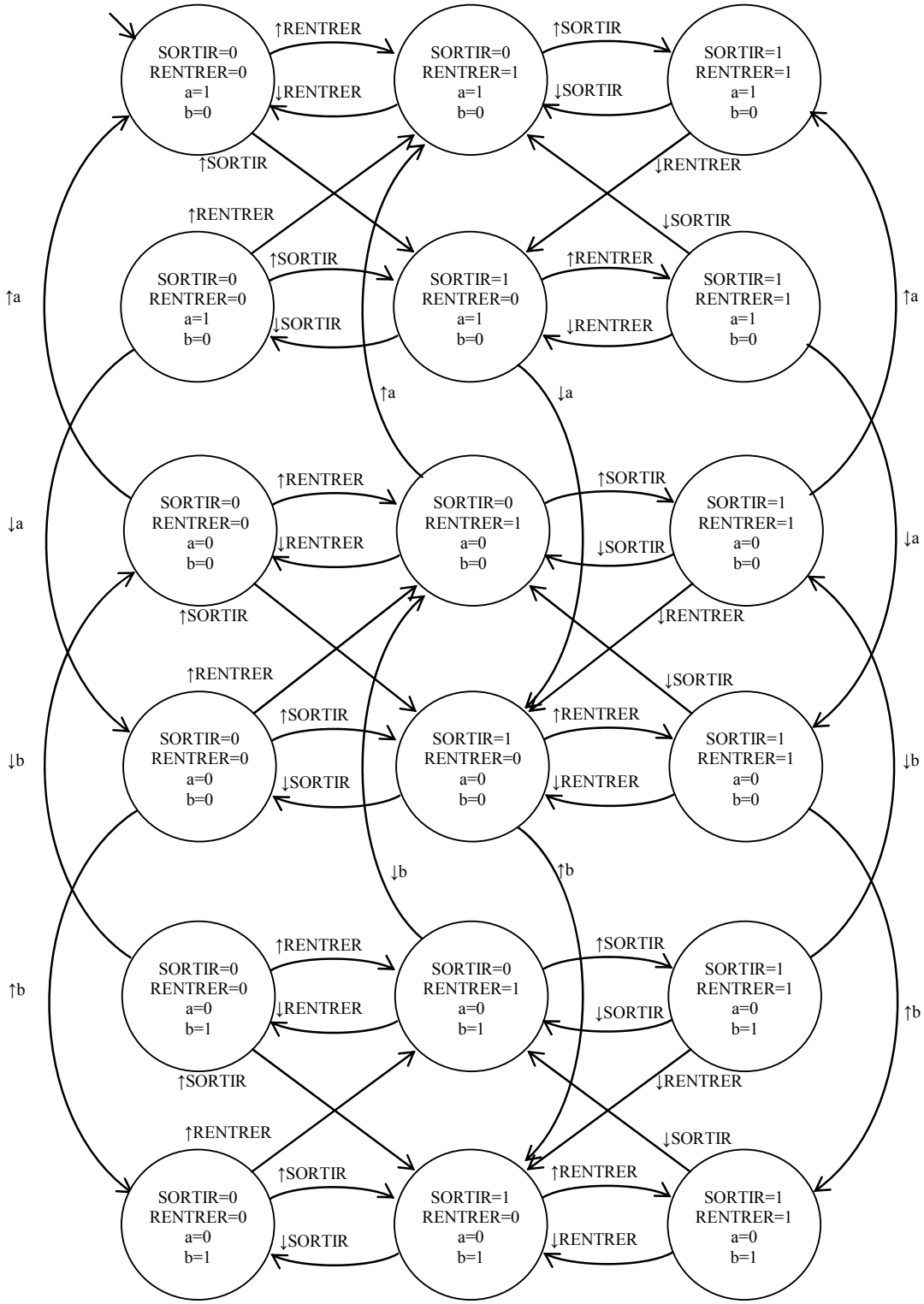


Figure 3.24 : modèle de partie opérative de la chaîne fonctionnelle

La modélisation peut également ne pas prendre en considération l'hypothèse de corrélation entre le mode d'alimentation déterminé par les commandes et l'action se produisant sur l'actionneur. Dans ce cas de figure, il n'y a plus aucune limitation des évolutions des capteurs en fonction des commandes. Le modèle de partie opérative conserve la même structure que le modèle de partie opérative de la chaîne fonctionnelle mais comporte également les évolutions des capteurs susceptibles d'intervenir dans une des actions (Figure 3.25).

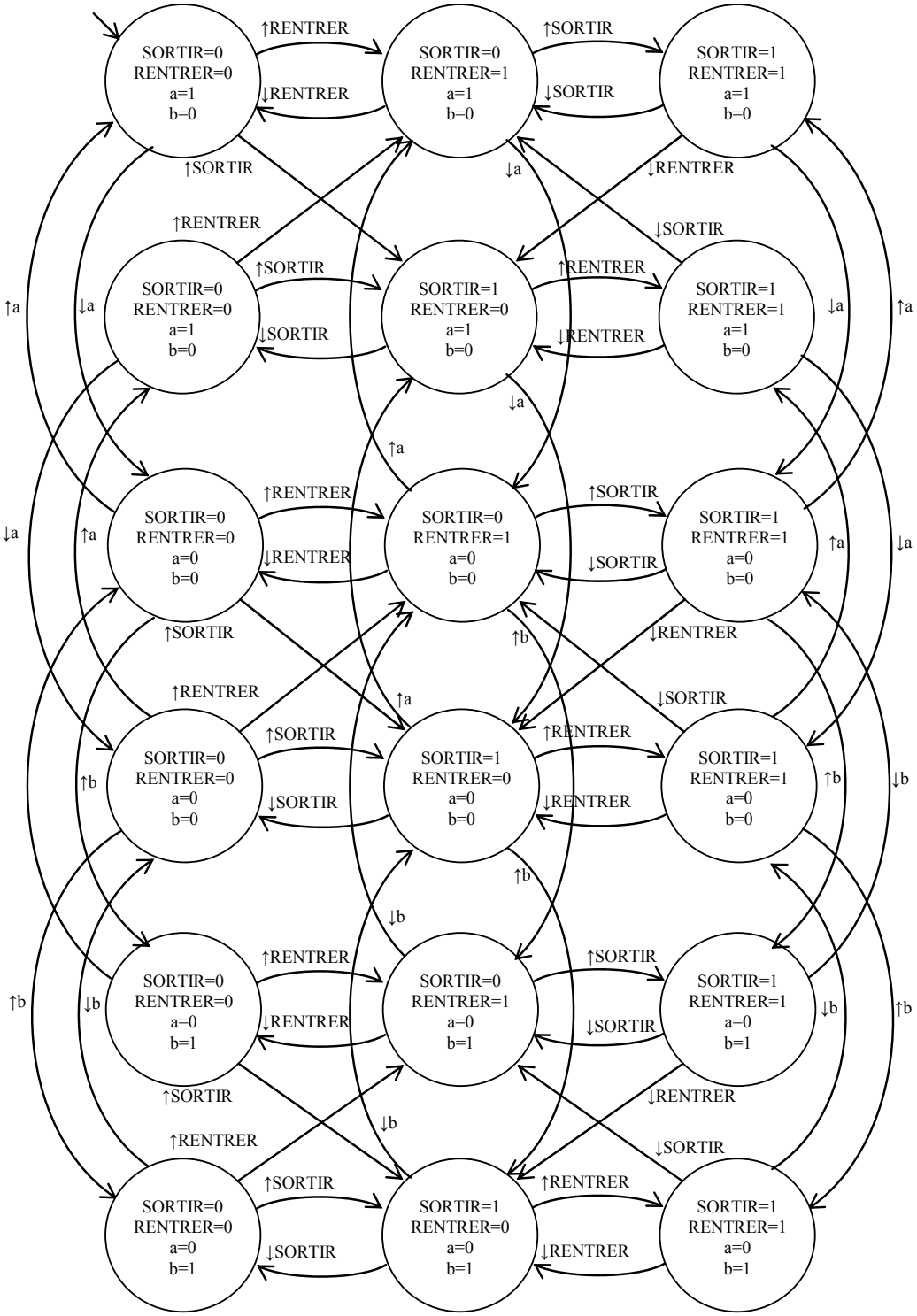


Figure 3.25 : modèle de partie opérative de la chaîne fonctionnelle sans hypothèse

Cette démarche de modélisation permet également de représenter le comportement de chaînes fonctionnelles ayant un comportement plus complexe. La chaîne fonctionnelle constituée d'un vérin double effet associé à un distributeur à trois positions et équipé de trois capteurs mesurant à la fois les limites de fin de course et une position intermédiaire utilisée pour arrêter la course du vérin dans une position supplémentaire définie. Deux types de distributeurs permettent d'assurer cette fonction avec une particularité : le distributeur à centre ouvert tolère que la tige du vérin se déplace par des mouvements exercés par une action extérieure à l'alimentation en énergie pneumatique générée par le distributeur. Le distributeur à centre fermé ne tolère pas qu'un mouvement de la tige du vérin se produise lorsque le vérin n'est pas alimenté en énergie. Cette différence peut être mise en avant lors de la modélisation de la chaîne fonctionnelle avec les deux types de distributeur. Les deux variantes sont successivement présentées dans la suite.

c. Modélisation du vérin avec un distributeur à trois positions centre ouvert et trois détecteurs

Le distributeur à centre ouvert de la chaîne fonctionnelle représentée par le schéma de la Figure 3.26 est composé de trois positions possibles permettant soit de sortir la tige du vérin, soit de rentrer la tige ou soit de laisser libre le mouvement de la tige. La modélisation du comportement de ce système peut être réalisée par la démarche utilisée jusqu'à présent.

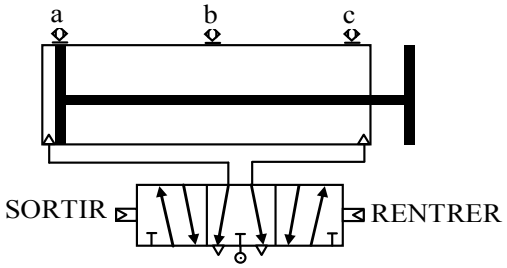


Figure 3.26 : chaîne fonctionnelle « vérin pneumatique avec distributeur à trois positions centre ouvert, deux capteurs fin de course et un capteur intermédiaire »

Le distributeur utilisé pour effectuer l'alimentation en énergie pneumatique du vérin comporte trois modes d'alimentation en air comprimé suivant les niveaux pris par les commandes (Figure 3.27). A chaque état du distributeur correspond un mode d'alimentation en air comprimé de l'actionneur. Dans l'état 1, le distributeur alimente le vérin dans le but de faire sortir la tige du corps du vérin. Dans l'état 3, le vérin est alimenté pour faire rentrer la tige dans le corps du vérin. L'état intermédiaire 2 permet de ne pas alimenter les chambres du

vérin en air comprimé. Les chambres du vérin sont toutes les deux mises à l'échappement ce qui permet d'effectuer des mouvements par le vérin sans que le distributeur ne les empêche.

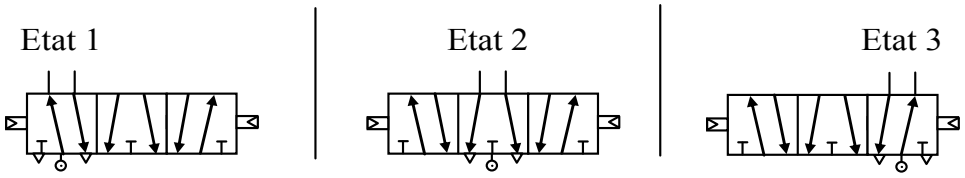


Figure 3.27 : positions possibles du tiroir du distributeur 5/3

La table de vérité permet de décrire la relation entre les niveaux des commandes et l'état pris par le distributeur. La table de vérité est construite en énumérant l'état pris par le distributeur en fonction de toutes les configurations pouvant être prises par les commandes (Figure 3.28). Lorsqu'une seule commande est au niveau haut, le distributeur prend l'état que la commande qui est au niveau haut lui demande de prendre. En revanche, lorsque les commandes sont simultanément au niveau haut ou au niveau bas, le troisième état du distributeur permet de ne pas conserver l'un ou l'autre des états précédemment pris par le distributeur. Dans cet état, le distributeur permet au vérin de se déplacer librement.

RENTRER	SORTIR	Etat du distributeur
0	0	Etat 2
0	1	Etat 3
1	0	Etat 1
1	1	Etat 2

Figure 3.28 : table de vérité du distributeur à trois positions en fonction des commandes

Cette table de vérité permet d'obtenir le modèle de comportement du préactionneur sous la forme d'un modèle automate à états finis. Cette table de vérité peut être proposée à partir des constatations effectuées lors d'essais préalables ou à partir de la documentation technique fournie par le constructeur. Pour construire ce modèle, on liste toutes les configurations possibles des niveaux des commandes et de l'état du préactionneur. Dans le cas présent, à chaque configuration prise par les commandes, il n'y a qu'une seule possibilité pour l'état pris par le préactionneur. Par conséquent, il n'y a que quatre configurations possibles correspondant aux quatre lignes de la table de vérité. Le modèle de préactionneur se compose de quatre états. Les évènements décrivent tous les changements possibles de niveau des commandes. Le modèle nécessite de décrire un état initial en cohérence avec les niveaux des commandes et l'état du préactionneur lors de la mise ne marche du système. On suppose qu'à l'initialisation les commandes sont au niveau bas et donc que le préactionneur est dans le mode d'alimentation 2 (Figure 3.29). Depuis chaque état du modèle, chaque commande est susceptible de changer de niveau. Il y a donc deux transitions sortantes pour chacun de ces états. Le modèle de préactionneur permet ainsi de déterminer, sous réserve que le domaine de

validité du modèle soit atteint, le mode d'alimentation en énergie du vérin piloté par ce distributeur.

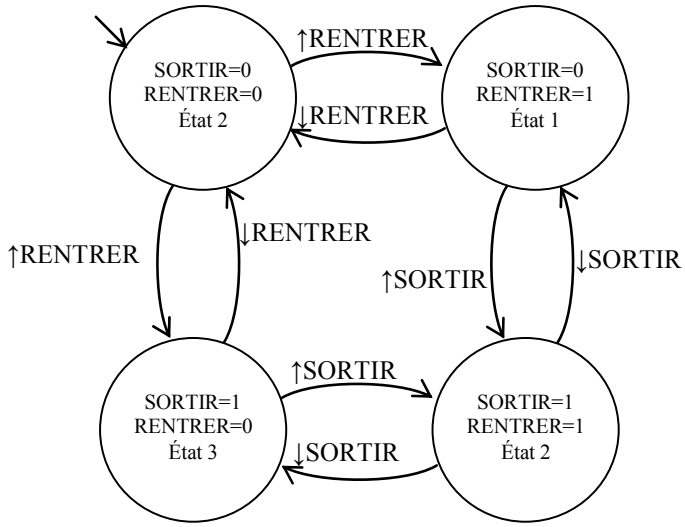


Figure 3.29 : modèle de distributeur 5/3

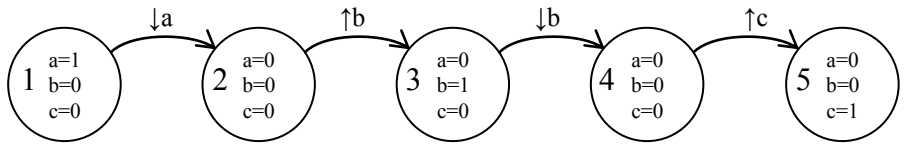
La modélisation du comportement de l'actionneur passe par la seule vision partielle perçue au travers des capteurs disposés sur la course du vérin. Pour modéliser l'ensemble actionneur/capteurs, il faut tout d'abord lister les situations de l'actionneur qui peuvent être distinguées par les capteurs. Toutes ces situations sont représentées dans le tableau de la Figure 3.30. A partir de l'observation de la disposition des capteurs sur l'actionneur, cinq situations différentes peuvent être distinguées. Trois d'entre elles correspondent aux endroits où un des capteurs est au niveau haut. Les deux autres situations sont liées au passage intermédiaire entre le passage au niveau bas d'un capteur au passage au niveau haut d'un autre capteur. Ces deux situations ne peuvent pas être distinguées l'une de l'autre uniquement à partir des niveaux des différents capteurs. Il est nécessaire d'avoir une information supplémentaire apportée par l'historique des évolutions des capteurs. Comme nous allons le voir par la suite, cette information n'est pas toujours suffisante pour effectuer la distinction entre ces deux situations.

	situation	**niveau du capteur**
1	rentrée	a=1, b=0, c=0
2	entre rentrée et intermédiaire	a=0, b=0, c=0
3	intermédiaire	a=0, b=1, c=0
4	entre intermédiaire et sortie	a=0, b=0, c=0
5	sortie	a=0, b=0, c=1

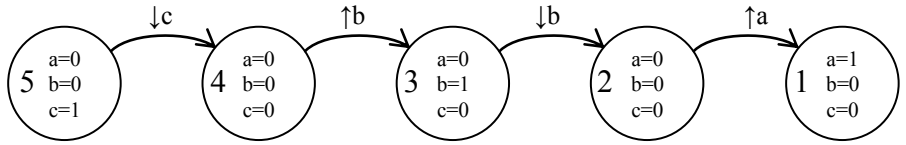
Figure 3.30 : situations distinguables du vérin équipé de trois capteurs

La modélisation de l'actionneur passe par une représentation formelle des changements de situations susceptibles de se produire en fonction de l'action que l'actionneur effectue.

Action de sortie



Action de rentrée



Action libre

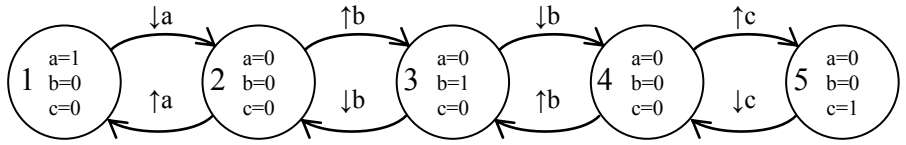


Figure 3.31 : évolution des capteurs en fonction des actions

La modélisation du comportement lors de l'action de sortie permet de représenter la succession d'évolutions des capteurs susceptibles de se produire lors de cette action. Ainsi, en fonction de la situation courante de l'actionneur identifiée à l'aide des capteurs, le vérin passe de la situation rentrée à la situation sortie en passant par la situation intermédiaire et les deux situations où aucun des capteurs ne sont au niveau haut. Pour l'action de sortie, il s'agit de la même opération mais dont le sens est inversé.

Au delà de ces deux actions déjà rencontrées dans les précédents modèles, il est nécessaire d'évoquer un troisième cas de figure lorsque que ni l'action de sortie ni l'action de rentrée ne sont actives. En effet, l'observation de l'alimentation en énergie de l'actionneur suppose que l'actionneur est susceptible d'être laissé sans alimentation en énergie. Dans ce dernier cas de figure, il est indispensable de s'intéresser aux évolutions de capteurs qui pourront intervenir. La dénomination d'action libre évoque que l'actionneur n'est pas influencé par le préactionneur dans ses mouvements. Le modèle d'actionneur/capteurs pour cette action est donc constitué de toutes les évolutions possibles générées par un mouvement extérieur que celui de l'alimentation en énergie fournie par le préactionneur. Ce modèle est construit en combinant à la fois l'action de rentrée et l'action de sortie.

Cette construction laisse néanmoins apparaitre un problème d'indéterminisme lors de l'occurrence du front descendant du capteur intermédiaire « *b* » depuis l'état où le capteur « *b* » est au niveau haut. En effet, il n'est pas possible d'arbitrer sur la transition à franchir dans le modèle. Le modèle d'actionneur/capteurs doit cependant ne pas faire apparaitre ce cas

de figure pour pouvoir être utilisable dans la réalisation d'un modèle de partie opérative décrit sous la forme d'un automate à états finis.

Ce problème d'indéterminisme entre ces deux évènements similaires ne peut pas être résolu par les seules informations disponibles par la partie commande. Pour pouvoir être représenté par l'outil automate à états finis, la description du modèle doit être modifiée pour masquer l'indéterminisme dans le modèle. La solution pouvant être mise en œuvre pour ce cas consiste à définir un état supplémentaire. Il s'agit d'un état où il n'est pas possible de savoir quelle est la situation réellement courante. Les états dans lesquels la situation est connue sont conservés. L'indéterminisme peut éventuellement être levé a posteriori si une évolution future est capable de déterminer quelle était la situation effective. Le modèle de partie opérative est ainsi capable de reprendre un fonctionnement normal suite à ce problème d'indéterminisme. Ce modèle est donc composé de six états correspondant à cinq états pour les situations de l'actionneur pouvant être distinguées à l'aide des capteurs et une situation correspondant à l'indéterminisme entre deux situations.

L'ajout de cet état nécessite de redéfinir la liste des situations pouvant être évaluées à l'aide des capteurs pour prendre en considération cette situation particulaire (Figure 3.32).

	situation	**niveau du capteur**
1	rentrée	a=1, b=0, c=0
2	entre rentrée et intermédiaire	a=0, b=0, c=0
3	intermédiaire	a=0, b=1, c=0
4	entre intermédiaire et sortie	a=0, b=0, c=0
5	sortie	a=0, b=0, c=1
6	entre rentrée et intermédiaire ou entre intermédiaire et sortie	a=0, b=0, c=0

Figure 3.32 : nouvelle définition des situations distinguables du vérin équipé de trois capteurs

A partir de cette liste de situations, la description du comportement pour chacune des actions possibles du système est réalisée sur la Figure 3.33. Pour l'action de sortie ou de rentrée, la structure du modèle est conservée. Le modèle composé des cinq états précédents est conservé. Le sixième état décrivant l'indéterminisme est ajouté. Aucune transition ne permet d'atteindre cet état puisqu'il n'y a aucun indéterminisme qui se pose dans ces actions. En revanche, les transitions partant du sixième état permettent de lever l'indéterminisme et de confirmer la situation atteinte suite à une évolution levant l'indéterminisme. Suivant l'occurrence de l'évènement, deux cas de figure peuvent être traités : l'action en cours limite les évolutions possibles aux seules évolutions pouvant avoir lieu dans la situation d'indéterministe.

Pour l'action libre, toutes les évolutions ne posant pas de problème d'indéterminisme sont également conservées. En revanche, l'évènement posant problème depuis la situation 3 est

associé à la transition vers l'état contenant l'indéterminisme. Toutes les évolutions susceptibles de se produire depuis l'état contenant la situation 6 sont insérées dans le modèle pour lever l'indéterminisme et représenter la situation atteinte. Ainsi, le passage au niveau haut du capteur intermédiaire signal que l'actionneur se trouve dans la situation 3. Pour l'activation d'un des capteurs « *a* » ou « *b* », l'état 6 indique s'il correspondait à l'état 2 ou l'état 4. Dans le cas de l'activation du capteur « *b* », il n'est pas possible de savoir lequel des états 2 ou 4 correspondaient l'état 6.

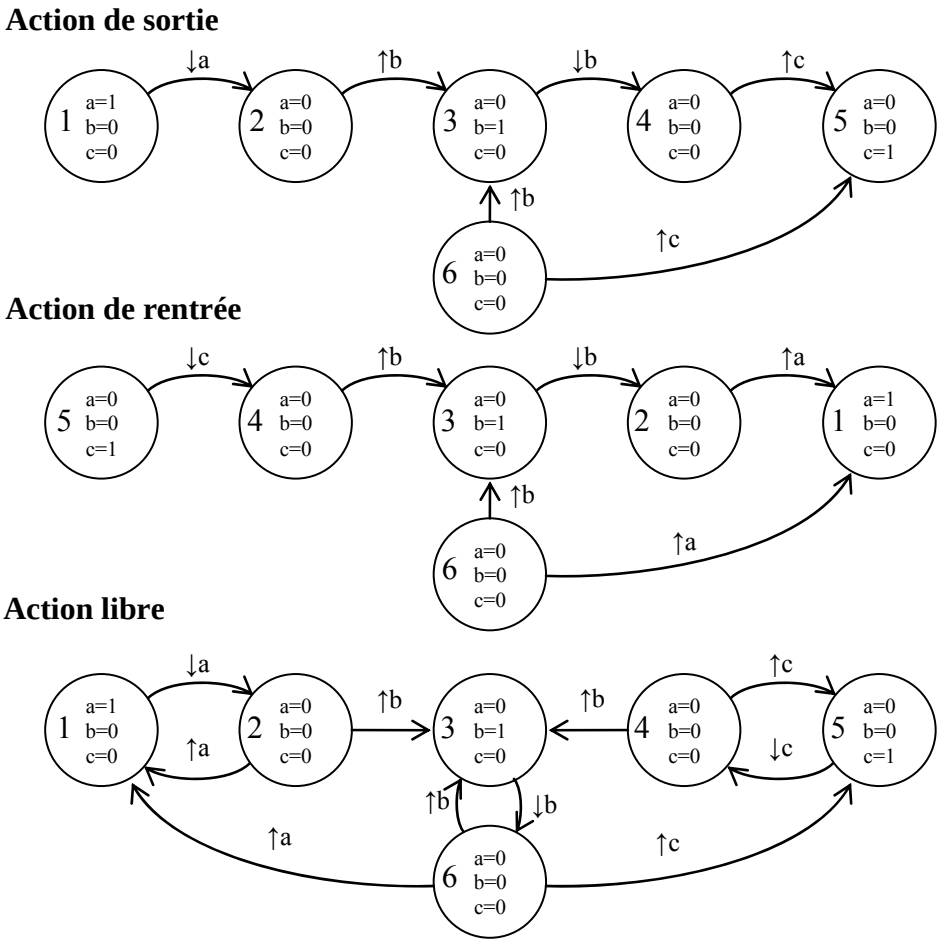


Figure 3.33 : évolution des capteurs en fonction des actions avec résolution de l'indéterminisme

Cette manipulation ne permet pas de lever totalement l'indéterminisme pouvant avoir lieu mais permet d'obtenir une représentation fidèle du comportement de la chaîne fonctionnelle et d'être compatible avec l'outil de modélisation.

Cependant, on peut noter qu'en fonction de l'utilisation qui est faite du modèle de partie opérative, certaines applications peuvent voir leur fonctionnement altéré par cet indéterminisme : dans les applications de diagnostic, la résolution de l'indéterminisme peut masquer une défaillance dans le fonctionnement du système puisqu'à cause de

l'agglomération des situations, le modèle de partie opérative suppose que les deux évolutions sont possibles dans la suite du fonctionnement.

Le modèle de partie opérative est ensuite obtenu par l'assemblage du modèle de préactionneur et le modèle d'actionneur/capteurs. L'hypothèse permettant d'assimiler les actions du modèle de l'actionneur avec les modes d'alimentation fournies par le modèle de préactionneur peut être formulée si les conditions d'application de cette hypothèse sont vérifiées.

Les trois modes de d'alimentation possibles de l'actionneur par le préactionneur dans le modèle de préactionneur sont définis à partir des évolutions des commandes. Les trois actions qui ont été décrites dans le modèle d'actionneur/capteurs décrivent les évolutions des capteurs en fonction de l'action en cours. En associant une action à chaque mode d'alimentation, on restreint les évolutions possibles des capteurs en fonction de l'évolution des commandes.

Le modèle présenté sur la Figure 3.34 représente le comportement de la chaîne fonctionnelle. Toutes les évolutions correspondent aux changements de niveaux des commandes et des capteurs. Seules les évolutions cohérentes avec le fonctionnement réel du système sont représentées. Les traits en pointillés représentent les transitions liant les états représentés avec les états qui ne sont pas représentés pour des raisons de lisibilité.

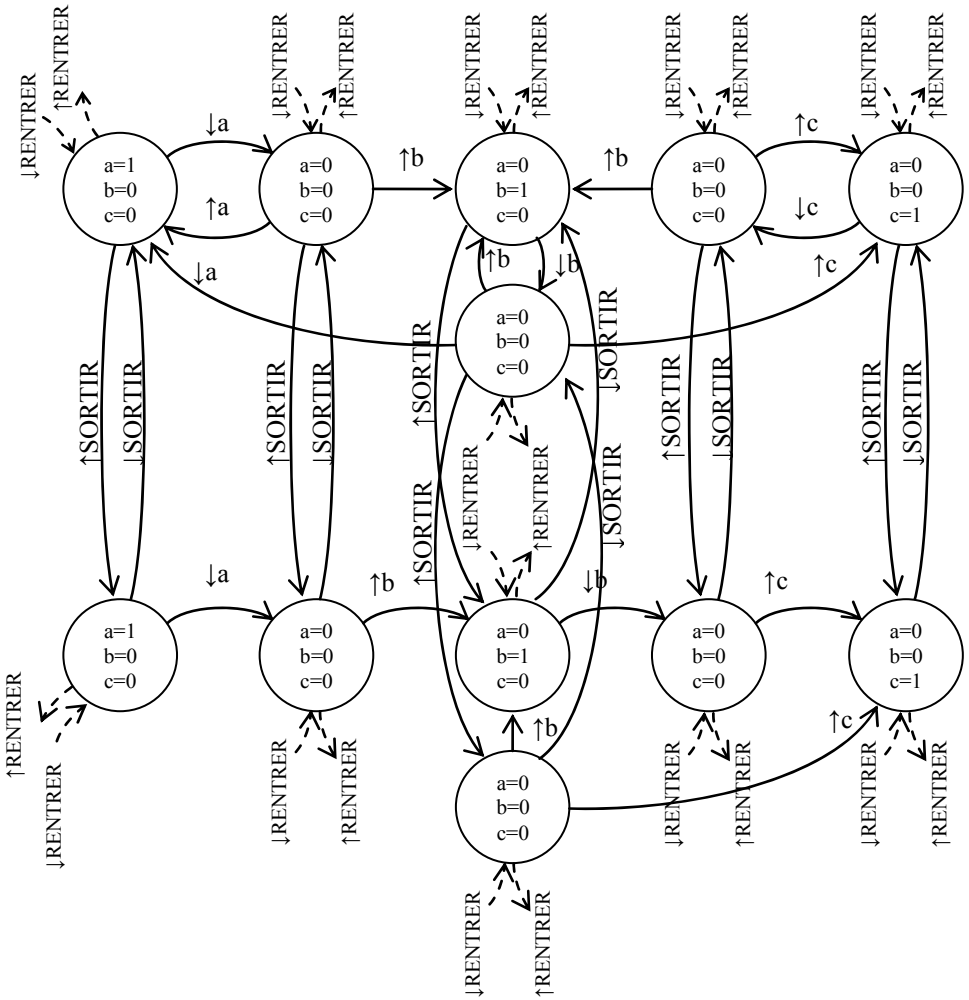


Figure 3.34 : modèle de partie opérative partiel de la chaîne fonctionnelle

d. Modélisation du vérin avec un distributeur à trois positions centre fermé et trois détecteurs

Le comportement du distributeur 5/3 à centre fermé diffère du comportement du distributeur 5/3 à centre ouvert. Cette différence peut être mise en avant en modélisant une chaîne fonctionnelle similaire à la précédente en ne changeant que le type de distributeur. L'actionneur est donc toujours un vérin pneumatique double effet avec deux capteurs mesurant les positions de fin de course et un capteur intermédiaire. Le distributeur à centre fermé de la chaîne fonctionnelle représentée par le schéma de la Figure 3.35 comporte trois états possibles permettant soit de sortir la tige du vérin, de rentrer la tige ou de bloquer le mouvement de la tige. La modélisation du comportement de ce système peut être réalisée par la démarche utilisée jusqu'à présent.

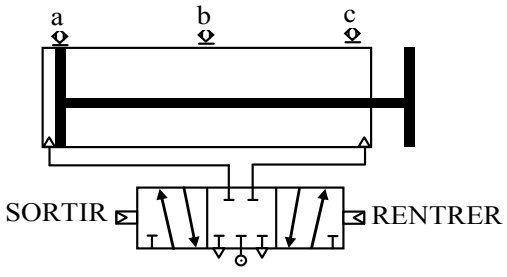


Figure 3.35 : chaîne fonctionnelle « vérin pneumatique avec distributeur à trois positions centre fermé, deux capteurs fin de course »

Le distributeur utilisé pour effectuer l'alimentation en énergie pneumatique du vérin comporte trois modes d'alimentation en air comprimé suivant les niveaux pris par les commandes (Figure 3.27). A chaque état du distributeur correspond un mode d'alimentation en air comprimé de l'actionneur. Dans l'état 1, le distributeur alimente le vérin dans le but de faire sortir la tige du corps du vérin. Dans l'état 3, le vérin est alimenté pour faire rentrer la tige dans le corps du vérin. L'état intermédiaire 2 permet de ne pas alimenter les chambres du vérin en air comprimé. Les chambres du vérin sont toutes les deux obturées ce qui empêche tout mouvement de la tige du vérin.

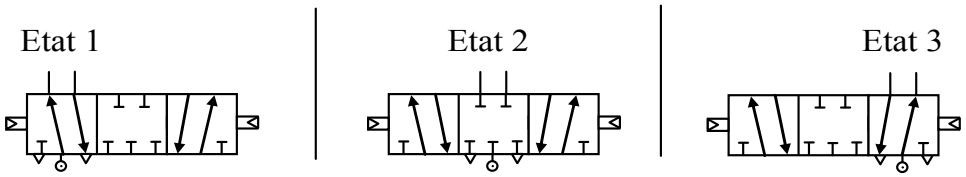


Figure 3.36 : positions possibles du tiroir du distributeur 5/3

L'établissement du modèle de préactionneur est réalisé de façon similaire à la modélisation du distributeur 5/3 à centre ouvert. La différence entre les deux distributeurs ne se situe pas au niveau de son comportement à proprement parlé mais des modes d'alimentation en énergie de l'actionneur. Le modèle de comportement du distributeur est donc conservé pour le modèle du distributeur 5/3 qu'il soit à centre ouvert ou à centre fermé.

L'actionneur et les capteurs sont les mêmes que ceux utilisés dans la chaîne fonctionnelle avec le distributeur 5/3 à centre ouvert. Les situations distinguées à l'aide de ces capteurs sont donc identiques quelque soit le distributeur utilisé.

La différence de comportement apparait lors de la description du comportement pour chaque action possible du système. Le modèle comporte six états correspondant aux cinq situations et la situation comportant l'indéterminisme. L'action de sortie et de rentrée ont le même comportement que dans le modèle précédent. En revanche, l'action de blocage apparait en remplacement de l'action libre. Dans cette action, aucune évolution de capteur n'est acceptée car l'actionneur est bloqué. Les capteurs ne peuvent évoluer que s'il y a une évolution de la position du piston du vérin. Le problème d'indéterminisme évoqué dans l'action libre avec le

distributeur 5/3 à centre ouvert n'intervient pas. Il n'y a donc pas lieu d'ajouter un état supplémentaire dans le modèle de l'actionneur. Les différentes évolutions de capteurs sont représentées sur la Figure 3.37.

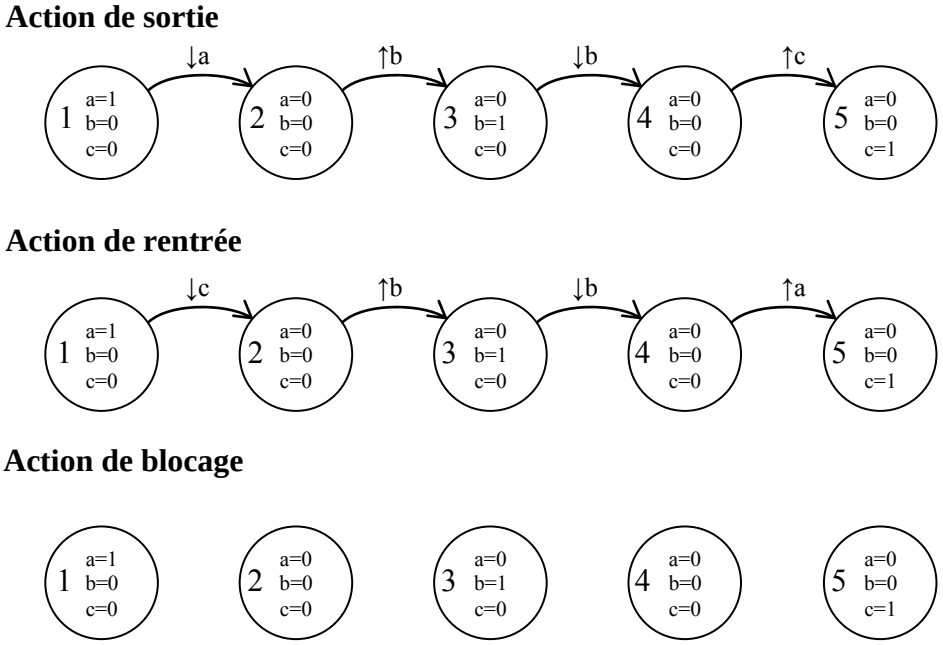


Figure 3.37 : évolution des capteurs en fonction des actions

Le modèle de partie opérative est obtenu en combinant le modèle des actions avec le modèle de préactionneur. Le modèle obtenu en effectuant l'hypothèse que l'action est connue à partir du mode d'alimentation déterminé dans le modèle de préactionneur. Dans ce cas de figure qui ne permet pas de tenir compte des temps de commutation du distributeur et de l'inertie de l'actionneur, le modèle de partie opérative de la chaîne fonctionnelle inhibe les évolutions des capteurs lorsque les commandes demandent à l'actionneur de ne pas se déplacer.

Si l'hypothèse n'est pas formulée, il n'est pas possible de déterminer quelle est l'action qui est en train de se produire. Cela ne permet donc pas d'utiliser le modèle décrivant les évolutions des capteurs en fonction des actions. Dans ce cas, la seule solution consiste à prendre en compte toutes les évolutions possibles des capteurs sans prendre en considération les commandes. Les modèles de partie opérative des chaînes fonctionnelles utilisant le distributeur 5/3 à centre ouvert et à centre fermé sont alors similaires puisqu'il n'est pas possible de distinguer quelle action est effectuée.

e. Modélisation d'un plateau tournant à deux sens de rotation et deux détecteurs

Une chaîne fonctionnelle n'est pas toujours composée d'un actionneur évoluant suivant une course limitée : tous les actionneurs examinés jusqu'à présent évoluaient suivant une course où il était possible de définir des limites. D'autres actionneurs n'ont pas cette propriété : les

moteurs rotatifs pouvant être utilisés pour déplacer un plateau tournant n'ont pas de limites quant à la course du mouvement. La modélisation de la partie opérative doit donc tenir compte et faire apparaitre cette particularité dans le résultat obtenu. Pour pouvoir illustrer ce cas de figure, la chaîne fonctionnelle choisie comme exemple est composée d'un actionneur pneumatique rotatif. Deux capteurs permettent d'identifier deux positions caractéristiques de l'axe du moteur. Le préactionneur utilisé est un distributeur 5/3 à centre fermé. Les deux commandes permettant de piloter le moteur sont notées « *T. SENS* + » et « *T. SENS* – » et permettent de faire tourner le moteur dans un sens ou dans l'autre. Cette chaîne fonctionnelle est représentée sur la Figure 3.38.

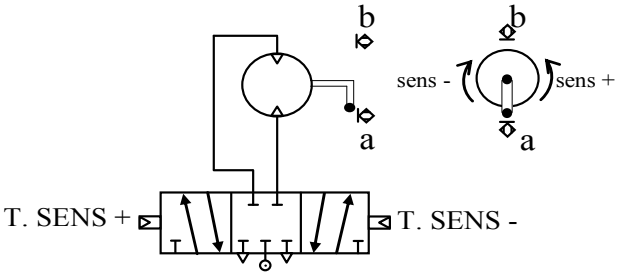


Figure 3.38 : chaîne fonctionnelle « moteur rotatif pneumatique avec distributeur à trois positions centre fermé, deux capteurs »

Le distributeur 5/3 centre fermé a déjà été modélisé dans les exemples précédents. Le modèle est représenté sur la Figure 3.39 avec une adaptation des noms donnés aux commandes.

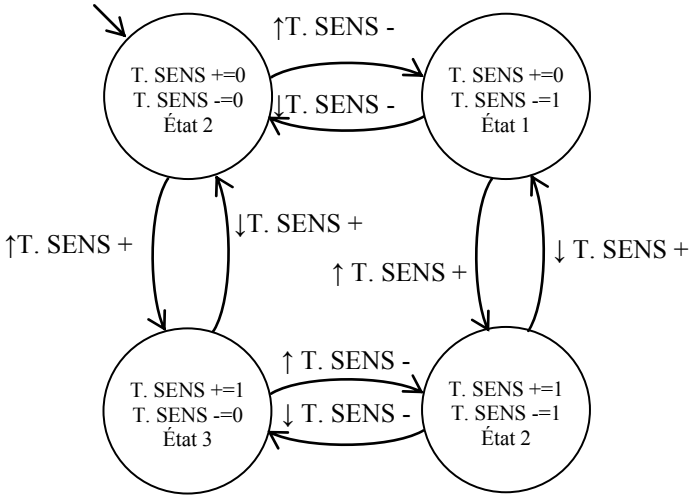


Figure 3.39 : modèle de distributeur 5/3

La position de l'actionneur est perçue au travers des capteurs indiquant la position de l'axe du moteur. La première étape consiste donc à lister les situations d'actionneur pouvant être distinguées à partir des données issues des capteurs. Avec les deux capteurs présents, quatre situations peuvent être distinguées dans l'orientation de l'axe du moteur. Deux orientations sont directement mesurées à l'aide des capteurs présents. Lorsque les deux capteurs ne

détectent pas le dispositif installé sur l'axe, deux situations sont possibles en fonction du mouvement qu'à effectuer l'axe du moteur. La notation « entre a et b » signifie que l'axe du moteur est situé dans la zone où l'axe du moteur passe du capteur *a* vers le capteur *b* pour une rotation dans le sens positif. Ces informations sont rassemblées dans le tableau de la Figure 3.40.

	situation	**niveau du capteur**
1	position a	a=1, b=0
2	entre a et b	a=0, b=0
3	position b	a=0, b=1
4	entre b et a	a=0, b=0

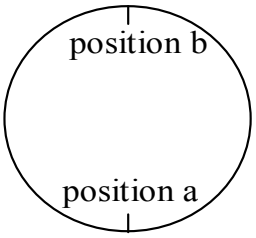


Figure 3.40 : situations distinguables du moteur et des deux capteurs

Le modèle d'actionneur/capteurs décrit les évolutions possibles des capteurs en fonction de l'action se produisant. Trois actions sont possibles : la rotation dans le sens positif, la rotation dans le sens négatif et le blocage. Le blocage interdit toutes les évolutions possibles des capteurs. Une rotation du plateau implique qu'il y ait toujours des évolutions possibles des capteurs car il n'y a pas de situation stable atteignable lorsque l'action est exercée suffisamment longtemps (Figure 3.41).

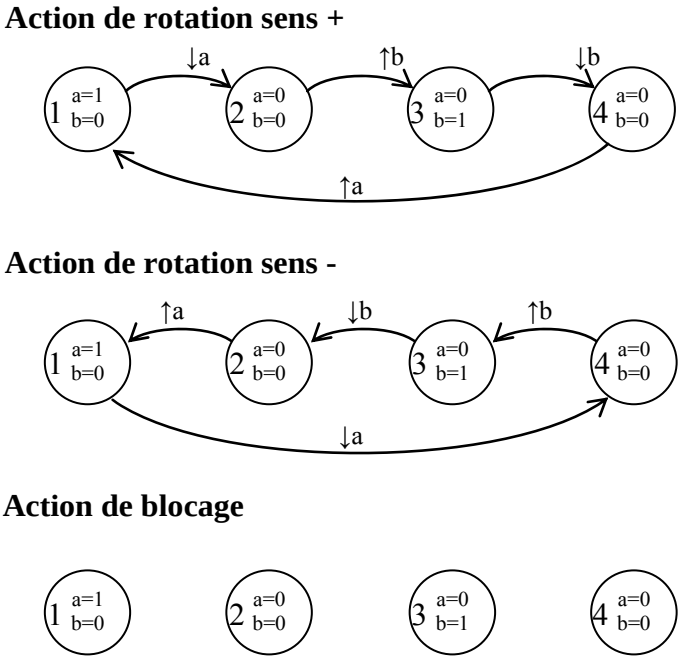


Figure 3.41 : évolution des capteurs en fonction des actions

Le modèle de partie opérative est obtenu en combinant le modèle des actions avec le modèle de préactionneur. Le modèle obtenu en effectuant l'hypothèse que l'action est connue à partir

du mode d'alimentation déterminé dans le modèle de préactionneur. Dans ce cas de figure qui ne permet pas de tenir compte des temps de commutation du distributeur et de l'inertie de l'actionneur, le modèle de partie opérative de la chaîne fonctionnelle inhibe les évolutions des capteurs lorsque les commandes demandent à l'actionneur de ne pas se déplacer.

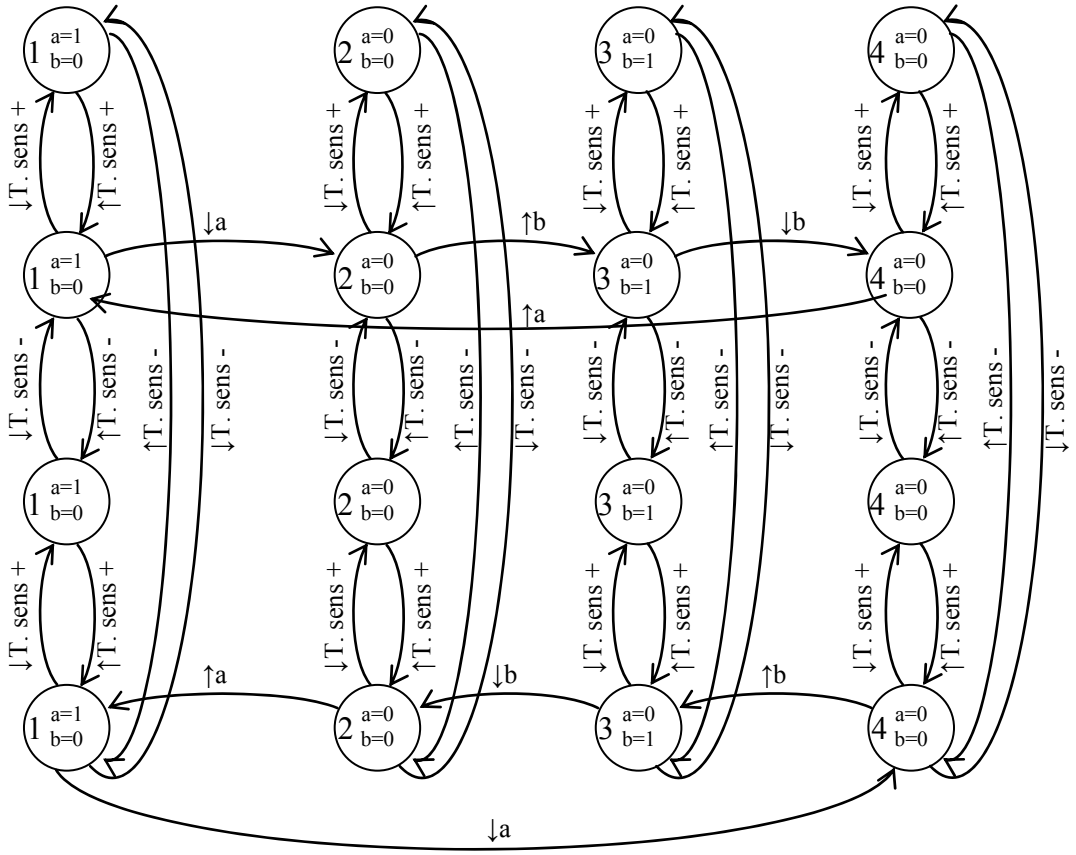


Figure 3.42 : évolution des capteurs en fonction des actions

Ce modèle inclut toutes les possibilités d'évolution de la partie opérative pour le plateau tournant. La méthodologie de modélisation permet donc de prendre également en compte les actionneurs qu'ils aient une course limitée ou en boucle illimitée.

La modélisation à l'aide des automates à états finis permet de représenter le comportement d'un système automatisé par les évolutions successives qui ont lieu au cours du temps. Les travaux utilisant cet outil peuvent exploiter la démarche de modélisation proposée pour utiliser des modèles de partie opérative. Il ne s'agit néanmoins que d'une représentation partielle car elle n'apporte qu'une vision discrète du fonctionnement de la partie opérative. Pour obtenir une modélisation plus proche de la réalité pour les modèles de simulation, il peut être nécessaire de prendre en considération le comportement continu des évolutions physiques de la partie opérative d'un système automatisé.

5. Applications

Pour montrer l'apport de la modélisation de la partie opérative, les différentes approches ont été mises en œuvre sur des systèmes automatisés. Différentes solutions d'implémentation ont été éprouvées pour définir les avantages et les inconvénients de ces solutions. Une partie opérative, la plateforme Cellflex, a été utilisée pour mettre en application la démarche de modélisation.

a. Plateforme Cellflex

L'Université de Reims Champagne-Ardenne s'est dotée d'une plateforme d'automatisation appelée Cellflex en 2007. Cette plateforme (Figure 3.43) à vocation emballage et conditionnement, regroupe toutes les opérations d'embouteillage.

Figure 3.43 : plateforme Cellflex

La mise en place de cette plateforme a nécessité une forte implication pour atteindre un fonctionnement correct. Depuis la rédaction du cahier des charges de l'appel d'offre à la réception de l'équipement et à l'organisation des premières séances d'enseignement, le temps consacré à la plateforme a permis de mettre en marche et de développer les premières applications sur cette plateforme. Les travaux d'installation de l'énergie, l'organisation de la

salle dans laquelle la plateforme a été installée ont du être suivis. Les postes informatiques ont été préparés, mis en place et les logiciels informatiques nécessaires à l'utilisation de la plateforme ont été installés. Suite à la réception des différents matériels, les réglages mécaniques indispensables au bon fonctionnement ont également du être effectués pour permettre un comportement normal et répétable de la plateforme. Le suivi de la formation et la rédaction de documents permettant de réutiliser les connaissances acquises pour concevoir la commande et la supervision de la plateforme ont été effectués. La plateforme Cellflex a également fait l'objet de l'organisation de plusieurs manifestations telles que l'inauguration officielle le 24 Avril 2008, la fête de la science, classes en fac et les nombreuses présentations auprès des industriels locaux.

La plateforme se compose de six stations ayant chacune des fonctionnalités.

- La station de mixage permet le mélange de liquides primaires contenus dans des cuves avant le transport vers la station de remplissage.

- La station de préparation des bouchons permet la sélection de la couleur de bouchon utilisée pour fermer la bouteille. Les bouchons de couleur blanche, rouge ou noir sont mis en place dans trois racks. Le bouchon de la couleur sélectionnée est envoyé sur un convoyeur pour être mis sur la file d'attente. Les bouchons sont ensuite expédiés vers la station de remplissage.

- La station de remplissage est constituée d'un plateau tournant à six positions sur lesquelles des opérations élémentaires sont réalisées. Les flacons vides sont d'abord acheminés vers le plateau tournant. Le flacon est ensuite rempli avec le liquide issu du mélange. Le bouchon précédemment préparé est ensuite posé sur le flacon. Le bouchon est ensuite vissé sur le flacon avant d'être évacué. Les données de fabrication sont enfin inscrites dans l'étiquette électronique RFID contenue dans le bouchon.

- La station de transfert est composée d'un convoyeur contenant les flacons qui ont été fabriqués sur la station de remplissage. Un préhenseur équipé d'une pince pouvant prendre trois flacons est disposé à l'extrémité du convoyeur. Celui-ci permet de charger, en deux mouvements, six flacons dans un emballage. Le sixpack permet de représenter le lot de fabrication.

- Le convoyeur principal permet de véhiculer les sixpacks entre les différentes stations de traitement du lot. Il est constitué d'un anneau de convoyage sur lequel transitent des palettes avec des emplacements d'arrêt et de traitement. Les sixpacks sont arrimés sur les palettes pour être déplacés. Une antenne RFID est présente sur le long du convoyeur pour inscrire les informations de traçabilité dans l'étiquette présente sur le côté du sixpack. Celle-ci permet de tracer le lot de fabrication.

- La station de stockage des sixpacks permet de représenter les flux internes circulant dans une entreprise. Elle est composée d'un bras pouvant déplacer un sixpack entre les palettes du convoyeur principal et les seize emplacements du magasin de stockage.

- La station d'import/export représente les flux externes autour de l'entreprise. Un convoyeur permet de prendre en charge les sixpacks vides. Un préhenseur permet de mettre en place les

sixpacks sur les palettes du convoyeur. Les sixpacks pleins peuvent également être évacués à l'aide de ce même convoyeur vers deux racks d'évacuation.

Ce système automatisé complexe utilise une solution de commande évoluée. Celle-ci est composée de six automates communiquant sur le réseau Ethernet. Les capteurs et les actionneurs communiquent selon différents réseaux de communication suivant les stations. Cinq écrans tactiles permettent de configurer le fonctionnement de Cellflex. Une borne Wifi permet d'accéder au réseau à distance depuis un ordinateur portable. Dix postes informatiques et un serveur de données permettent de programmer et de piloter la plateforme. La technologie de programmation particulièrement novatrice permet à l'Université de Reims Champagne-Ardenne de devenir pôle de compétence pour la programmation orientée composant d'automatisme. Etant donné la complexité de la plateforme Cellflex, seules quelques chaînes fonctionnelles ont été utilisées pour appliquer la démarche de modélisation. Nous proposons dans la suite de cette partie de décrire deux solutions mises en œuvre implémentant le modèle à états finis dans le cadre d'une utilisation en ligne.

b. Application au diagnostic intégré à l'API

Les automates Siemens S7-300 équipant la plateforme Cellflex sont programmés à l'aide de l'atelier logiciel Step7. Parmi les extensions possibles de ce logiciel, l'outil S7-Higraph permet d'intégrer des modèles de commande conçus sous la forme de graphes d'états en particulier pour l'analyse de défaillances. L'implantation dans un API nécessite un moteur d'exécution. Le modèle de partie opérative déclaré sous la forme d'un graphe d'états implanté dans l'API ne réagit donc pas de façon similaire au modèle conçu à partir de la définition mathématique des automates à états finis utilisée jusqu'à présent. En effet, l'utilisation d'un API implique nécessairement le traitement informatique du code implanté. Ce traitement génère des phénomènes de retard, de synchronisation entre les évolutions de la partie opérative. Bien que ces phénomènes aient de fortes incidences sur la modélisation, le fonctionnement du moniteur d'exécution des graphes d'états n'est pas abordé dans la documentation du constructeur. Par ailleurs, les fonctionnalités offertes par cet outil sont limitées : il ne s'agit que d'une interface graphique générant du code implanté dans l'API. Ce code interprété peut être modifié depuis l'atelier logiciel. S7-Higraph est simplement une aide graphique pour représenter un modèle en vue de son implémentation dans un API.

Une chaîne fonctionnelle composée d'un vérin avec un distributeur monostable et un capteur de fin de course a été l'objet de l'implantation d'un modèle de partie opérative dans un API de la plateforme Cellflex. Une telle chaîne fonctionnelle est nécessaire pour la distribution des bouchons dans la station de préparation des bouchons. Le modèle implanté dans S7-Higraph est présenté sur la Figure 3.44. Il est réalisé à partir de la démarche proposée précédemment (Figure 3.15) et se compose de quatre états et des transitions associées.

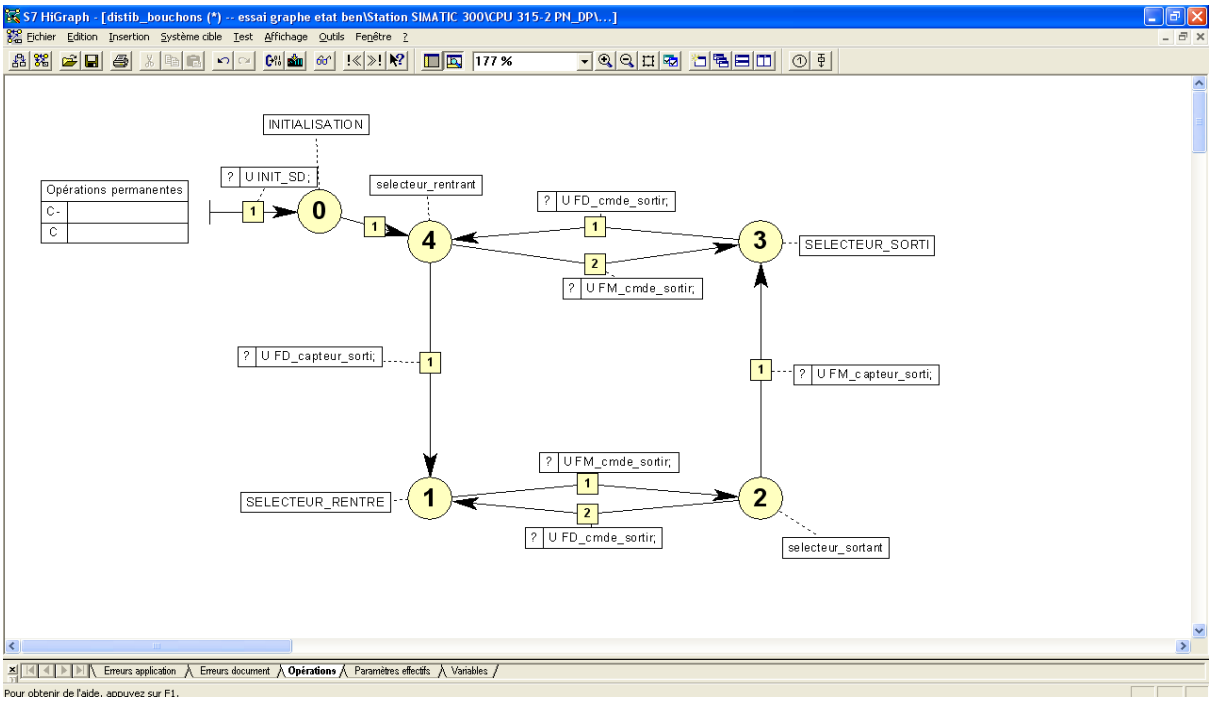


Figure 3.44 : exemple de modèle implanté dans l'API à l'aide de l'outil S7-Higraph

L'implémentation de ce modèle dans l'API pose néanmoins quelques problèmes liés à l'interprétation du modèle. Le temps de cycle de l'API laisse apparaître des phénomènes de franchissement simultanés de transitions. Pour résoudre ce problème, S7-Higraph utilise des priorités entre les transitions sortantes d'un même état. Les nombres inscrits sur les transitions représentent cette priorité. Il n'y a aucune possibilité de réaliser des franchissements simultanés car les priorités ne peuvent être égales. L'état initial est activé par la transition entrante dont la réceptivité est la mise fonctionnement du modèle de partie opérative. Le logiciel ne permet pas d'utiliser des fronts pour mettre des conditions sur les transitions. Les fronts sont donc mémorisés artificiellement dans la mémoire de l'API au sein du programme de commande classique avant l'exécution du modèle de partie opérative. Une des fonctionnalités offerte par cet outil est d'autoriser des échanges d'information entre modèles à l'aide de messages envoyés ou reçus lors du franchissement d'une transition ou lorsqu'un état est actif. Cette fonctionnalité n'est pas exploitée avec la définition de l'outil de modélisation choisie dans le cadre de la démarche de modélisation. Enfin, l'outil de modélisation ne permet pas de détecter les évolutions non prévues dans le modèle de partie opérative. Les évolutions non prises en compte dans le modèle de partie opérative ne sont pas détectées. Pour pouvoir identifier les défaillances, il est nécessaire d'ajouter des états signalant les défaillances qui seront atteints suite à une évolution non prévue dans le modèle initial. Toutes les évolutions non prévues initialement doivent être indiquées dans ce modèle de partie opérative sous peine de ne pas prendre en considération les défaillances associées.

Le résultat de cette modélisation a été implanté dans l'API et le fonctionnement normal a pu être testé. Le fonctionnement anormal n'a pas pu être mis en évidence par la plateforme

Cellflex. Par contre, la simulation du comportement a pu mettre en évidence la détection de comportements anormaux par l'ajout d'états défaillants. L'utilisation en mode pas-à-pas du mode de simulation (exécution cycle par cycle) a permis de montrer les limites de l'utilisation de cette solution d'implémentation lors de l'occurrence d'évènements simultanés. Une des solutions qui pourrait être apportée pour lever cette impossibilité actuelle serait de définir des transitions combinant plusieurs évolutions simultanées. L'outil S7-Higraph permet une telle représentation mais la complexité du modèle nécessite de s'interroger sur la représentativité graphique et la lisibilité du modèle dans ce cas de figure.

L'intérêt de l'implémentation du modèle de partie opérative dans la mémoire de l'API est de restreindre les effets de synchronisation liés aux temps de cycle de l'API au minimum. En effet, la prise en compte des évolutions de la commande et du modèle de partie opérative se font suivant des constantes de temps similaires. L'inconvénient majeur de cette implémentation intégrée du modèle de partie opérative à la commande réside dans l'utilisation des capacités restreintes de l'API. En effet, l'espace mémoire disponible dans un API est limitée à quelques kilo-octets dans le meilleur des cas. De plus, l'ajout de fonctionnalités pour traiter le modèle de partie opérative a pour conséquence d'augmenter le temps de cycle de l'API et donc de détériorer les performances intrinsèques de la commande.

Pour résoudre ce problème, une solution consiste à extraire du modèle de la commande les évolutions de la partie opérative pour les exporter vers un système externe de traitement et ainsi animer le modèle de partie opérative à partir de ces données.

c. Application à la supervision industrielle

Les systèmes de supervision industrielle de type SCADA consistent à dissocier les opérations de suivis de la production du modèle de commande. L'implantation d'interfaces opérateur sur des dalles tactiles ou dans des pupitres nécessite l'utilisation de matériels informatiques possédant des capacités de stockage et de traitement plus importantes que les API. Ces ressources peuvent être exploitées pour implanter le modèle de partie opérative. Les systèmes SCADA communiquent avec l'API suivant des protocoles très variés. L'utilisation toujours plus grande des outils informatiques issus des sciences de l'information et de la communication ont permis d'aboutir à des standards normalisés d'échanges de données au niveau applicatif.

Cette facilité d'accès à l'information est exploitée dans l'application informatique développée pour utiliser la modélisation de la partie opérative en ligne en vue d'apporter de l'information sur les interfaces opérateur. Le standard OPC (OLE for Process Control) est une extension de la communication logicielle développée par Microsoft pour les logiciels d'automatisme. Pour être mis en œuvre, la communication requiert un serveur OPC, installé sur un poste informatique, qui est configuré pour récupérer les données de l'API par la communication de la couche physique et les mettre à disposition des applications tierces. Le client OPC exploite et traite les données issues du serveur OPC suivant les fonctionnalités à réaliser. Les logiciels SCADA utilisés en production sont des clients OPC. L'implantation d'un modèle de partie

opérative dans un SCADA n'est néanmoins pas facile à mettre en œuvre. En effet, les SCADA sont conçus pour des fonctions précises de visualisation. La modélisation de la partie opérative pourrait faciliter la mise en œuvre de ces solutions logicielles mais elle n'est pas mise en avant. Pour utiliser la modélisation de la partie opérative, notre choix s'est donc porté vers le développement dans un des langages de programmation de la suite .NET d'un client OPC dédié à la modélisation de la partie opérative [Rohee, 2007a]. Le langage choisi est le C# car il combine une syntaxe proche du C++ avec la programmation orientée objet du VisualBasic. La solution retenue est de synchroniser le modèle de partie opérative à la partie opérative réelle avec les évolutions des commandes et des capteurs contenus dans l'API vers l'application. Une interface de programmation a été développée pour décrire le modèle de partie opérative dont la Figure 3.45 présente le diagramme de classe UML de cette interface de programmation.

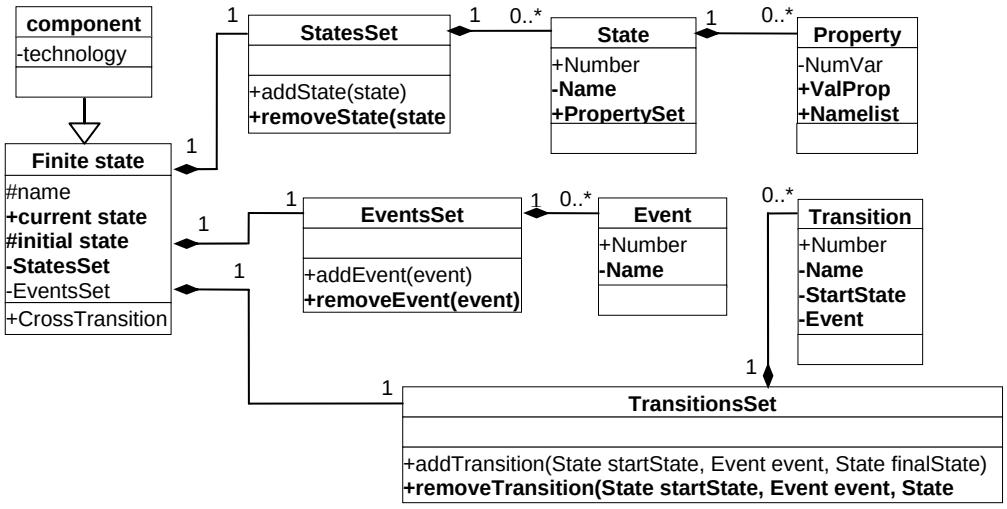


Figure 3.45 : diagramme de classe UML de l'interface de programmation

Les transitions du modèle de partie opérative sont ensuite associées aux variables tampon images des commandes et des capteurs contenues dans le serveur OPC. Le protocole de communication du serveur OPC est configuré pour permettre l'échange des données entre l'API et le poste informatique au travers du réseau Ethernet (Figure 3.46). L'automate à états est implémenté au travers une interprétation du comportement. Le serveur OPC actualise périodiquement les données de l'API et prévient l'application développée des changements éventuels. Le modèle étant implémenté sous la forme de lignes de code, l'ordre d'exécution des lignes prend de l'importance dès lors que plusieurs changements simultanés interviennent. Cette forme d'interprétation nécessite donc que le délai minimum entre les évolutions soit supérieur au temps de rafraichissement périodique des données contenues dans le serveur OPC de l'ordre de 25ms lors des essais réalisés sur Cellflex.

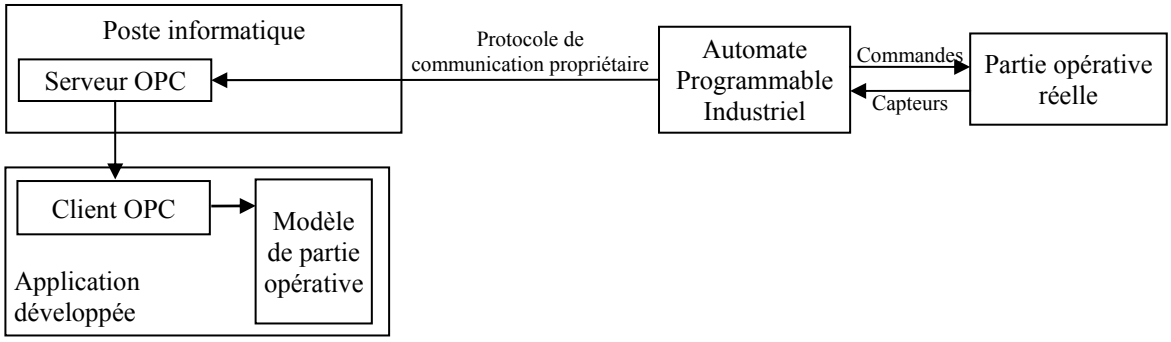


Figure 3.46 : implantation du modèle de partie opérative dans un poste informatique distant

Les informations construites à l'aide du modèle de partie opérative sont ensuite exploitées pour générer l'affichage de l'état de la partie opérative et de ses tendances sur un écran graphique complétant l'application ainsi développée. L'exemple qui a été choisi est le préhenseur utilisé sur la plateforme Cellflex pour mettre en place les flacons produits dans les sixpacks. Le préhenseur considéré est constitué de deux vérins équipés de distributeurs bistables et de deux détecteurs de fin de course (Figure 3.47).

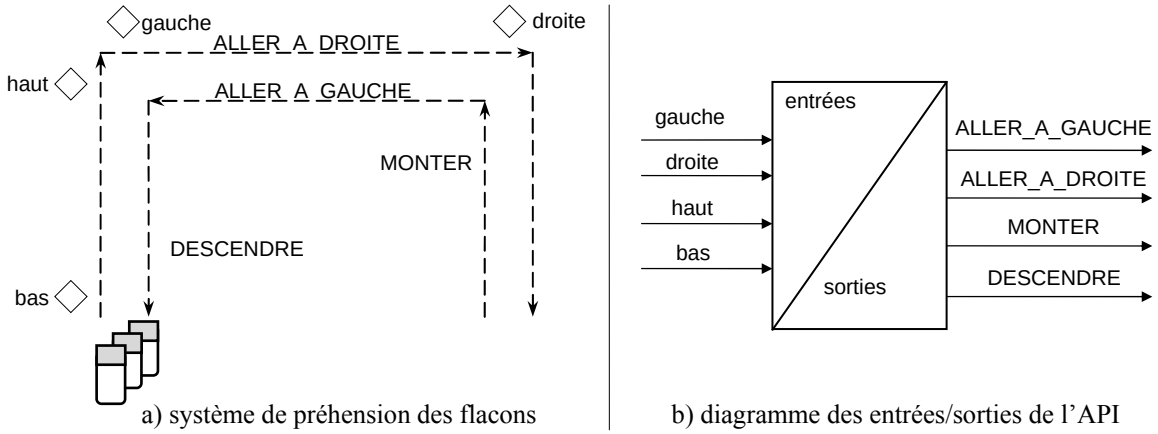


Figure 3.47 : système de préhension des flacons

Les essais réalisés à l'aide de cette application ont permis de montrer les avantages et les inconvénients de cette implantation. L'interface graphique de la Figure 3.48 exploitant le modèle permet de se rendre compte de la configuration dans laquelle se trouve la partie opérative. Le préhenseur est représenté par l'assemblage des deux rectangles formant le bras. Les losanges représentent l'état des capteurs : un losange plein signifie que le capteur est au niveau logique haut tandis qu'un losange vide signifie que le capteur est au niveau logique bas. Les flèches indiquent l'action déterminée à l'aide des commandes en cours. Toutes ces informations sont directement extraites du modèle de partie opérative : à chaque état du modèle d'une chaîne fonctionnelle correspond une configuration (état des commandes et des capteurs) connue de l'actionneur et des actions. Une image est donc générée pour chaque

chaîne fonctionnelle sur l'interface de visualisation représentant la configuration de l'actionneur et des actions. Le modèle a été agrémenté d'une fonction permettant d'avertir lorsque qu'une évolution imprévue dans le modèle de partie opérative est détectée. Un message est alors indiqué sur l'interface pour avertir l'opérateur de la détection d'un comportement anormal.

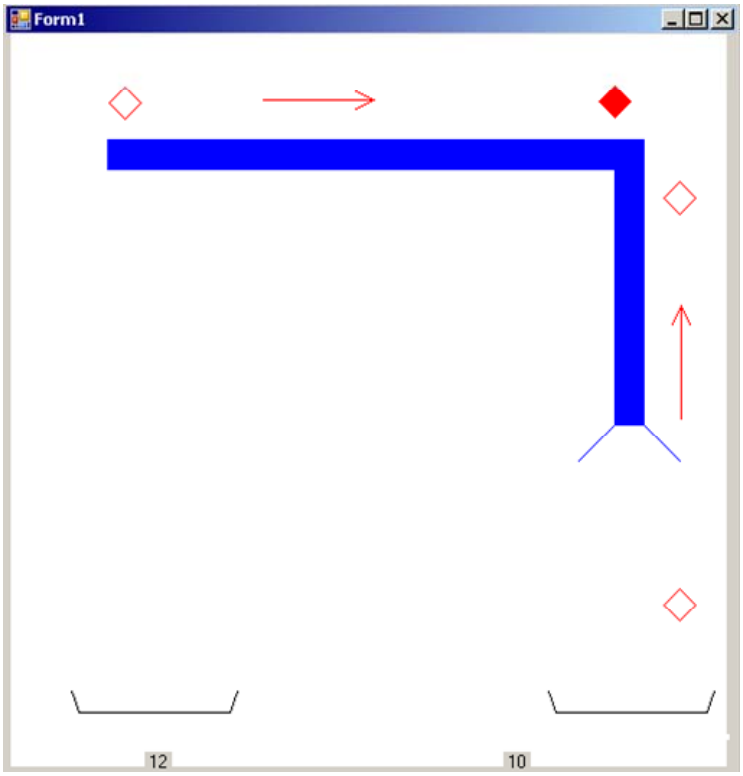


Figure 3.48 : interface graphique de visualisation du préhenseur

Le modèle de partie opérative n'est plus limité par les capacités de traitement et de stockage de l'API qui sont prises en compte par le poste informatique sur lequel l'application est lancée et donc les capacités de traitement et de stockage sont plus importantes. En revanche, de nombreux problèmes interviennent dans cette configuration. En effet, la nécessité de mettre en place une communication entre le poste informatique et l'API a pour effet d'induire un temps de latence entre l'apparition d'une évolution dans l'API et le traitement dans l'application développée. Ce temps, généralement supérieur à 10 fois le temps de cycle de l'API favorise les phénomènes gênants de synchronisation entre évolutions et le temps de latence entre l'évolution réelle sur la partie opérative et l'évolution du modèle de partie opérative. La quantité de données à échanger entre l'API et le poste informatique peut devenir importante lorsque les commandes et les capteurs présents sur la partie opérative considérée sont nombreux. Une proposition pour restreindre ces inconvénients est de mixer l'approche intégrée de la modélisation de la partie opérative avec le modèle de commande et l'approche externe à l'aide des technologies utilisée dans les outils de SCADA.

6.Conclusion

La démarche de modélisation présentée dans ce chapitre est principalement basée sur la prise en compte de la technologie physique des matériels constituant la partie opérative. La démarche propose un découpage en chaîne fonctionnelle et permet l'obtention pas-à-pas du modèle de partie opérative. La présentation de plusieurs modèles issus de matériels différents a permis de montrer que la démarche s'applique dans les cas génériques et qu'il faut retenir certaines hypothèses pour que le modèle soit réaliste. Les exemples traités ont montré comment les modèles de partie opérative peuvent être implantés sur le système réel avec des fonctionnalités diverses. Les limites rencontrées lors de l'implémentation rendent le modèle de partie opérative peu exploitable sur des systèmes industriels complexes. L'utilisation en ligne des modèles pose les problèmes de la gestion du temps, du traitement des évolutions simultanées et du traitement des données en temps réel.

La démarche de modélisation utilisée peut utiliser d'autres outils de modélisation répondant à des besoins différents. La simulation de comportement de la partie opérative peut exploiter cette méthodologie de modélisation avec un degré de finesse et un niveau d'abstraction différent de celui présenté pour la commande avec les automates à états finis. La partie opérative étant, par nature, un système continu, la représentation du comportement de la partie opérative par un modèle hybride combine les aspects discrets pour la représentation des commandes et des capteurs avec un modèle continu pour représenter le comportement des actionneurs. Le chapitre 4 est donc consacré à la modélisation de la partie opérative par une représentation hybride en vue de la simulation du comportement.

Chapitre 4
Modélisation de la partie opérative pour la simulation

Dans le chapitre précédent, nous avons pu voir que la modélisation de la partie opérative sous ses aspects discrets engendrait une vision restreinte des systèmes automatisés manufacturiers. Toutefois, la partie opérative est avant tout un système physique continu qui est perçu de façon discrète par la partie commande. La notion de discrétisation a pour effet de masquer le comportement global de la partie opérative pour ne distinguer qu'une quantité limité de situations différentes. Le temps fait également l'objet d'une discrétisation : la partie opérative n'est pas observée en permanence mais suit l'exécution cyclique de la partie commande. Au delà de la modélisation discrète vue précédemment, certaines applications nécessitent de prendre en compte le comportement de la partie opérative dans une modélisation combinant l'aspect continu physique à l'aspect discret du traitement effectué par la partie commande. Ce chapitre propose d'utiliser la démarche de modélisation abordée au chapitre 3 pour modéliser le comportement de la partie opérative sous des aspects hybrides. Les réseaux de Petri hybrides sont l'outil de modélisation retenu pour représenter le comportement de la partie opérative sous cette forme. Après avoir présenté l'outil de modélisation et intégrer la démarche de modélisation, des exemples de modélisation de la partie opérative sont proposés puis une application en simulation d'un système réel exploite la contribution.

1) Aspects continus et discrets de la modélisation

La commande d'un système automatisé de production manufacturière étant réalisée par les API dont le traitement des données impose une vision discrète, l'idée d'utiliser un modèle totalement discret peut sembler pertinente. Cependant, le comportement des équipements physiques s'apparente à un système continu et le système de commande à un système discret. Pour obtenir un modèle de partie opérative plus réaliste du comportement de la partie opérative, nous proposons une modélisation hybride. Celle-ci permet de prendre en considération le comportement continu des équipements physiques et les interconnexions discrètes de la commande.

L'API observe la partie opérative à travers les capteurs. L'ensemble des capteurs positionnés sur la partie opérative extrait des informations TOR de toutes les grandeurs physiques évoluant dans la partie opérative. Toutes ces informations permettent, par une forme de discrétisation, d'identifier un nombre fini de situations des grandeurs physiques.

La discrétisation n'est pas seulement liée aux grandeurs physiques de la partie opérative mais concerne également l'évolution temporelle. Le temps s'écoule continument vis-à-vis des grandeurs physiques de la partie opérative. L'API qui héberge la partie commande traite les informations cycliquement. Les informations contenues dans l'API ne sont donc pas toujours représentatives des grandeurs physiques réelles à un instant donné. L'actualisation cyclique de ces informations génère donc une discrétisation dans le temps des évolutions de la partie opérative telle qu'elle est perçue par la partie commande.

Bien qu'elle soit de nature continue, la partie opérative est perçue comme un système discret tant pour les grandeurs physiques mise en jeu que pour la gestion temporelle.

2) Présentation des réseaux de Petri hybrides et des capacités de modélisation

Les réseaux de Petri sont un ensemble d'outils de modélisation possédant des capacités de représentation plus importantes que les automates à états finis. Parmi toutes les variantes de réseaux de Petri, les réseaux de Petri hybrides permettent de modéliser les interactions pouvant avoir lieu dans un système constitué d'une partie continue et d'une partie discrète ayant des liens forts entre eux. Cet outil de modélisation est donc particulièrement approprié pour modéliser plus finement le comportement de la partie opérative au travers de la démarche de modélisation présentée précédemment. Les modèles peuvent ainsi être exploités dans les outils de simulation nécessitant une représentation fine du comportement de la partie opérative et notamment lorsqu'il s'agit d'avoir une représentation réaliste.

a) Modélisation discrète des réseaux de Petri hybrides

La partie discrète des réseaux de Petri hybrides est constituée de places liées par des transitions. Les places contiennent des jetons. On appelle marquage d'une place le nombre de jetons présents dans une place à un instant donné. Les transitions décrivent les évolutions possibles des jetons entre les places. Les transitions peuvent modifier le marquage de plusieurs places simultanément : les évolutions de marquage des places peuvent être synchronisées sur une même transition.

Les places sont représentées graphiquement par des cercles simples, des transitions représentées par des traits simples reliés aux places par des arcs orientés. La condition pour qu'une transition soit franchie est que les places amonts à la transition soient marquées par le nombre de jetons indiqué sur l'arc entrant de la transition. A défaut, on admet que si rien n'est indiqué, il est nécessaire d'avoir un jeton dans la place amont. Le franchissement de la transition est également conditionné par un événement extérieur. Après franchissement de la transition, les places avals sont incrémentées du nombre de jetons indiqués sur les arcs sortant

de la transition. Dans les Réseaux de Petri discrets autonomes, on admet que le franchissement de la transition est immédiat dès lors que toutes les conditions sont vérifiées.

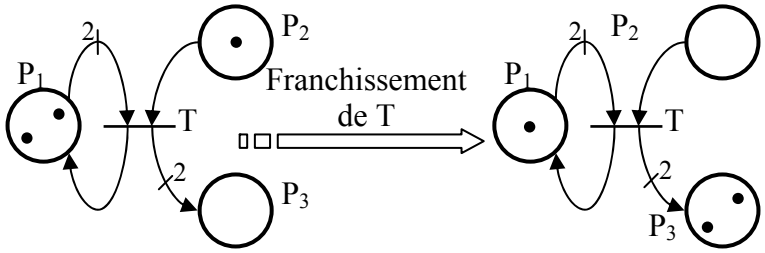


Figure 4.1 : franchissement d'une transition discrète

Dans l'exemple de la Figure 4.1, le franchissement de la transition T est conditionné par la présence de deux jetons dans la place P1 et d'un jeton dans la place P2. Au franchissement, la place P1 est décrémentée de deux jetons et la place P2 est décrémentée d'un jeton. La place P1 est alors incrémentée d'un jeton et la place P3 de deux jetons.

Dans le cadre dans lequel nous utilisons les réseaux de Petri hybrides, on admet que, dans la partie discrète, il n'y a qu'un seul jeton au même instant dans l'ensemble des marquages de toutes les places. Cette limitation de l'outil de modélisation permet d'affirmer que le réseau de Petri représente le même comportement qu'un automate à état fini pour la partie discrète.

b) Modélisation continue des réseaux de Petri hybrides

La partie continue des réseaux de Petri hybrides est constituée de places dont le marquage est représenté par une valeur réelle positive. Le franchissement des transitions continues est réalisé selon une loi indiquant la vitesse de franchissement de la transition en fonction du marquage continu des différentes places.

Les Réseaux de Petri continus sont composés de places représentées graphiquement par des cercles doubles, des transitions représentées par des traits doubles reliés aux places par des arcs orientés. La condition pour qu'une transition soit franchie est que toutes les places amont à la transition soient marquées par un nombre réel strictement positif. Des proportions de flux liés au franchissement peuvent être indiquées sur les arcs. A défaut, on admet que, si rien n'est indiqué, les places sont diminuées d'un rapport *1*. La vitesse de franchissement de la transition peut être conditionnée par une équation temporelle sur les valeurs des places amont. Pendant le franchissement progressif de la transition, les places amont sont proportionnellement diminuées avec les rapports indiqués sur les arcs entrant de la transition. Les places avals sont proportionnellement augmentées avec les rapports indiqués sur les arcs sortant de la transition. Si les transitions sont régies par des équations différentielles, le

franchissement de la transition est dépendant de cette équation. Il est cependant nécessaire de définir les conditions initiales sur les flux en plus du marquage initial.

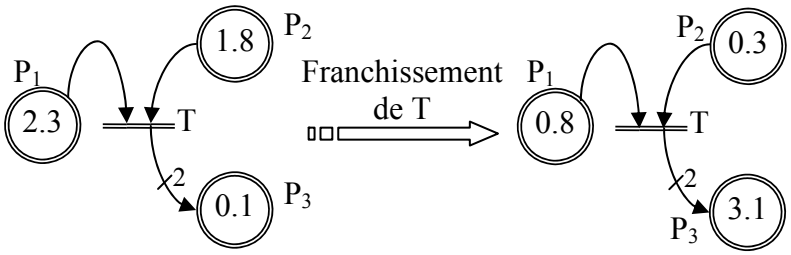


Figure 4.2 : franchissement d'une transition continue

Dans l'exemple de la Figure 4.2, le franchissement de la transition T est conditionné par la présence de marquage dans les places P1 et P2. Pour un franchissement de flux de *1.5*, le marquage de la place P1 est diminué de *1.5* et la place P2 de *1.5*. La place P3 est augmentée de *2 x 1.5*.

c) Synchronisation du comportement continu et discret des réseaux de Petri hybrides

A l'aide des parties discrètes et continues présentées précédemment, il est possible de prendre en compte dans la modélisation le comportement discret et continu d'un système. Néanmoins, il reste nécessaire de prendre en considération les interactions pouvant avoir lieu entre ces deux parties de modèle. Cette interaction peut être de deux types : la partie discrète du modèle peut restreindre ou accepter une évolution dans la partie continue et le modèle continu peut inhiber ou autoriser des évolutions dans la partie discrète du modèle hybride.

Les Réseaux de Petri hybrides reprennent la représentation graphique des Réseaux de Petri continus et discrets : les cercles doubles représentent les places continues et les cercles en trait simple les places discrètes. Les transitions discrètes sont représentées par des traits simples reliés aux places par des flèches. Il est également possible d'utiliser des transitions discrètes reliées à des places continues. Dans ce cas, le franchissement de la transition incrémente ou décrémente de la quantité indiquée sur l'arc respectivement les places avals ou amonts de façon discrète. L'incrémentation ou la décrémentation est discrète : le marquage des places continues réalise un saut de valeur.

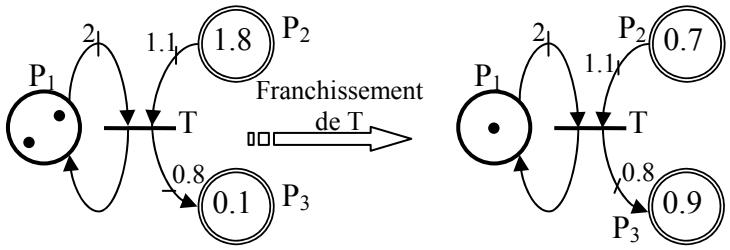


Figure 4.3 : franchissement d'une transition discrète dans un réseau de Petri hybride

Dans l'exemple de la Figure 4.3, le franchissement de la transition T est conditionné par la présence de deux jetons dans la place P1 et de la présence d'un marquage de la place P2 supérieur à la valeur indiquée sur l'arc amont. Au franchissement, la place P1 est décrémentée de deux jetons et la place continue P2 est décrémentée de la valeur indiquée sur l'arc. La place P1 est simultanément incrémentée d'un jeton et la place P3 est augmentée de la quantité indiquée sur l'arc aval.

Les transitions continues sont représentées par des traits doubles reliés aux places par des flèches. Il est possible d'utiliser des transitions continues reliées à des places discrètes. Cette notation est utilisée pour synchroniser la partie continue du modèle hybride d'un système avec la partie discrète. L'évolution de la partie continue est alors conditionnée par le marquage de la partie discrète. Un arc amont et aval relie la place discrète à la transition continue.

Dans l'exemple de la Figure 4.4, le franchissement de la transition continue depuis la place P2 vers la place P3 est conditionné par la présence d'un jeton discret dans la place P1. La transition ne peut pas être franchie s'il n'y a pas de jeton présent dans la place P1.

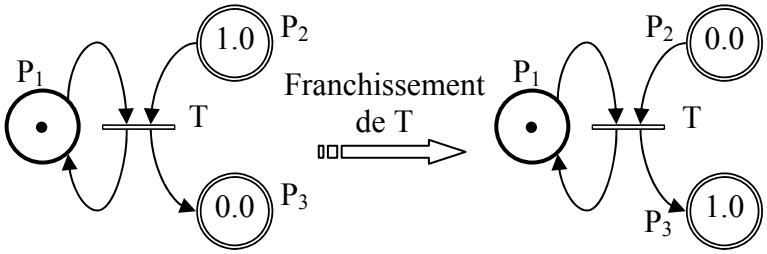


Figure 4.4 : franchissement d'une transition continue synchronisée par une place discrète

Pour représenter le comportement des systèmes hybrides, il faut également pouvoir synchroniser la partie discrète par le modèle continu. La notion d'arc inhibiteur permet d'interdire le franchissement d'une transition discrète en fonction du marquage d'une place continue. La transition n'est franchissable que si la condition associée à l'arc inhibiteur n'est pas vérifiée. Le marquage de la place continue n'est pas affecté par la présence d'arcs

inhibiteurs. L'arc inhibiteur est représenté par un arc orienté depuis une place continue pour aboutir à une transition où son extrémité est marquée par un cercle (Figure 4.5).

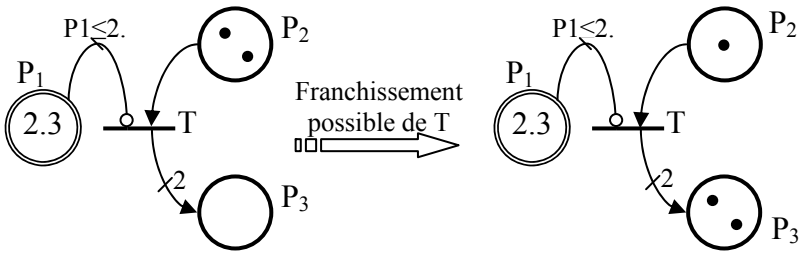


Figure 4.5 : franchissement d'une transition discrète synchronisée par une place continue à l'aide des arcs inhibiteurs

Sur l'exemple présenté Figure 4.5, le franchissement de la transition T peut être inhibé suivant le marquage de la place P1. Le marquage de la place P1 est supérieur à la valeur *2.0*. La condition P1≤*2.0* n'est donc pas vérifiée. La transition T peut donc être franchie. La place amont P2 est décrémentée d'un jeton et la place P3 est incrémentée de deux jetons.

La définition formelle de ces réseaux de Petri permet de les implémenter facilement pour les simuler. Nous proposons de modéliser le fonctionnement des actionneurs par une modélisation continue, le comportement des préactionneurs étant modélisé par la partie discrète. L'utilisation des Réseaux de Petri hybrides permet alors de synchroniser les modèles continus et discrets de ces différents éléments.

d) Application de la démarche de modélisation en utilisant les réseaux de Petri hybrides

La démarche de modélisation de la partie opérative avec les réseaux de Petri hybrides permet de prendre en considération certaines évolutions continues du comportement physique (Figure 4.6). La démarche est donc modifiée et contribue à améliorer le comportement physique continu de la partie opérative [Rohee, 2007b]. L'identification de la chaîne fonctionnelle est similaire : une chaîne fonctionnelle est associée à chaque actionneur.

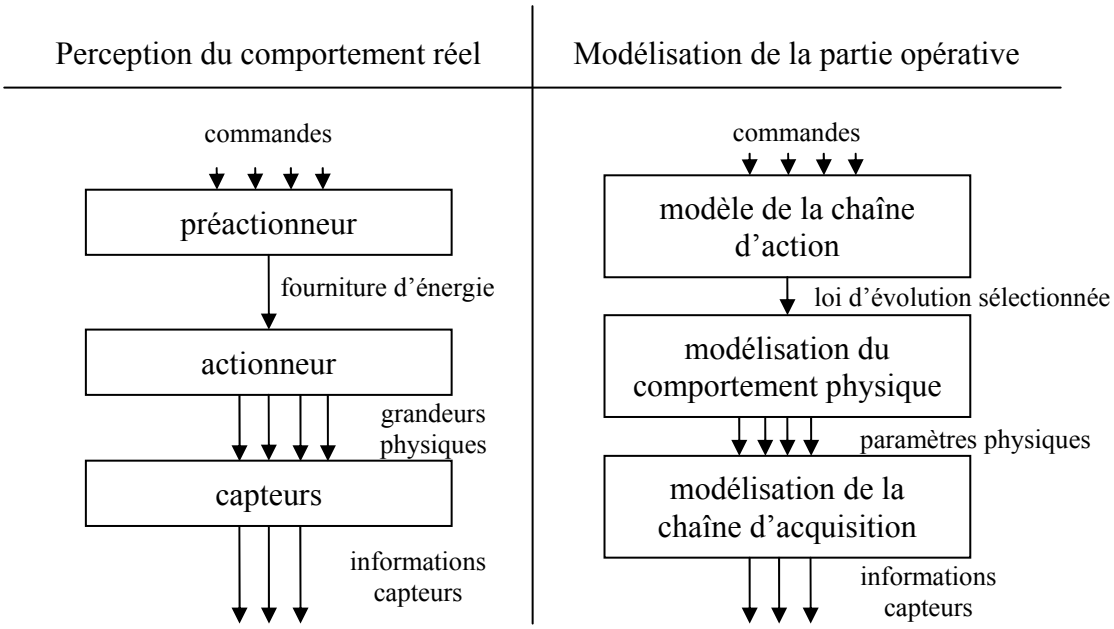


Figure 4.6 : comparaison entre la perception du réel et la modélisation

Les étapes de la modélisation sont représentées sur la Figure 4.7. La modélisation de la chaîne d'action détermine comment le système évolue. La combinaison des commandes permet de savoir quelle loi régit les évolutions du système. La modélisation du comportement du préactionneur permet d'établir quelle est l'énergie fournie à l'actionneur. Ces modes permettent d'établir les réactions de l'actionneur.

La modélisation du comportement physique établit comment évoluent les paramètres physiques continus représentant les grandeurs physiques dans la partie opérative réelle. Ce modèle décrit pour chaque loi d'évolution le comportement des paramètres physiques.

La modélisation de la chaîne d'acquisition permet d'établir la valeur des capteurs à partir des paramètres physiques. Le modèle des capteurs doit permettre de représenter l'association existant entre les grandeurs physiques et les mesures effectuées par les capteurs.

L'assemblage de ces trois modèles permet d'établir les états de capteurs en fonction des commandes tout en tenant compte des lois d'évolution qu'ont les grandeurs physiques présentes dans les systèmes réels.

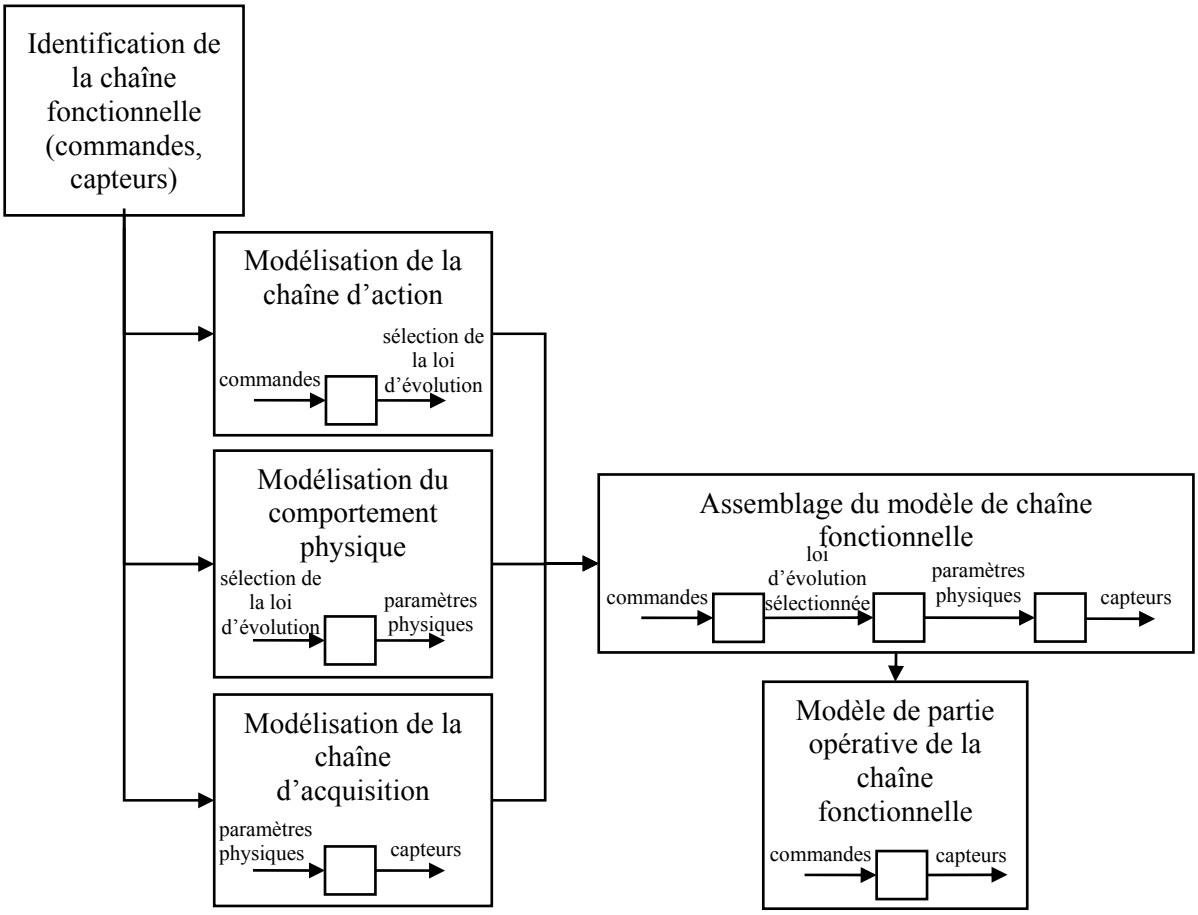


Figure 4.7 : adaptation de la démarche de modélisation aux systèmes hybrides

1) Modélisation de la chaîne d'action

Comme indiqué dans la modélisation discrète, le préactionneur fournit de l'énergie à l'actionneur en fonction des commandes issues de la partie commande. Les combinaisons des commandes permettent d'établir comment l'énergie est fournie à l'actionneur. Le modèle du préactionneur doit donc représenter les différents modes de fourniture d'énergie à l'actionneur en fonction des commandes. En admettant que le préactionneur prenne un nombre fini d'états, modéliser le comportement du préactionneur revient à déterminer quelles sont les lois d'évolution suivies par les grandeurs physiques pilotées par l'actionneur (Figure 4.8).

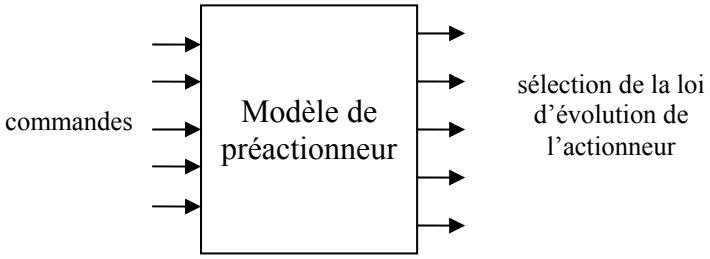


Figure 4.8 : positionnement du modèle de préactionneur

L'outil de modélisation pour représenter le comportement du préactionneur est le Réseau de Petri discret. Les changements de commande sont placés sur les arcs et les places et représentent les lois d'évolution de l'actionneur (Figure 4.9). La modélisation par les réseaux de Petri génère un modèle de préactionneur proche du modèle obtenu en utilisant les automates à états finis. Chaque place contient ainsi le mode de fourniture d'énergie.

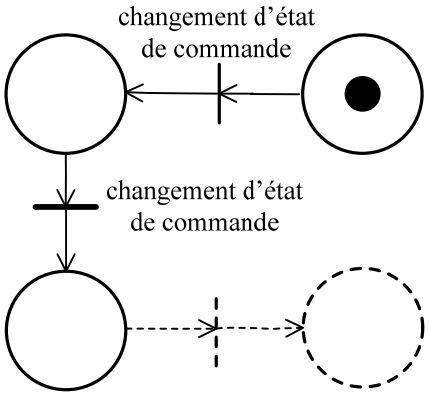


Figure 4.9 : modélisation d'un préactionneur

2) Modélisation de l'actionneur

La modélisation de l'actionneur détermine comment évoluent les paramètres physiques à partir de la loi d'évolution sélectionnée par le modèle de préactionneur (Figure 4.10). Ce modèle contient donc une liste de lois d'évolution continues décrivant le comportement de l'actionneur modélisé.

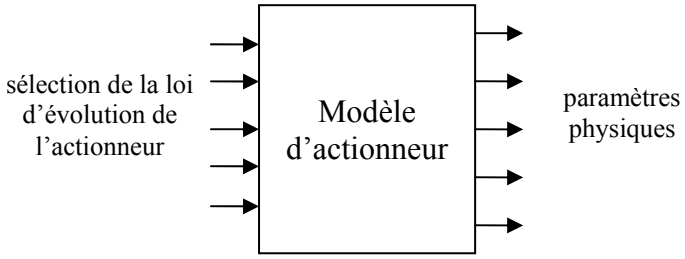


Figure 4.10 : positionnement du modèle d'actionneur

Les paramètres physiques contrôlés par l'actionneur sont représenté par des places continues (Figure 4.11). Ces paramètres physiques doivent nécessairement comporter une référence : à l'initialisation, les paramètres physiques doivent être synchronisés avec les grandeurs physiques initiales réelles de l'actionneur pour que le modèle soit cohérent.

Les transitions continues du réseau de Petri expriment le comportement des paramètres physiques suivant les lois d'évolution sélectionnées par le modèle du préactionneur. Les évolutions des paramètres physiques pour chaque loi d'évolution peuvent être modélisées et identifiées par un modèle continu linéaire du premier, du second ordre ou non linéaire en tenant compte des éventuelles perturbations extérieures. Il est nécessaire de synchroniser

l'état du préactionneur avec la loi d'évolution de l'actionneur : la présence d'un jeton dans une place du modèle de préactionneur conditionne le franchissement de la transition continue.

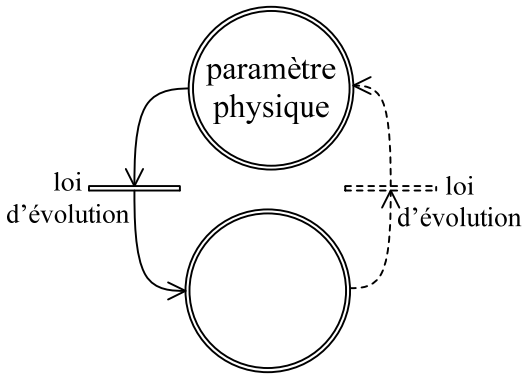


Figure 4.11 : modélisation de l'actionneur

Les lois de comportement utilisées doivent comporter les cas limites de l'évolution de l'actionneur : les fins de course d'objets en translation, les butées de vérins… installées sur le système réel doivent être prises en compte. Dans certains cas, suivant la définition du domaine de validité des paramètres physiques, il est possible de ne pas afficher ces limites (en exploitant la propriété de marquage positif des places des réseaux de Petri continus).

Les lois d'évolution liées aux transitions peuvent changer le sens du flux de la transition : un flux négatif indiqué sur une transition évoque un flux inversé. L'orientation du sens du flux nécessite de prendre beaucoup de précautions : si, graphiquement, il est plus aisé de comprendre le sens du flux positif dans le sens principal, les équations doivent traduire ce qui est représenté graphiquement.

3) Modélisation des capteurs

La modélisation des capteurs est réalisée indépendamment pour chaque capteur. Le comportement de chaque capteur est synchronisé par rapport aux paramètres physiques représentant la grandeur physique contrôlée par l'actionneur. Les changements d'états des capteurs dépendent de l'évolution de l'actionneur. Le modèle de capteurs est donc dépendant des paramètres physiques. Dans un système réel, les capteurs sont positionnés pour mesurer des plages de grandeurs physiques. Il est donc nécessaire de définir des seuils et des sens d'accostage dans le modèle des capteurs. Le changement d'état d'un capteur est conditionné par les franchissements de ces seuils. Il n'est pas possible de déterminer l'état du capteur uniquement en fonction de la situation de l'actionneur car les seuils d'activation et de désactivation ne sont pas obligatoirement les mêmes. Il peut exister un « effet d'hystérésis » sur le fonctionnement du capteur.

Les réseaux de Petri hybrides permettent de représenter le comportement des capteurs à l'aide des arcs inhibiteurs. Pour chaque capteur, deux places sont nécessaires pour représenter les états que le capteur TOR peut prendre. Les transitions correspondent aux changements d'état liés aux capteurs. Les conditions d'activation et de désactivation doivent cependant être

écrites de manière complémentées car les arcs inhibiteurs n'autorisent pas le franchissement d'une transition lorsque la condition associée est vraie. Les conditions d'activation et de désactivation sont écrites comme la combinaison entre un sens croissant ou décroissant du paramètre physique (front montant ou descendant) et d'une plage de validité dans lequel le front génère le changement d'état. Si le capteur doit être activé ou désactivé à plusieurs reprises dans le domaine de variation des paramètres physiques, les conditions d'activation et de désactivation sont assemblées par des fonctions OU logiques.

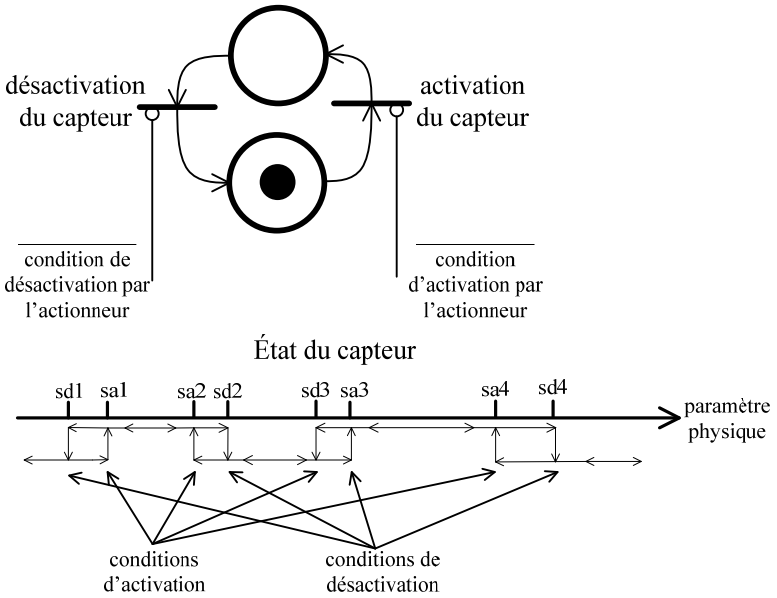


Figure 4.12 : modélisation des capteurs

Dans l'exemple de la Figure 4.12, le capteur TOR change d'état en fonction d'un paramètre physique. Le capteur change d'état à plusieurs reprises suivant le comportement de ce paramètre physique. Les effets d'hystérésis peuvent se produire lors des variations du paramètre physique et sont pris en compte par la dissociation du seuil d'activation et de désactivation des capteurs. Ainsi, quand la variable physique augmente, le passage à la valeur *sa1* permet une activation du capteur (passage à l'état haut). Lorsque le seuil « *sd2* » est atteint, le capteur est désactivé. Le seuil « *sa3* » permet à nouveau une activation du capteur tandis que le seuil « *sd4* » désactive le capteur. Quand le paramètre physique décroît, le seuil « *sa4* » et « *sa2* » permettent d'activer le capteur tandis que les seuils « *sd3* » et « *sd1* » désactive le capteur. La différenciation des seuils entre l'activation et la désactivation du capteur vis-à-vis du paramètre physique permet ainsi de prendre en compte la mémorisation pouvant avoir lieu lors des évolutions du paramètre physique. Cette étude permet d'écrire les équations décrivant les changements de capteurs sur les arcs inhibiteurs. Celles-ci sont constituées du sens de variation de la variable interne et de la valeur de la variable.

$$\text{condition d'activation du capteur} = \left(\frac{\mathrm{d}Var}{\mathrm{d}t} > 0\right).\left(Var = sa1 \vee sa3\right) + \left(\frac{\mathrm{d}Var}{\mathrm{d}t} < 0\right).\left(Var = sa2 \vee sa4\right)$$

et

$$\text{condition de désactivation du capteur} = \left(\frac{\mathrm{d}Var}{\mathrm{d}t} > 0\right).\left(Var = sd2 \vee sd4\right) + \left(\frac{\mathrm{d}Var}{\mathrm{d}t} < 0\right).\left(Var = sd1 \vee sd3\right)$$

3) Exemple de modèles de chaîne fonctionnelle par les réseaux de Petri hybrides

Pour illustrer la démarche de modélisation par les réseaux de Petri hybrides, quatre exemples de modèles de partie opérative de chaînes fonctionnelles sont présentés. Les modèles proposés exploitent des équipements matériels qui sont mis en œuvre dans les chaînes fonctionnelles composant les systèmes réels. Ces modèles ont également pour objectifs d'illustrer comment sont réalisées les connexions entre les modèles de la chaîne d'action, le modèle du comportement physique et le modèle de la chaîne d'acquisition.

a) Modèle d'un vérin pneumatique avec un distributeur 3/2 et un capteur de fin de course

Un vérin simple effet équipé d'un distributeur à trois orifices et deux positions et d'un capteur de fin de course est le premier exemple choisi pour être modélisé par un réseau de Petri (Figure 3.8).

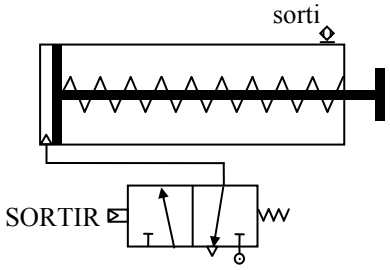


Figure 4.13 : chaîne fonctionnelle « vérin pneumatique avec distributeur monostable et un capteur de fin de course »

L'établissement du modèle de la chaîne d'action est réalisé par la modélisation du préactionneur. Le distributeur présent dans cette chaîne fonctionnelle peut prendre deux positions distinctes. A chacune de ces positions correspond un mode d'alimentation en énergie de l'actionneur. Les situations transitoires et le temps de passage d'une position du distributeur à l'autre sont négligés dans le niveau de modélisation choisi. L'état du préactionneur est une fonction combinatoire des entrées. Le préactionneur est donc modélisé par un réseau de Petri discret à deux états (Figure 4.14).

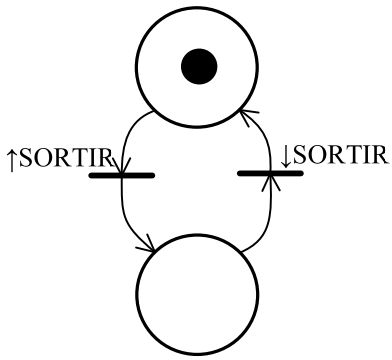


Figure 4.14 : modélisation du préactionneur

L'actionneur est modélisé par un réseau de Petri continu : la position de la tige par rapport à chacune de ses extrémités est associée à un paramètre physique représenté par la variable réelle *position* comprise entre 0 et 1. La variable est initialisée à *0.0* traduisant que le vérin est rentré à la mise en fonctionnement. Les changements de ce paramètre sont rendus possibles par le franchissement des transitions continues conditionnées par le modèle de préactionneur (Figure 4.15). Les lois d'évolution régissent le franchissement de ces transitions au cours du temps. Ces lois d'évolution constituent le modèle de comportement physique. Le comportement choisi pour représenter les franchissements des transitions traduit un modèle linéaire de franchissement.

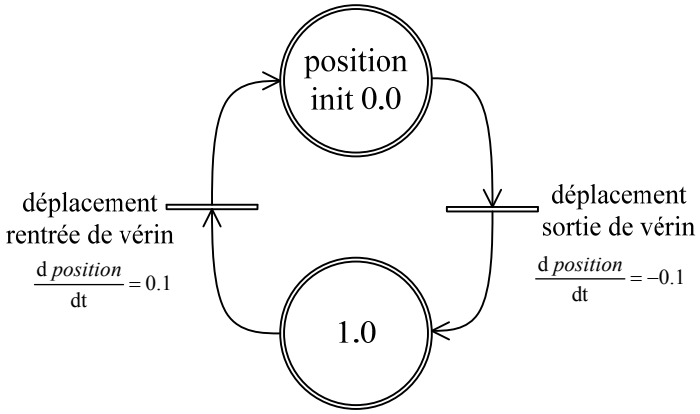


Figure 4.15 : modélisation de l'actionneur

La modélisation de la chaîne d'acquisition est réalisée par la modélisation du capteur positionné à l'extrémité de la course du vérin. Les changements d'états de ces capteurs sont conditionnés par la position de la tige du vérin et le sens d'accostage du capteur. On modélise avec un arc inhibiteur l'autorisation de changement d'état du capteur pour le synchroniser avec le modèle de comportement physique. Le capteur est positionné sur la course du vérin et détecte une plage de positions. Ainsi, le capteur est supposé ne pas avoir d'effet d'hystérésis mais celui-ci détecte la sortie du vérin lorsqu'il est effectivement sorti à 95% de la course totale (Figure 4.16).

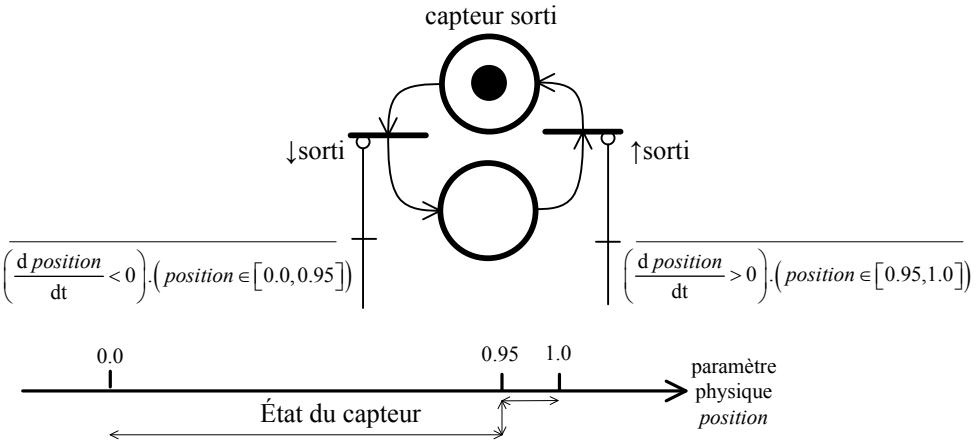


Figure 4.16 : modélisation du capteur

Les modèles de préactionneur, d'actionneur et de capteurs peuvent être assemblés pour former le modèle de partie opérative de la chaîne fonctionnelle (Figure 4.17). Le modèle du préactionneur conditionne le franchissement des transitions par la présence du jeton dans la place considérée. Les arcs inhibiteurs du modèle de capteurs sont associés à la place continue contenant la variable continue *position*.

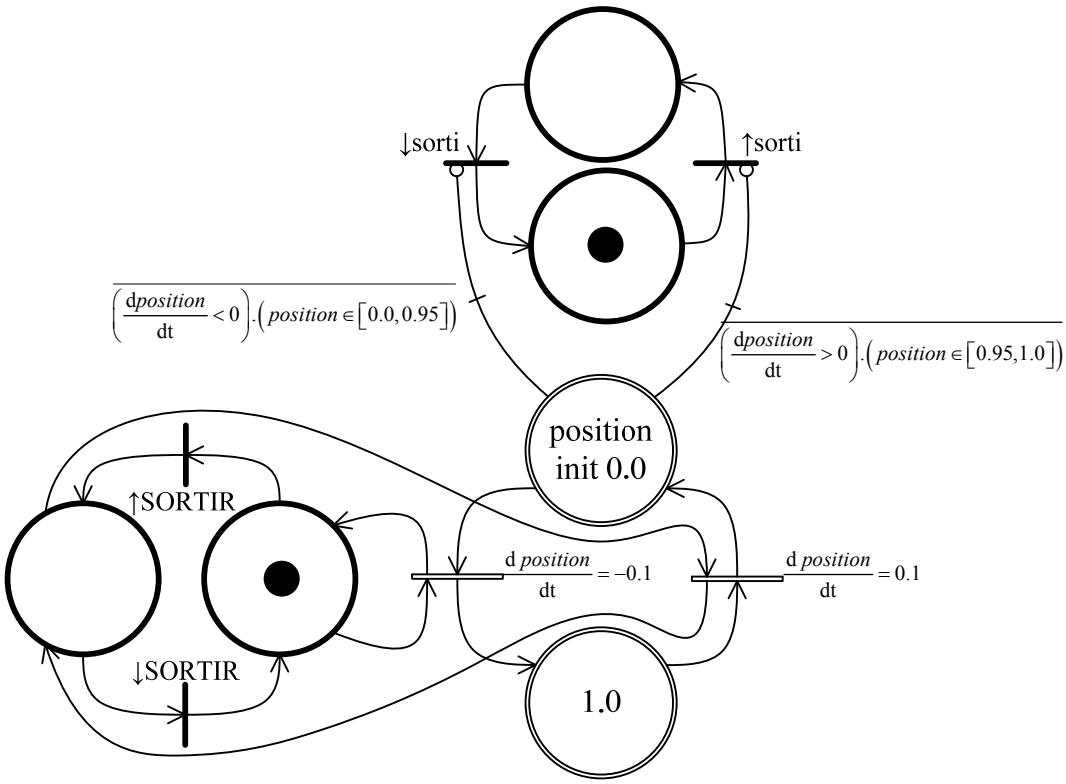


Figure 4.17 : modèle de partie opérative de la chaîne fonctionnelle

b) Modèle d'un vérin pneumatique avec un distributeur 3/2 et deux capteurs de fin de course

Un vérin équipé d'un distributeur à trois orifices et deux positions et de deux capteurs de fin de course est le second exemple choisi (Figure 4.18). Les capteurs utilisés sont différents : le capteur *sorti* est un capteur de proximité tandis que le capteur *rentré* est un détecteur mécanique à galet présentant des seuils d'activation et de désactivation différents générant ainsi un effet d'hystérésis.

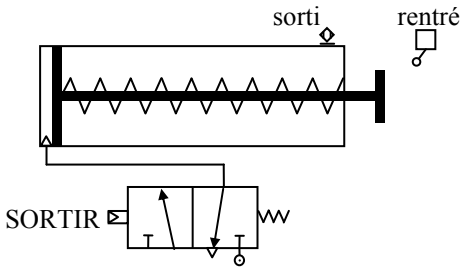


Figure 4.18 : chaîne fonctionnelle « vérin pneumatique avec distributeur monostable et deux capteurs de fin de course »

Le distributeur et l'actionneur sont similaires à ceux de la chaîne fonctionnelle de l'exemple précédent. Les modèles de la chaîne d'action et du comportement physique sont donc conservés.

Le modèle de la chaîne d'acquisition est différent car le capteur mécanique *rentré* qui a été ajouté doit être pris en compte. Le modèle du capteur *sorti* est similaire au précédent.

Le modèle du capteur *rentré* est composé de deux places discrètes dont les transitions sont inhibées par les évolutions du paramètre physique *position* à l'aide des arcs inhibiteurs. Le phénomène d'hystérésis est présent sur ce capteur : le seuil d'activation est fixé à 5% de la course du vérin tandis que le seuil de désactivation est fixé à 10%. Les conditions indiquées sur les arcs inhibiteurs prennent donc en compte ces paramètres pour que le comportement des capteurs soit plus représentatif (Figure 4.19).

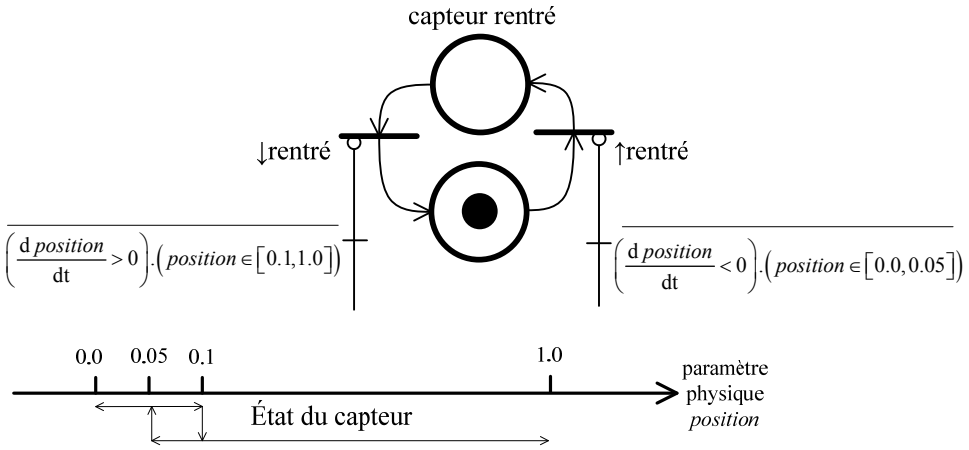


Figure 4.19 : modélisation du capteur *rentré*

Le modèle de partie opérative est ensuite assemblé à partir des différents modèles établis précédemment (Figure 4.20).

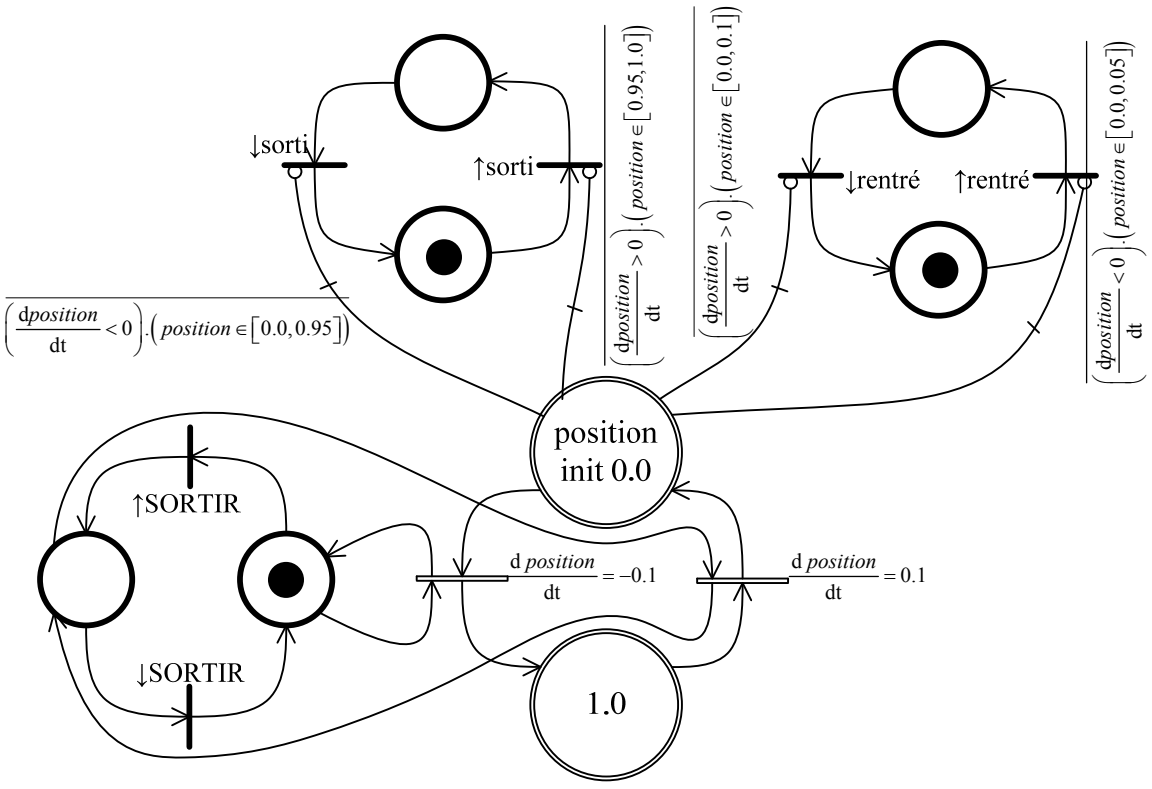


Figure 4.20 : modèle de partie opérative de la chaîne fonctionnelle

c) Modèle d'un vérin pneumatique avec un distributeur 5/2 et d'un capteur de fin de course

Un vérin équipé d'un distributeur à cinq orifices et deux positions et d'un capteur de fin de course est le troisième exemple choisi (Figure 4.21).

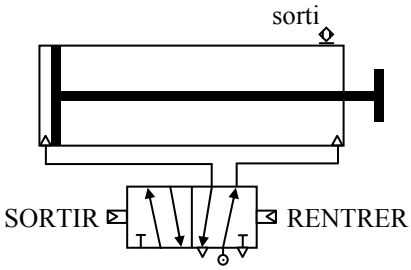


Figure 4.21 : chaîne fonctionnelle « vérin pneumatique avec distributeur bistable et un capteur de fin de course »

Le modèle de la chaîne d'action est obtenu à travers la modélisation comportementale du distributeur. Le modèle du distributeur en réseau de Petri discret comporte six places discrètes permettant de sélectionner la loi d'évolution à suivre. Le modèle est similaire au modèle obtenu par la modélisation avec les automates à états finis pour ce distributeur (Figure 4.22).

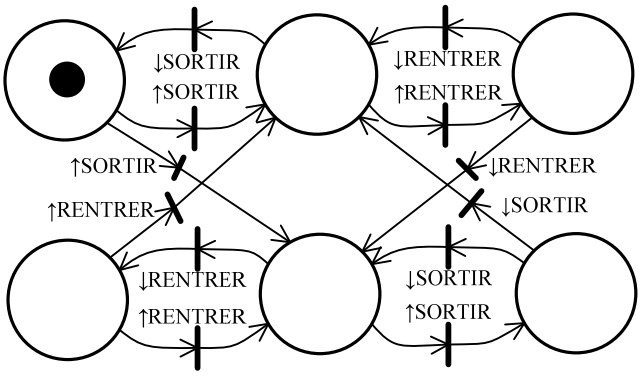


Figure 4.22 : modélisation du préactionneur

L'actionneur utilisé n'a pas la même vitesse de déplacement dans les deux sens. Le modèle d'actionneur doit refléter ce comportement. Le modèle indiqué sur la Figure 4.23 comporte deux places continues. Les transitions décrivant les lois d'évolution de l'actionneur n'ont pas le même comportement suivant le sens de déplacement : la vitesse de franchissement indiquée diffère d'une transition à l'autre.

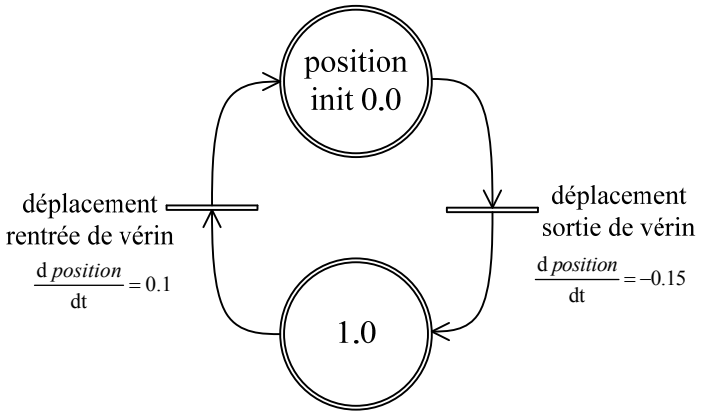


Figure 4.23 : modélisation de l'actionneur

La modélisation de la chaîne d'acquisition est réalisée par la modélisation du capteur positionné à l'extrémité de la course du vérin. Le modèle utilisé pour ce capteur a déjà été présenté dans le premier exemple. L'ensemble du modèle de comportement de la partie opérative considéré peut être construit à partir des modèles établis. Ce modèle n'est pas représenté à cause de la complexité inhérente au modèle. Le modèle de la Figure 4.23 montre trois états du modèle du préactionneur et les transitions associées ainsi que le modèle d'actionneur et trois transitions en relation avec les états du modèle du préactionneur. Les transitions du modèle d'actionneur peuvent être dupliquées car une transition doit être définie pour chaque état du modèle de préactionneur. Le modèle de partie opérative ne comporte pas la modélisation du capteur.

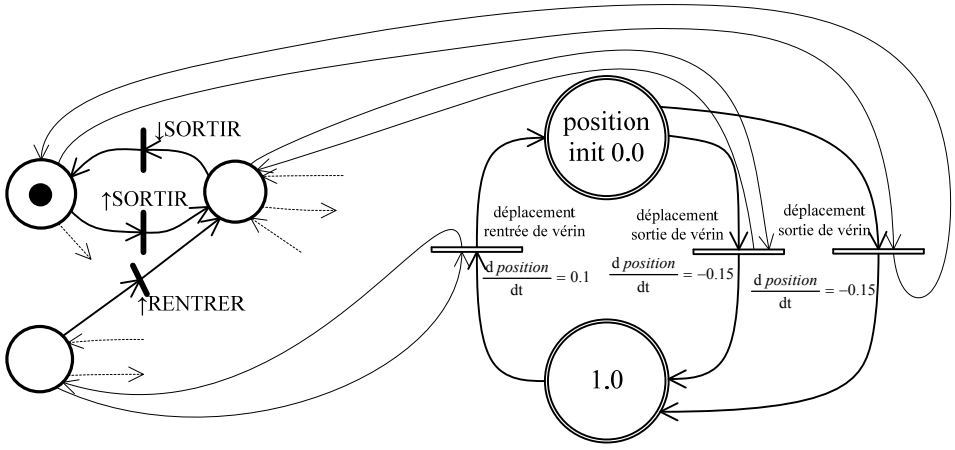


Figure 4.24 : modélisation partielle de partie opérative

d) Modèle d'un vérin pneumatique avec un distributeur 5/3 et un capteur intermédiaire

Un vérin équipé d'un distributeur à cinq orifices et trois positions à centre fermé et d'un capteur au milieu de la course est le dernier exemple (Figure 4.25).

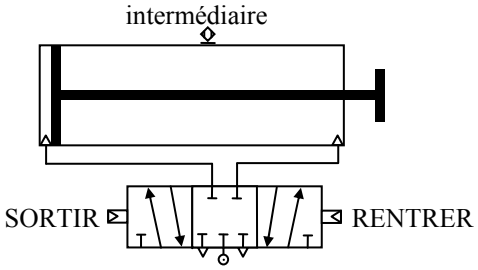


Figure 4.25 : chaîne fonctionnelle « vérin pneumatique avec distributeur à trois positions et un capteur intermédiaire »

Le modèle de la chaîne d'action est obtenu par la modélisation comportementale du distributeur. Le modèle du distributeur en réseau de Petri comporte quatre places discrètes permettant de sélectionner la loi d'évolution à suivre et les transitions illustrant les changements de commande.

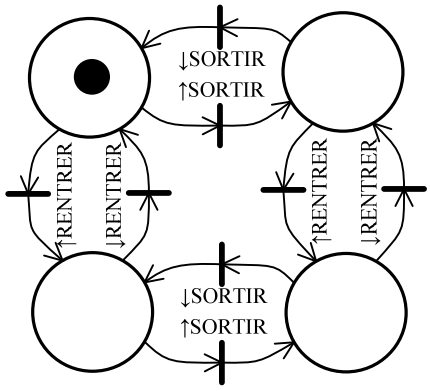


Figure 4.26 : modélisation du préactionneur

L'actionneur est alimenté suivant trois modes différents : le mode de « sortir de la tige du vérin », le mode « rentrer » et le mode de blocage. Lorsque le distributeur se trouve dans l'état central, le vérin ne peut se déplacer. Le modèle d'actionneur comporte donc trois transitions différentes pouvant être associées pour chacune d'elle avec un état du modèle de préactionneur (Figure 4.27). La transition représentant le blocage peut néanmoins être retirée du modèle car l'absence de transition est équivalente à une transition ne laissant pas passer de flux.

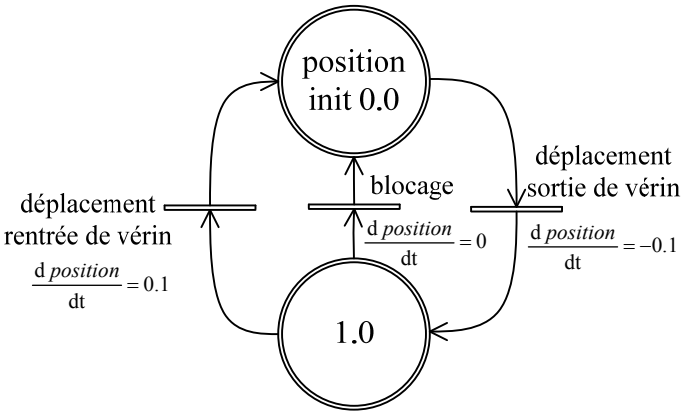


Figure 4.27 : modélisation de l'actionneur

Le modèle de la chaîne d'acquisition ne comprend que la modélisation du capteur « *intermédiaire* ». Le modèle du capteur « *intermédiaire* » est composé de deux places discrètes dont les transitions sont inhibées par les évolutions du paramètre physique *position* à l'aide des arcs inhibiteurs. On admet que le capteur ne comporte pas d'effet d'hystérésis et que le seuil d'activation du capteur est compris entre 45% et 55% de la course totale du vérin (Figure 4.28). Chaque arc inhibiteur contient donc la condition de changement d'état du capteur dans les deux sens du mouvement.

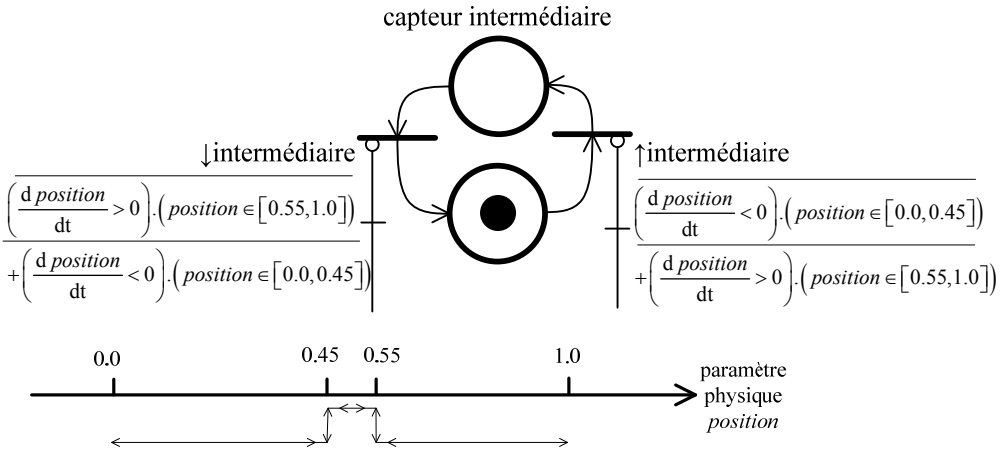


Figure 4.28 : modélisation du capteur *intermédiaire*

Le modèle de partie opérative de la chaîne fonctionnelle est obtenu par l'assemblage des modèles de préactionneur, d'actionneur et du capteur considérés (Figure 4.29). Les transitions de blocage ont été supprimées puisqu'elles ne sont pas indispensables.

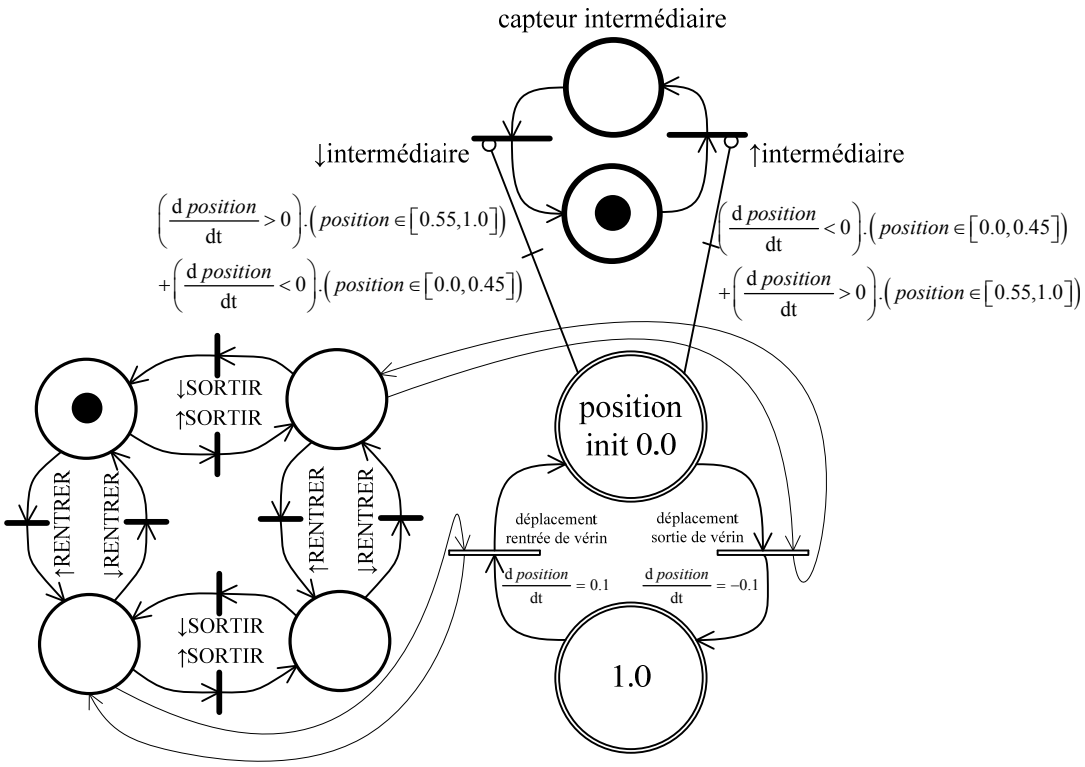


Figure 4.29 : modélisation de la chaîne fonctionnelle

4) Application à la simulation des chaînes fonctionnelles

Pour illustrer l'intérêt de la modélisation hybride par les chaînes fonctionnelles, deux exemples sont traités. Le premier exemple est consacré à la modélisation d'un axe numérique utilisé dans les établissements d'enseignement. Le second exemple traite la modélisation d'une porte de garage automatisée.

a) Modélisation d'un axe numérique

Le système considéré est constitué d'un chariot en translation sur un axe (Figure 4 30). Le système est tout d'abord modélisé à l'aide de la démarche en utilisant les réseaux de Petri hybrides. Le modèle est ensuite implanté dans un outil de simulation WINSIM [Riera, 2001] associé à un logiciel de supervision industrielle Panorama pour simuler le comportement du système réel.

Figure 4 30 : banc Emericc représentant un chariot en translation

Le banc Emericc est un système didactique consacré à l'étude de l'asservissement en position et en vitesse sur un axe numérique avec l'étude des facteurs dynamiques tels que l'inertie. La simulation ne concerne pas toutes les fonctionnalités du banc mais uniquement la représentation comportementale du déplacement en fonction des commandes et les évolutions des capteurs de fin de course et du capteur intermédiaire. Deux commandes permettent de piloter les relais d'alimentation électrique du moteur déplaçant le chariot avec une commande bistable. Les capteurs commutent au niveau haut lorsque le chariot atteint les positions extrêmes. La représentation du chariot dans le logiciel de supervision est réalisée par l'image d'un véhicule se déplaçant dans un intervalle donné.

Le mouvement du chariot a été modélisé en vitesse par un système du premier ordre (prise en compte de l'inertie dans le modèle). Pour implanter le modèle dans l'outil de simulation WINSIM, une discrétisation temporelle et de la position sont effectuées : les positions discrétisées ne sont pas calculées sur une grandeur temporelle continue mais à une période d'échantillonnage fixe. Le véhicule représenté sur l'interface se déplace donc de manière discrète, les incréments étant infinitésimaux. Il est nécessaire de fournir une plage d'activation des capteurs et non plus une valeur d'incrémentation car la discrétisation empêche d'atteindre la valeur de façon continue. Le comportement du chariot lorsqu'il atteint les butées mécaniques de fin de course est d'arrêter le mouvement instantanément. Les contraintes dues aux efforts mécaniques du chariot sur la butée de fin de course et les phénomènes de surchauffe dus au blocage du moteur ne sont pas pris en compte dans le modèle. La vitesse de déplacement du véhicule est calculée par le calcul de la dérivée de la position.

Le modèle de comportement a été implanté dans le logiciel de simulation et le logiciel de supervision Panorama pour visualiser le comportement du modèle implémenté au travers une interface graphique (Figure 4.31). La course de déplacement du véhicule est représentée par un déplacement de gauche à droite de l'image du véhicule sur l'interface graphique. Les commandes sont représentées à l'aide des boutons fléchés en haut. Les trois capteurs sont représentés par des voyants pouvant être éteints ou allumés. Un compteur de vitesse permet de suivre la vitesse de déplacement du véhicule. La partie basse de la figure affiche un graphique « au fil de l'eau » représentant la position du véhicule sur la course possible de mouvement et la vitesse correspondant à la dérivée de la position du véhicule au cours du temps.

Lorsque le modèle est utilisé, on s'aperçoit que le comportement du préactionneur est bien pris en compte suivant les commandes demandées. Ainsi, lorsque les commandes « *DROITE* » et « *GAUCHE* » sont actives en même temps le véhicule est arrêté. Lorsque le véhicule se déplace, la vitesse de déplacement répond à une sollicitation d'un système du premier ordre. Le choix des paramètres de comportement a été réalisé avec la même dynamique et donc les mêmes valeurs de temps de réponse quelque soit le sens de déplacement du véhicule. Le déplacement du véhicule suit donc une loi de comportement du second ordre. Dans l'exemple de la figure, le véhicule a atteint la position extrême à droite avant de se déplacer vers la gauche par des petits déplacements successifs.

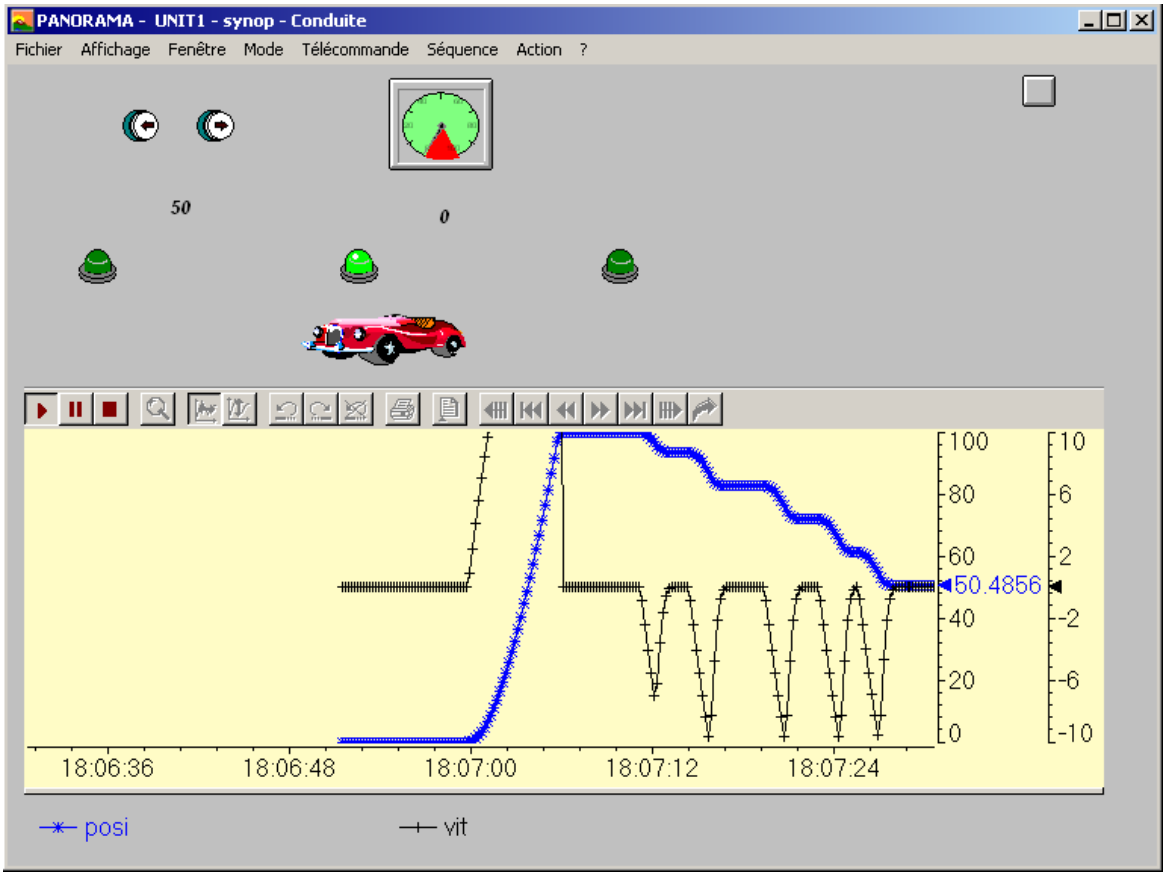


Figure 4.31 : interface graphique illustrant le déplacement du véhicule

Cet exemple illustre donc d'un point de vue simulation le comportement physique d'un système en tenant compte des paramètres d'inertie. L'utilisation d'un outil de modélisation formel permet entre autre de faciliter la conception de l'application de simulation grâce à la réutilisation de composants définis une seule fois et paramétrés. La représentation permet également de s'assurer que toutes les évolutions des commandes peuvent se produire et qu'elles sont prises en considération dans le modèle.

La modélisation de l'actionneur avec la simulation du comportement des paramètres physiques permet ainsi d'obtenir un simulateur de partie opérative ou un modèle de comportement de référence à comparer avec l'évolution réelle du système pour détecter les incohérences de comportement du système réel. L'axe numérique n'est pas un système

complet : il est souvent utilisé dans un environnement industriel pour gérer un asservissement en position et en vitesse.

b) Modélisation de la porte de garage automatisée

Le second exemple de système modélisé par les réseaux de Petri hybrides de ce chapitre est consacré à la simulation d'un système didactique classiquement utilisé dans la formation aux automatismes.

1) Contexte des maquettes virtuelles

Depuis plusieurs années le concept de maquettes virtuelles pour l'enseignement de l'automatique a été développé entre autres par des chercheurs du CReSTIC [Riera, 1999a, 1999b, 2001, 2005, 2008]. Les différences fondamentales entre une maquette virtuelle et une simulation classique sont :

- un rendu réaliste de la structure, des fonctions et du comportement du système,
- le respect de la dynamique du système.

La manipulation de la maquette virtuelle par l'étudiant doit être similaire à la maquette réelle. Le rendu visuel et sonore allié au respect de la dynamique doit favoriser l'interactivité avec l'étudiant tout en permettant un passage efficace entre la théorie et la pratique. La maquette virtuelle peut donc être utilisée à tous les niveaux (cours, TD, TP, auto-formation...) et s'intègre parfaitement dans l'utilisation des Nouvelles Technologies Educatives (NTE).

Il est clair que la maquette réelle présente des avantages d'un point de vue pédagogique. Une maquette réelle est une reproduction physique en miniature d'un système réel. Elle impose aux apprenants de la manipuler comme ils le feraient dans une activité professionnelle. En revanche, la maquette réelle présente aussi des inconvénients. Son coût est souvent élevé et elle nécessite une maintenance avec du personnel qualifié. De plus, elle est soumise à rude épreuve et, de ce fait, n'est pas toujours disponible compte tenu d'un manque de fiabilité. Enfin, il est rarement possible de simuler volontairement des défaillances (pannes de capteurs ou d'actionneurs par exemple). Les problématiques sont également souvent simplifiées par rapport aux systèmes industriels par l'utilisation de systèmes d'envergure restreinte. De ce fait, les expérimentations possibles sont souvent réduites à une initiation aux problèmes d'automatisme. De plus, les séances de travaux pratiques avec des maquettes réelles nécessitent pour les enseignants des temps de préparation importants, des coûts élevés et des configurations le plus souvent figées. Enfin, pour les étudiants, la maquette réelle permet de valider seulement partiellement les acquis théoriques car elle offre peu de liberté et est accessible uniquement pendant les séances de TP.

La maquette virtuelle apporte une complémentarité à la maquette réelle. En effet, il devient possible, pour les enseignants entre autres, de créer à moindre coût des TP « clés en main » aux nombreuses fonctionnalités, de tester avant d'implémenter, de faire des TD interactifs

mais aussi de permettre de faire du TP à distance. Pour les étudiants, la maquette virtuelle est attrayante car réaliste mais aussi ludique. Elle est facile à utiliser même en auto-formation et ne présente pas de risque. Par conséquent, la maquette virtuelle va permettre de valider les acquis théoriques avec plus de souplesse. En revanche, la maquette réelle va développer le sens physique de l'étudiant et l'appréhension du danger face aux risques mécaniques. Par conséquent, l'approche que nous préconisons consiste à ne pas remplacer la maquette réelle mais à la rentabiliser, d'un point de vue pédagogique, au maximum. Des étudiants bien préparés à une séance de TP par exemple, doivent en principe en retirer des bénéfices tant du point de vue théorique que personnel. Enfin, il est évident qu'il est possible de créer des TP utilisant exclusivement des maquettes virtuelles connectées à une partie commande qui peut être simulée ou non. Il est certain que cela n'est pas sans poser des problèmes pédagogiques car la simulation et le virtuel ne peuvent pas remplacer totalement la pratique sur des systèmes physiques réels. En revanche, il est certain que la maquette virtuelle présente des avantages en termes de coût, de disponibilité et de fonctionnalités (simulation des pannes par exemple).

Les maquettes virtuelles développées au CReSTIC reposent sur l'utilisation d'un moteur de calcul « auteur » WINSIM [Riera, 2001] et des progiciels de supervision (INTOUCH, PANORAMA, …). En effet, une maquette virtuelle s'apparente en quelque sorte à une supervision parfaite reprenant fidèlement les fonctions, la structure et le comportement du système. Les maquettes virtuelles peuvent se connecter à une partie commande réelle de type API au moyen d'un serveur OPC. Pour cela, on utilise le fait que tous les SCADA sont clients OPC. La Figure 4.32 illustre ce principe de communication. Le module de simulation (WINSIM dans notre cas) communique avec le logiciel de supervision grâce à une communication directe (de type DLL) ou un lien DDE (Dynamic Data Exchange) car WINSIM est serveur DDE. Le moteur de simulation, à partir de l'état des actionneurs (*Ssu1*) détermine l'état des capteurs (*Esu1*). En complément de ces informations, le module de simulation fournit aussi l'état de variables d'état permettant une visualisation de la Partie Opérative simulée sur une Interface Homme-Machine (IHM). Le SCADA assure aussi l'échange de données entre l'API et le moteur de simulation (*Ssu1=Esu2* et *Ssu2=Esu1*).

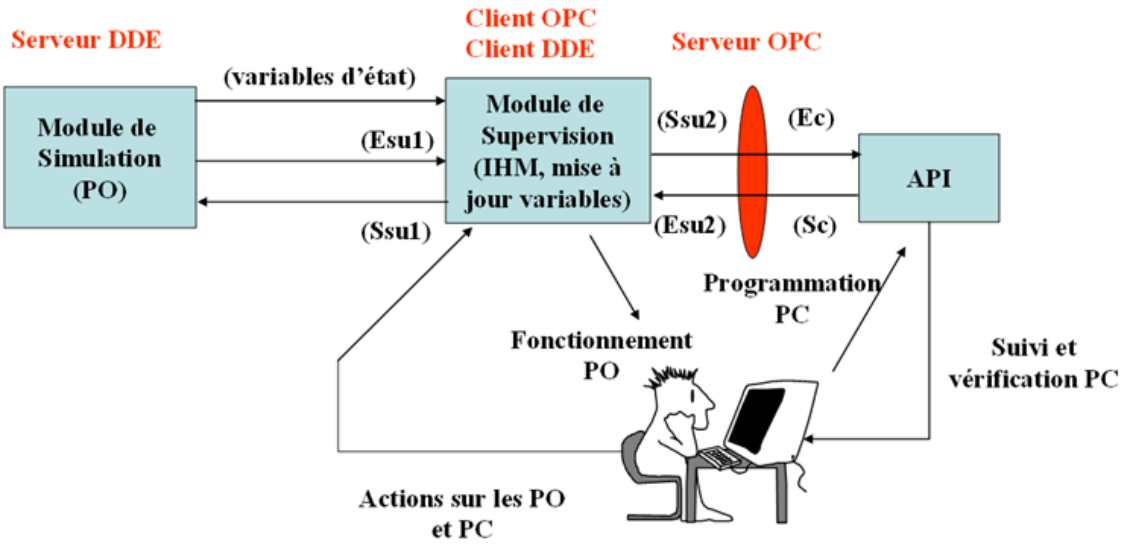


Figure 4.32 : principe de communication

La Figure 4.33 montre un exemple de maquette virtuelle représentant une porte de garage. Nous avons travaillé sur l'amélioration du modèle dynamique de ce système en utilisant les travaux présentés dans ce chapitre.

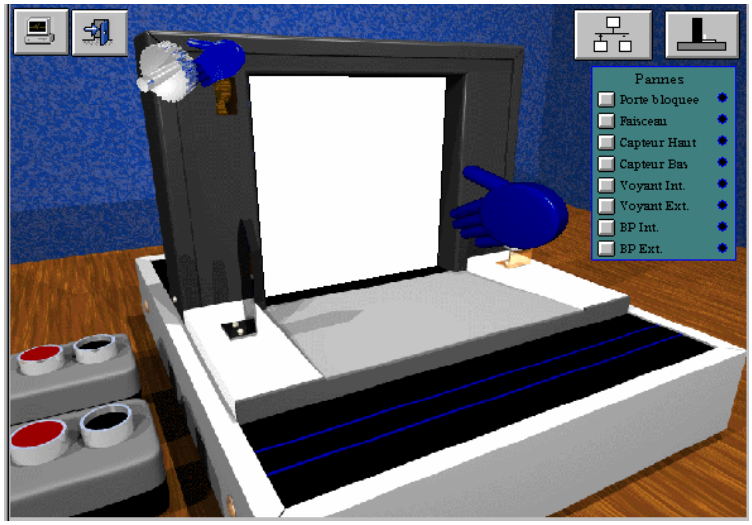


Figure 4.33 : maquette virtuelle de la porte de garage

Ces maquettes virtuelles ont été utilisées entre autres pour faire du télé-TP [Riera, 2003b]. L'idée de base est de permettre aux étudiants de profiter pleinement de la séance de TP en arrivant bien préparés. D'une façon très générale, une séance de TP de programmation d'API vise à transmettre aux étudiants un savoir-faire concernant les outils et les langages de programmation des automates mais surtout la possibilité d'appliquer les concepts théoriques vus en cours pour aboutir à une partie commande compatible avec le cahier des charges. Les étudiants sont amenés commettre des erreurs pour pouvoir assimiler les différentes facettes d'un principe théorique. Malheureusement, par manque de temps, la séance de TP permet

souvent uniquement à l'étudiant d'acquérir un savoir-faire partiel. Une expérimentation avait été réalisée avec pour objectif de mettre en évidence s'il est possible ou non à l'étudiant d'acquérir une partie du savoir-faire pendant une séance de télé-TP préalable en vue de se consacrer principalement aux problèmes de « fond » pendant la séance en salle de TP. Le dispositif technique retenu est présenté Figure 4.34. Trois salles distinctes ont été employées et instrumentées de manière spécifique :

• une salle de TP déjà existante, contenant le matériel d'automatisme nécessaire (API, maquettes réelle…),

• une salle de travail pour l'étudiant, lui permettant d'accéder à l'API distant via le réseau Intranet de l'Université

• une salle de travail pour l'enseignant, lui permettant de prendre la main sur le PC de l'étudiant et le PC de la salle de TP pour pouvoir éventuellement réaliser des opérations interdites aux étudiants.

La technologie VNC (Virtual Network Computing) est utilisée pour observer et piloter à distance un poste informatique à partir d'un poste distant. Le poste serveur VNC est le poste dont le contrôle peut être pris par un poste client VNC.

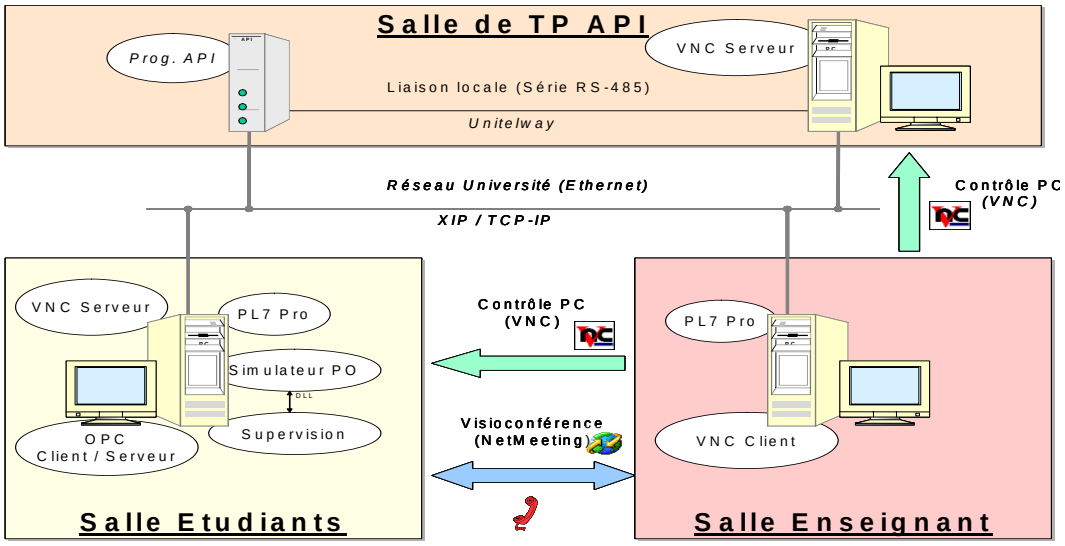


Figure 4.34 : dispositif technique

Les résultats obtenus ont été globalement satisfaisants, les étudiants ont été considérés comme bien préparés, même si le dispositif développé s'avère complexe. Un point très intéressant mérite d'être souligné quant au choix du système automatisé utilisé. La maquette virtuelle développée reprend la maquette réelle de porte de garage. Toutefois, la simulation de la partie opérative avait été simplifiée et l'inertie du moteur n'était initialement pas prise en compte. Comme cela va être montré dans le paragraphe suivant, les phénomènes d'inertie peuvent conduire à des problèmes de commande intéressants d'un point de vue pédagogique, et les étudiants n'en tiennent pas suffisamment compte dans l'élaboration de leur commande.

2) La maquette « porte de garage » réelle

La maquette « porte de garage » (Figure 4.35) est un système didactique proposé par la société SCHNEIDER pour l'apprentissage de l'automatisme.

Figure 4.35 : maquette réelle de la « porte de garage »

La porte de garage dispose de trois capteurs (capteur haut de la porte, capteur bas de la porte et présence voiture) et d'un actionneur à commande bistable (ouverture de la porte et fermeture de la porte). Le dialogue Homme-Machine est réalisé au moyen de deux boîtiers, comprenant chacun un voyant et un bouton-poussoir, représentant la demande d'ouverture par le conducteur de part et d'autre de la porte de garage (Figure 4.36).

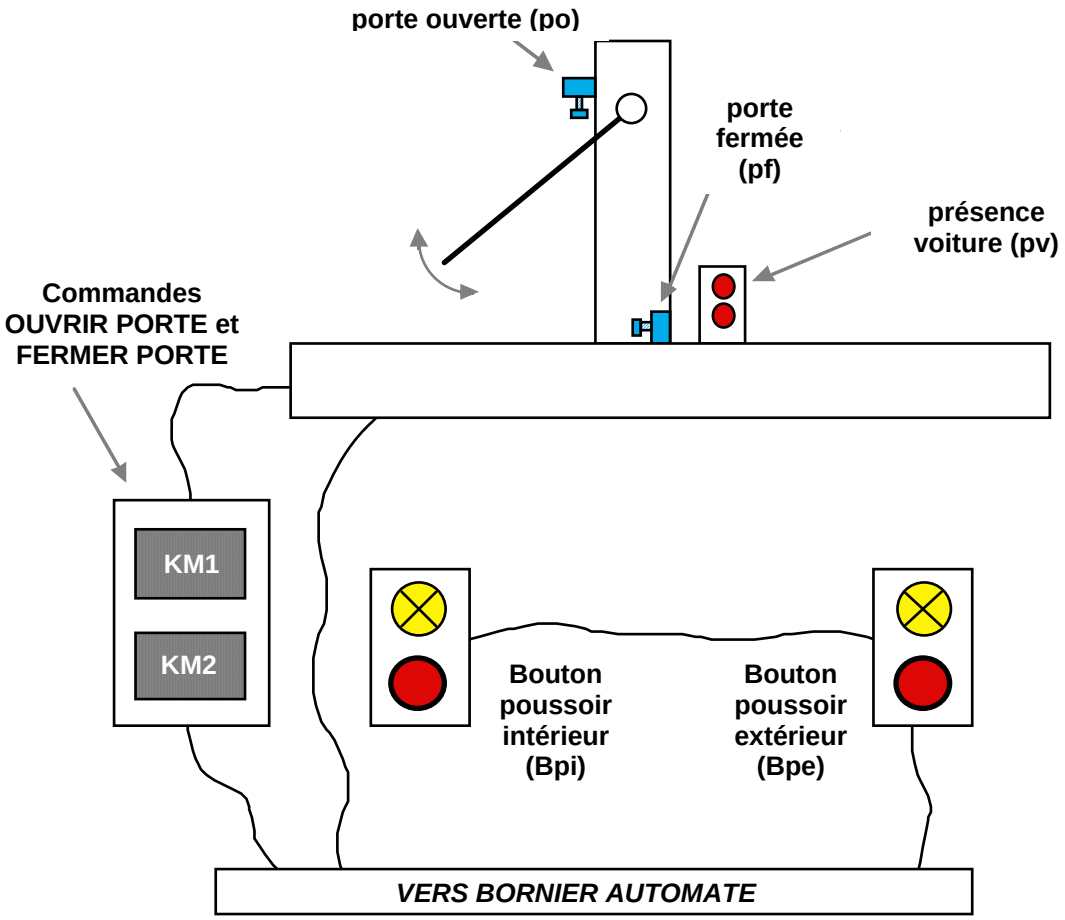


Figure 4.36 : instrumentation de la maquette « porte de garage »

Cette maquette présente une inertie importante qui est accentuée par la position du capteur haut. Cela entraîne des erreurs dans la commande très formatrices pour les étudiants. A titre d'illustration, considérons le cahier des charges suivant :

> Initialement, on considère que la porte est fermée. Aucune action n'est possible tant que le système n'est pas en condition initiale. L'action sur un des deux boutons poussoirs entraîne l'ouverture de la porte et l'allumage des voyants. Lorsque la porte est ouverte, deux cas peuvent se produire :
>
> 1. Au bout de 10 secondes, la voiture n'a pas franchi la porte (pas d'activation de la cellule), la porte se referme automatiquement,
> 2. La cellule est activée avant l'écoulement des 10 secondes, le système attend alors la désactivation de la cellule (fin du passage de la voiture) et commande la fermeture de la porte au bout de 10 secondes à condition qu'au cours de ces 10 secondes aucune demande d'ouverture n'ait eu lieu (intérieur ou extérieur) et qu'aucune nouvelle voiture n'ait re-déclenché la cellule,
>
> Une fois la porte de garage refermée, les voyants s'éteignent.

Parmi les solutions de commande proposées par les élèves, la solution présentée Figure 4.37 illustre le programme de commande en langage SFC implémenté dans l'API.

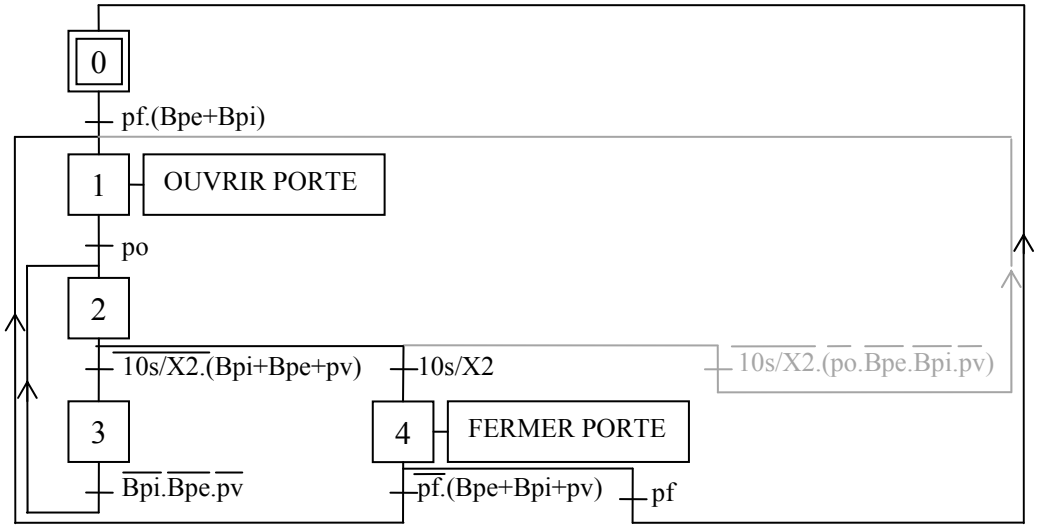


Figure 4.37 : exemple de programme SFC de la porte de garage présentant une instabilité et proposition de correction à apporter

Lors de l'expérimentation de cette solution sur la partie opérative virtuelle, on constate que bien que le cahier des charges proposé soit respecté, un comportement illogique apparaît. L'analyse du comportement du programme montre que l'étape 1 peut-être fugitive. En effet, si lorsque la porte commence à se refermer (étape 4), et que le capteur « *porte ouverte* » est toujours au niveau haut, la porte ne s'ouvrira pas et la porte par l'inertie continuera à se refermer légèrement. Il n'y a plus aucun moyen d'ouvrir la porte totalement sans qu'une fermeture totale n'ait lieu.

Une proposition de solution pour corriger le cahier des charges et résoudre ce problème est d'ajouter une transition en aval de l'étape 2 permettant l'ouverture de la porte si le capteur porte ouverte n'est plus au niveau haut et en indiquant l'exclusivité du franchissement de la transition. Cette solution relance l'ouverture de la porte et la temporisation lorsque le capteur *« porte ouverte »* n'est plus actif.

Le modèle de partie opérative initialement utilisé pour la simulation ne prenait pas en charge le comportement inertiel de la porte de garage. Ce type d'erreur n'étaient alors perçu qu'à l'implémentation du programme de commande sur la partie opérative réelle. L'apport de la modélisation de la partie opérative par les réseaux de Petri hybrides au niveau des maquettes virtuelles est de fournir une démarche méthodologique permettant d'obtenir un modèle de simulation plus proche de la réalité. Il nous semble donc intéressant de modéliser les phénomènes d'inertie en vue de permettre à l'étudiant de les prendre en compte dans la réalisation de la commande lors de l'utilisation de la maquette virtuelle.

3) Modélisation hybride de la « porte de garage » avec prise en compte de l'inertie

La démarche de modélisation permet de représenter le comportement de la partie opérative en tenant compte de l'inertie.

La chaîne d'action est réalisée par deux contacteurs électriques. L'étude du comportement de ces contacteurs révèle que le modèle de fonctionnement est composé de quatre places et des transitions associées (Figure 4.38).

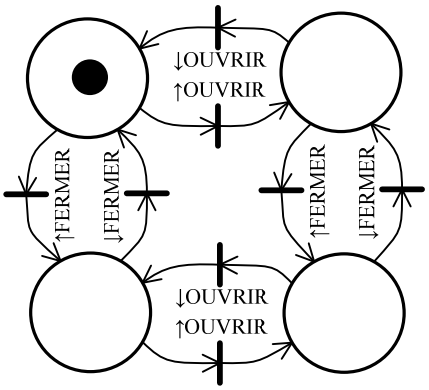


Figure 4.38 : modélisation du préactionneur

Le modèle d'actionneur (Figure 4.39) est composé de deux places continues. Une de ces places est consacrée au degré d'ouverture de la porte. Les trois transitions continues décrivent les évolutions possibles de la porte de garage suivant une équation différentielle du second ordre. Seul le second membre de cette équation diffère d'une transition à l'autre en fonction du mouvement à exécuter. Les paramètres A et B indiquent les constantes à régler dans la modélisation pour s'adapter à la dynamique réelle du système

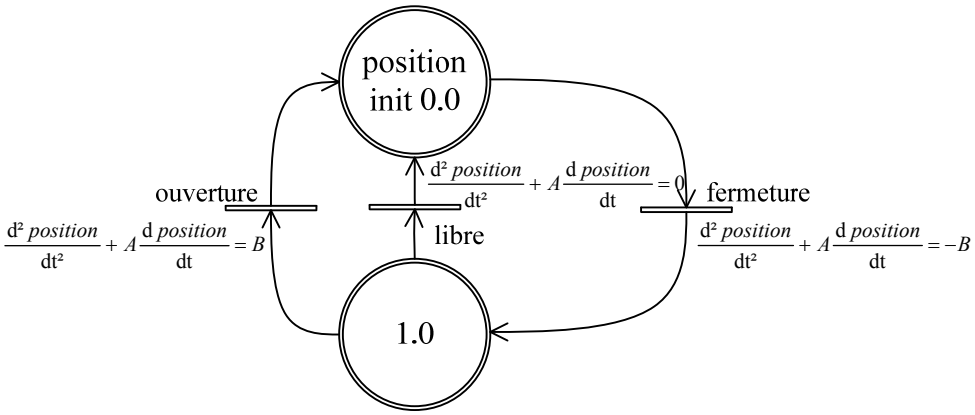


Figure 4.39 : modélisation de l'actionneur

Enfin, le modèle de la chaîne d'acquisition comporte la modélisation des deux capteurs de fin de course de la porte (Figure 4.40). La plage de détection est de 1% de la course totale sans phénomène d'hystérésis.

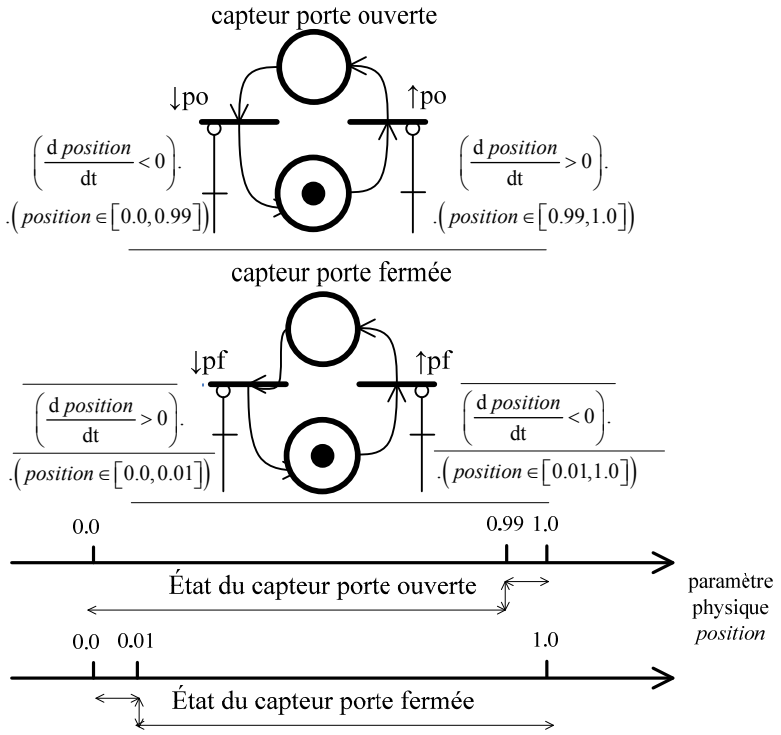


Figure 4.40 : modélisation des capteurs

Les modèles de préactionneur, d'actionneur et des capteurs sont assemblés pour former le modèle de partie opérative de la porte de garage. La représentation de ce modèle est trop complexe pour pouvoir être représenté dans ce mémoire. Il a toutefois été implémenté dans le logiciel de simulation sous la forme de lignes de code de programmation conformément au modèle présenté avec une discrétisation temporelle et physique faible vis-à-vis des grandeurs utilisées dans le modèle. Le choix des paramètres *A* et *B* du modèle d'actionneur sont dépendant des paramètres physiques des constituants de la partie opérative : les masses des pièces mécaniques, la puissance du moteur…

Les essais effectués ont montré que l'inertie peut être prise en compte dans cette modélisation. Le réglage des paramètres a été effectué de façon empirique mais permet désormais de détecter les comportements anormaux générés par les élèves.

5) Conclusion

Les différents points qui ont été abordés au cours de ce chapitre sont liés aux techniques de modélisation de la partie opérative à l'aide d'une démarche basée sur le découpage en chaînes fonctionnelles et à la modélisation comportementale de ces chaînes fonctionnelles. Les approches présentées ont exploité différents outils de modélisation couramment utilisés. Les outils uniquement basés sur une vision discrète du comportement de la partie opérative permettent une vision globale du fonctionnement mais ne permettent pas de tenir compte des comportements physiques continus se déroulant dans le fonctionnement de la partie opérative. La modélisation hybride mêlant les aspects continus des évolutions des grandeurs physiques avec les aspects discrets traités par la partie commande permet une vision plus large et plus réaliste du comportement de la partie opérative. Par contre, ce type de modélisation ne permet pas de décrire à l'avance l'ensemble des évolutions susceptibles de se produire par la suite : le manque de construction d'un modèle global décrivant toutes les évolutions pouvant avoir lieu depuis une situation donnée ne permet pas d'utiliser ces modèles dans toutes les applications liées à l'élaboration de la commande par synthèse par exemple. Néanmoins, dans le cadre de l'utilisation en vue de simuler le comportement de la partie opérative, ce type de modèle devient particulièrement intéressant car il reflète avec un maximum de réalisme de comportement de la partie opérative.

Les différents exemples de chaînes fonctionnelles qui ont été traitées ont illustré les spécificités pouvant être prise en compte dans la modélisation et la proximité avec l'applicatif dans la modélisation de la partie opérative des systèmes automatisés manufacturiers. Les applications de simulation sur les maquettes virtuelles ont montré l'intérêt d'une telle modélisation et l'apport dans la phase de réalisation du modèle.

Conclusion générale et perspectives

Nous avons proposé une démarche de modélisation du comportement de la partie opérative basée sur l'analyse des constituants physiques. Cette approche est dédiée à une utilisation pratique de la modélisation en vue de l'implantation réelle sur des systèmes automatisés. Le modèle de partie opérative peut être exploité dans des applications diverses telles que l'élaboration de la commande, la détection de comportements anormaux de la commande ou de la partie opérative réelle, la reconstruction d'informations utiles pour l'environnement du système automatisé au sein de l'entreprise…

La vision totalement discrète du comportement de la partie opérative permet de créer un modèle exhaustif décrivant l'ensemble des évolutions susceptibles de se produire. Cette vision se révèle limitée lorsqu'il s'agit d'implanter le modèle : l'explosion combinatoire, les difficultés de synchronisation entre le modèle et la réalité, les limites des outils de modélisation utilisés ne favorisent pas l'intégration de ces travaux dans les applications industrielles.

La vision hybride apporte à la perception discrète de la partie opérative au travers la commande, la prise en compte du fonctionnement physique continu mis en œuvre dans le système automatisé. Cependant, le comportement continu n'autorise pas l'énumération de toutes les situations que la partie opérative est susceptible de prendre. En revanche, l'aspect continu des grandeurs physiques et du temps de la modélisation hybride favorise la simulation réaliste des évolutions possibles. Les exemples d'implantation de modèles dans les différentes applications illustrent les atouts et les inconvénients de ces formes de représentation.

Nous proposons trois perspectives au travail de recherche présenté dans ce mémoire. La première concerne l'utilisation des modèles de PO dans le concept du Manufacturing Execution System (MES), c'est-à-dire les couches hautes de la pyramide CIM. La seconde perspective s'intéresse à la prise en compte du produit dans nos modèles de PO. Enfin, nous terminons par la possibilité d'intégrer nos modèles de PO dans le simulateur ITS PLC, pour des applications pédagogiques originales. Ce dernier point fait l'objet d'un partenariat industriel entre le CReSTIC et la société REAL GAMES.

Utilisation de la modélisation de la PO dans le concept MES

Le Manufacturing Executive System (MES) est un concept visant à intégrer le procédé de fabrication dans le système d'information de l'entreprise. L'idée consiste à interfacer le progiciel de gestion de l'entreprise Enterprise Ressource Planning (ERP) avec le process pour améliorer l'efficacité du moyen de production. L'ERP intègre toutes les données issues des

différents services tels que le service client, la logistique, la gestion du personnel, le service financier… L'intégration du service de la production dans le système d'information est limitée : il s'agit principalement pour l'ERP de proposer les ordres de fabrication à produire et d'intégrer les comptes rendus de production. Une telle vision de la production est très restrictive car l'utilisation d'un procédé de fabrication manufacturier automatisé implique nécessairement une interaction avec le système de pilotage. Le MES tend à combler ce manque de liens entre le système d'information global de l'entreprise et le système manufacturier au travers onze fonctionnalités (Figure 1).

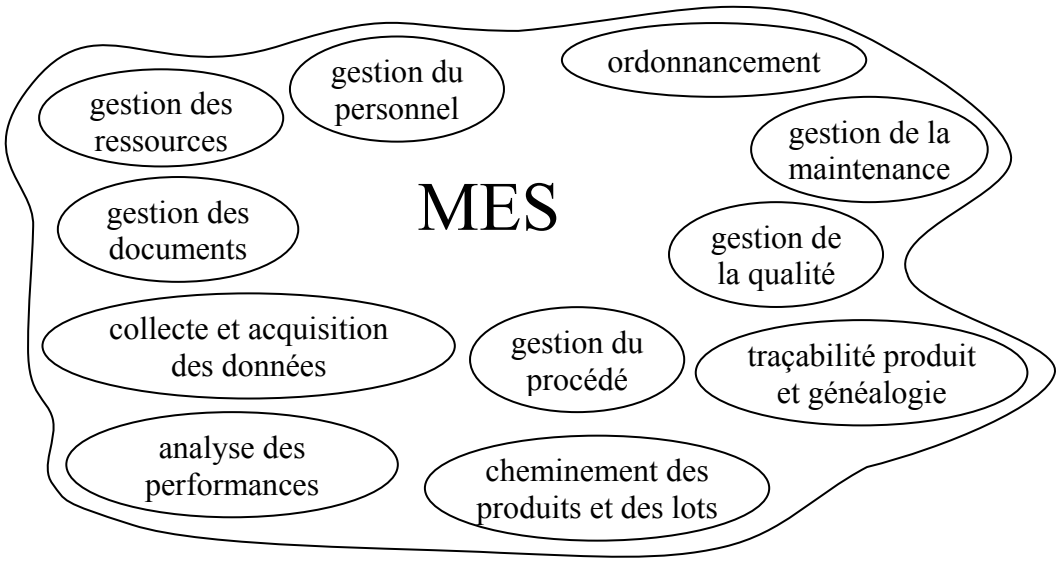


Figure 1 : les onze fonctions du MES

Les onze fonctions du MES permettent de structurer les échanges ayant lieu entre les différents niveaux hiérarchiques d'informations circulant dans l'entreprise. En effet, au niveau le plus bas, le contrôle-commande et la supervision collectent toutes les informations issues du process au travers des capteurs et des commandes bas-niveau. Il s'agit de piloter le système pour qu'il réalise les fonctions de production pour lesquelles il a été conçu. L'ordre de grandeur de l'échelle de temps à laquelle le système réagit se situe vers la seconde car il est nécessaire d'avoir la réactivité due à l'utilisation en temps réel du process physique. Ces process sont classés suivant trois groupes : les process batch mélangeant différents produits pour en former un nouveau, les process continus n'observant pas de cycle de fonctionnement et les process discrets effectuant successivement des opérations sur le produit.

Le MES se situe à un niveau au dessus du contrôle commande et de la supervision. Il utilise ces informations et les agrègent pour permettre un pilotage plus haut niveau de la production. Les optimisations pouvant être réalisées à ce niveau concernent l'ordonnancement des ordres de fabrication, la gestion des recettes, la traçabilité des données de production, la surveillance de la qualité, la mesure de performance du système de production… L'ordre de grandeur de l'échelle de temps utilisées pour le MES est la heure. Les temps de réactivité liés au process

physique ne sont pas présents à ce niveau mais les fonctionnalités du MES sont essentielles pour avoir la réactivité des informations liées à la production.

L'ERP se situe à un niveau bien plus élevé que le MES dans le concept CIM (Figure 2). Il constitue le système d'informations global de l'entreprise et regroupe toutes les données issues des différents services. Le partage et la fusion de ces informations entre tous les interlocuteurs permettent d'améliorer le rendement du secteur tertiaire de l'entreprise. L'optimisation pouvant être effectuée pour la production est de planifier quels sont les produits à réaliser en fonction de critères économiques et financiers. Les temps mis en jeu au niveau ERP sont de l'ordre de la journée. L'information traitée est très abstraite sans lien direct avec le pilotage du process de fabrication.

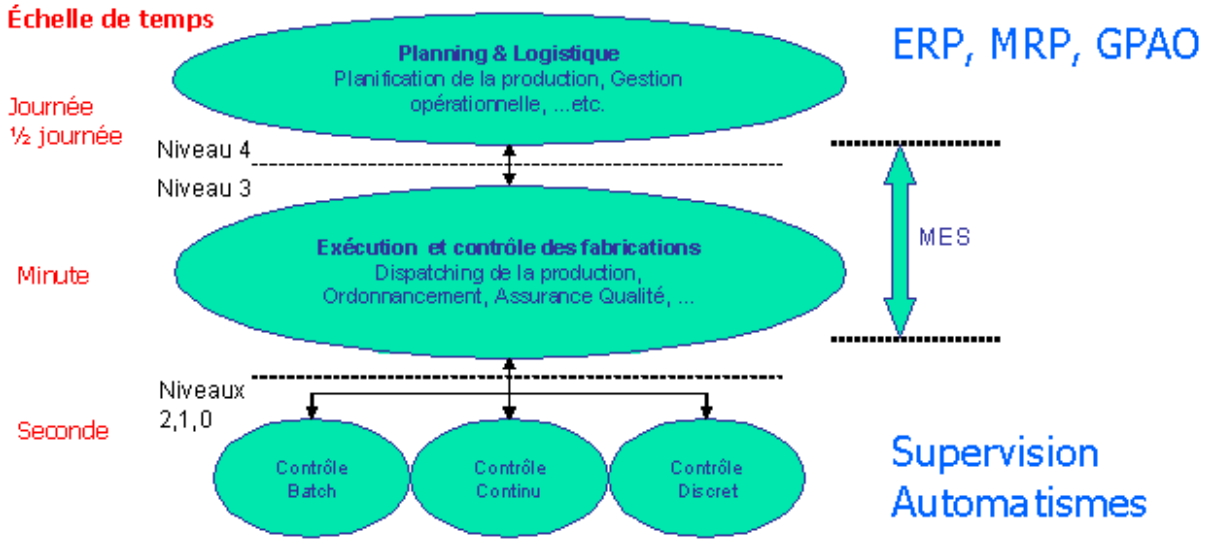


Figure 2 : Utilisation du MES dans le concept CIM

La mise en place du concept MES se heurte à des difficultés importantes liées au traitement de l'information. La structuration de l'application se révèle particulièrement difficile à effectuer car de nombreuses informations sont accessibles au travers de la supervision et de la commande mais cette profusion de données peut se révéler néfaste lorsqu'il s'agit d'extraire des informations exploitables pour les niveaux supérieurs du concept CIM. Certaines fonctions du MES doivent pour être performantes, selon notre point de vue, reposer sur un modèle dynamique de la PO. Il s'agit des 6 fonctions suivantes : Cheminement des produits et des lots, Gestion de la Qualité, Gestion du procédé, Gestion de la maintenance, Traçabilité produit et généalogie et Analyse des performances. Les normes ISA88 et ISA 95 apportent une structuration plus forte du domaine du MES et aident à lever certaines confusions en apportant des modèles de structure des données de bas et haut niveaux permettant la standardisation des échanges de données entre différents systèmes d'information. Ces structures de données reposent toutefois sur une vision purement statique du système de production. Un axe de recherche très prometteur est de parvenir à lier les modèles dynamiques de PO aux différents modèles de données proposées dans les normes ISA. Il deviendrait ainsi

beaucoup plus aisé d'exploiter les données pour obtenir de l'information pertinente pour améliorer les performances (au sens large) du système de production.

Toutefois ce travail passe nécessairement par une meilleure prise en compte du produit dans nos modèles de PO. Ce point fait l'objet d'une perspective importante qui est développée dans le paragraphe suivant.

Vers l'assemblage des chaînes fonctionnelles et la prise en compte du produit

La démarche de modélisation proposée se limite pour l'instant à la modélisation des chaînes fonctionnelles. Chaque actionneur et les équipements associés formant une chaîne fonctionnelle, la modélisation de ces chaînes fonctionnelles est réalisée de manière indépendante. Bien que les constituants de la partie opérative présente une modélisation au plus proche de la réalité, l'assemblage des chaînes fonctionnelles et des flux de production ne sont pas considérés dans ce mémoire. Pour réaliser cet assemblage, il faut prendre en compte les interactions entre les chaînes fonctionnelles. Dans un premier temps, toutes les grandeurs physiques introduites dans le modèle de simulation doivent être exploitées pour maitriser les interactions ayant lieu entre les chaînes fonctionnelles. Les collisions entre actionneurs peuvent être également définies en contraignant les évolutions des grandeurs physiques dans certaines situations. Cependant, certaines interactions sont beaucoup plus complexes à considérer lorsque le produit intervient. En effet, les produits présents dans les systèmes manufacturiers génèrent une grande diversité de collisions avec les actionneurs. Prendre en compte ces interactions se révèle particulièrement difficile car les informations sur la présence, l'état, la configuration… du produit ne sont pas toujours simples à déterminer surtout avec l'utilisation restreinte de capteurs. Par conséquent, les évolutions des produits nécessitent également d'être prises en compte dans la modélisation.

Cependant, la démarche proposée actuellement ne peut pas facilement être étendue pour prendre en compte le produit. Les caractéristiques du produit peuvent être assimilées à des grandeurs physiques mais les évolutions de ces grandeurs physiques ne sont pas directement dépendantes des commandes exécutées comme peuvent l'être les grandeurs physiques associées aux chaînes fonctionnelles. Les grandeurs physiques associées aux produits peuvent être synchronisées avec les évolutions des grandeurs physiques des actionneurs mais elles dépendent des grandeurs physiques des autres produits en cours de traitement. Les grandeurs physiques associées aux produits ont également des incidences sur les grandeurs physiques des actionneurs. Pour modéliser le comportement de la partie opérative en tenant compte des produits, il est indispensable de définir toutes les interactions pouvant avoir lieu entre les différentes grandeurs physiques associées à chaque produit et les grandeurs physiques dépendant des actionneurs. La Figure 3 illustre les liens devant être modélisés lorsque les produits sont pris en compte dans le modèle.

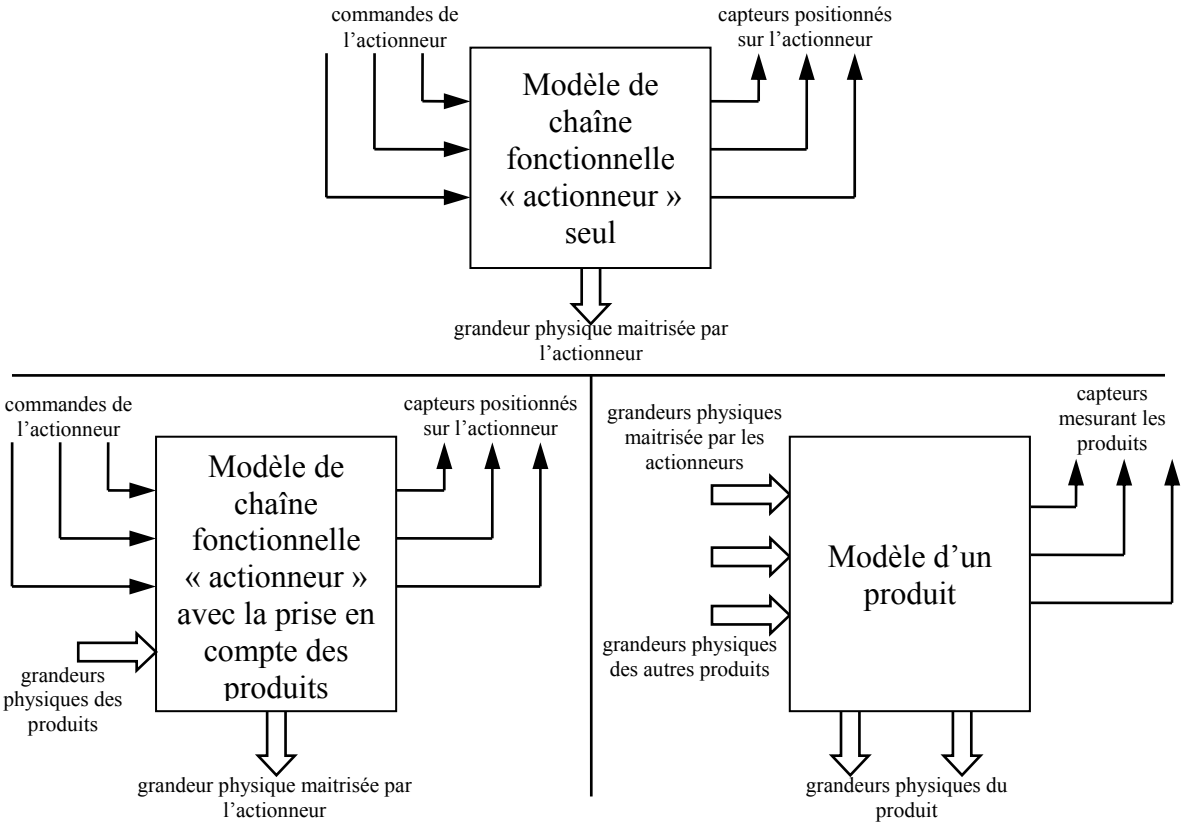


Figure 3 : conséquence de la prise en compte du produit dans la modélisation du comportement de la partie opérative

Dans le cas où les actionneurs sont modélisés indépendamment, la grandeur physique maîtrisée par la chaîne fonctionnelle n'évolue qu'à partir des commandes et modifie l'état des capteurs. Dans le cas où le produit doit être pris en compte, il faut prendre en considération les interactions ayant lieu entre les chaînes fonctionnelles et le produit et les interactions ayant lieu entre les produits eux-mêmes. Pour chaque produit, il est nécessaire de définir un modèle décrivant ses caractéristiques physiques et les évolutions ayant lieu lors du fonctionnement. Les évolutions de grandeurs physiques à prendre en compte sont les grandeurs physiques pilotées par les chaînes fonctionnelles et les grandeurs physiques qui ont une incidence sur le produit. Le modèle de comportement des chaînes fonctionnelles est dépendant des commandes mais également des grandeurs physiques des produits. En effet, la présence ou non d'un produit peut avoir des incidences sur le comportement de la chaîne fonctionnelle (modification des inerties lorsqu'un robot tient une pièce). Les évolutions des grandeurs physiques de chaque produit peuvent être dues aux évolutions des chaînes fonctionnelles (déplacements d'une pièce sur un tapis) mais aussi des évolutions des autres produits (contacts entre pièces qui se poussent). Des capteurs peuvent être positionnés sur le système automatisé pour détecter la présence et les caractéristiques des produits au cours de la fabrication. Ce concept de modélisation de la partie opérative en tenant compte des produits devient alors extrêmement complexe car les interactions se produisant entre toutes les grandeurs physiques sont nombreuses et difficiles à décrire. De plus, il peut apparaître des

configurations dans laquelle il y a des contradictions dans la modélisation : des boucles peuvent se former dans la modélisation de la partie opérative générant des conflits dans les évolutions des grandeurs physiques. Ces conflits nécessitent d'être arbitrés suivant des règles à définir. Il s'agit de résoudre un système d'équations hyper-contraint. Si on prend le cas des contacts mécaniques, il est possible de se retrouver dans une situation pour laquelle les contacts sur le produit génèrent des mouvements et/ou des efforts contradictoires. Ce type de problème hyperstatique ne peut se résoudre qu'en effectuant des hypothèses pour restreindre le nombre d'inconnues ou en prenant en compte la déformation des pièces ce qui a pour effet d'augmenter le nombre d'inconnues dans le système d'équations.

Les réflexions sur la place du produit dans la modélisation sont un point particulièrement important à considérer pour obtenir un modèle de partie opérative utilisable dans les applications réelles. Plutôt que de tout développer, une autre solution consisterait à exploiter et à réutiliser des moteurs physiques existants. Des perspectives détaillées dans le paragraphe suivant peuvent être mises en avant dans ce cadre pour l'intégration de la démarche de modélisation dans les outils de simulation existant.

Vers une intégration des modèles proposés dans le logiciel ITS PLC

Dans le chapitre 4, nous avons présenté très brièvement le logiciel de simulation de parties opératives ITS PLC. Ce logiciel pédagogique a pour vocation l'enseignement de la commande et la supervision des systèmes automatisés. Un partenariat scientifique et technique a été conclu entre le CReSTIC et la « jeune pousse » portugaise à l'origine du projet : « REALGAMES ». Des chercheurs du CReSTIC ont participé au développement de la version française du logiciel et à l'amélioration des parties opératives simulées actuellement disponibles [Rohee, 08], [Riera, 08]. Le partenariat entre le CReSTIC et la société REALGAMES devrait être accentué dans les années à venir et il est envisagé d'intégrer les modèles des chaînes fonctionnelles proposés dans cette thèse dans les futurs développements du logiciel. Les paragraphes suivants illustrent le principe de ce logiciel de simulation et comment exploiter la modélisation hybride présentée dans ces travaux.

ITS PLC Professional Edition est un logiciel éducatif adapté initialement à l'apprentissage de la programmation des Automates Programmables Industriels (API). Basé sur les dernières technologies informatiques, des environnements virtuels de machines automatisés sont proposés à partir de la simulation de parties opératives. Même si les concepts sont identiques, la présentation graphique en 3D, le rendu sonore et l'interactivité sont très supérieurs aux maquettes virtuelles utilisant le simulateur WINSIM et un SCADA présentées dans le chapitre 4. Cinq systèmes ont ainsi été intégrés empiriquement dans le logiciel (tri de caisses, mélangeur, palettiseur, robot « pick and place » et magasin automatisé). La Figure 4 montre l'affichage graphique rendu par le logiciel.

Figure 4 : représentation graphique du palettiseur dans ITS PLC

Ces parties opératives virtuelles sont pilotées directement par un API grâce à une interface physique liant les commandes et les capteurs aux entrées/sorties physiques de l'API (Figure 5). Les contraintes de synchronisation entre le modèle de simulation du logiciel et les informations contenues dans l'API limitent les phénomènes de latence les temps de réponse de la partie opérative virtuelle aux sollicitations de l'API sont moindre en comparaison avec la structure mise en place dans les exemples du chapitre 4 basés sur WINSIM. De plus, la partie opérative est liée physiquement à l'API au travers des entrées/sorties physiques utilisées avec les parties opératives réelles.

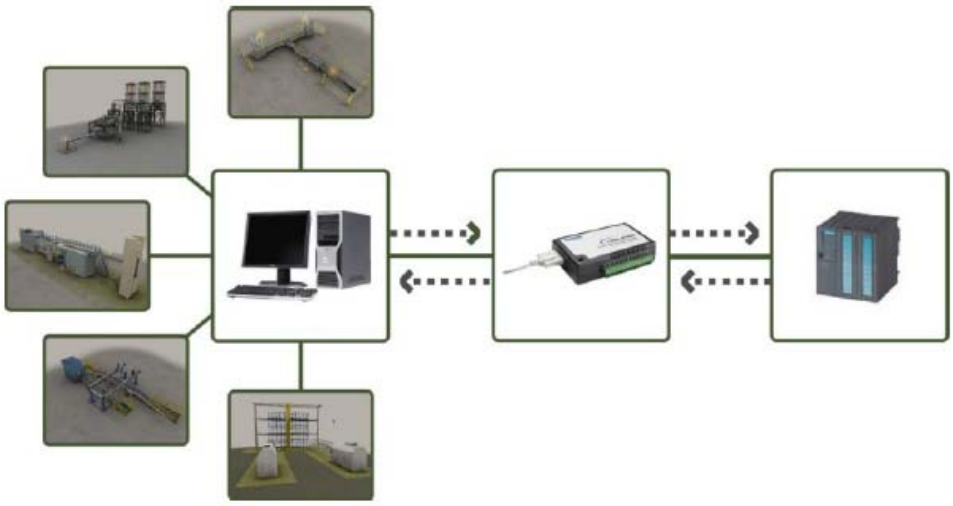


Figure 5 : utilisation du logiciel au travers une interface physique entre le modèle simulé et un API

ITS PLC repose sur l'utilisation de trois moteurs informatiques sous la forme de bibliothèques logicielles : graphique, physique et instrumentation (Figure 6). Le moteur graphique DirectX est consacré à l'affichage des textures et au rendu sonore. Le moteur physique Newton calcule les paramètres physiques consécutifs à l'assemblage des chaînes fonctionnelles et à la présence des produits par la simulation des lois de la physique (collisions, gravité, accélérations, efforts…) présents dans les jeux vidéos. Ce moteur permet, en autre, une simulation de l'assemblage des chaînes fonctionnelles évoquée dans les perspectives de ce mémoire.

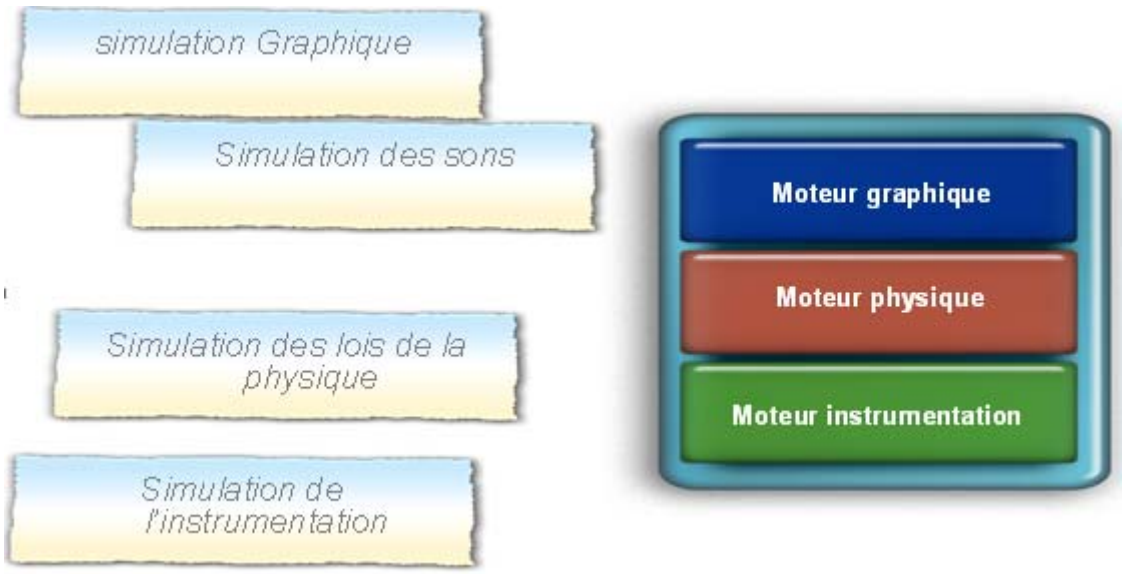


Figure 6 : moteurs informatiques de ITS PLC

Le moteur d'instrumentation représente le comportement des capteurs, des actionneurs et des préactionneurs en lien avec le moteur physique. Le moteur présent dans ITS PLC est peu développé voire simpliste : les modèles utilisés sont peu réalistes vis-à-vis des constituants classiquement présents sur les partie opératives réelles. Dans le cadre de la collaboration entre le CReSTIC et la société REALGAMES, il est envisagé d'utiliser les modèles hybrides proposés dans cette thèse pour améliorer le moteur d'instrumentation. Cela permettra entre autres d'introduire de l'inertie dans les actionneurs et un plus grand réalisme dans la représentation des actionneurs.

Les perspectives de développement autour des concepts de prise en compte des produits tels qu'ils ont été présentés ici sont nombreuses et nécessaires pour améliorer l'applicabilité des résultats de recherche vers des applications industrielles.

Bibliographie

[Achour, 2004] Achour Z., Rezg N., Commande par supervision des systèmes a événements discrets basée sur l\'utilisation du GRAFCET et la théorie des régions, Journal européen des systèmes automatisé, Vol. 38, N° 1-2, 37-58, 2004

[Akesson, 2003] Akesson K, Fabian M., Flordal H., Vahidi A., Supremica—a tool for verification and synthesis of discrete event supervisors, Proc. of the 11th Mediterranean Conference on Control and Automation, Rhodos, Greece, 2003.

[Akesson, 2007] Akesson K., Fabian M. Flordal H., Supremica in a Nutshell – Draft", 10/2007

[Balemi, 1992] Balemi, S., Control of discrete event systems: theory and application, PhD Thesis, Swiss Federal Institute of Technology, Zürich, Switzerland, 1992

[Balemi, 1993] Balemi S., Kozák P., Smedinga R., Discrete Event Systems: Modeling and Control, Progress in Systems and Control Theory, Birkháuser Verlag, Basel, Switzerland, 1993.

[Bekrar, 2006] Bekrar R., Messai N., Essounbouli N., Hamzaoui A., Riera B., Identification of Discrete Event Systems using ordinary Petri Nets, IAR-ACD- Workshop on Advanced on Control and Diagnosis, Nancy, France, 11/2006.

[Boussoffara, 1998] Boussoffara B., Elzer P., About human pattern matching and time information in S&C of large technical systems, Proceedings of the 17th European Annual Conference on Human Decision Making and Manual Control, Valenciennes, France, pp.161–166, 1998

[Carré-Ménétrier, 2002] Carré-Ménétrier V., Zaytoon J., Grafcet: behavioural issues and control synthesis, European Journal of Control (EJC), Vol. 8, n°4, pp. 375-401, 2002.

[Chandra, 2001a] Chandra V., Kumar R., A new modeling formalism and automata model generator for a class of discrete event systems, In Proc American Control Conference, 2001

[Chandra, 2001b] Chandra V., Kumar R., A discrete event system modeling formalism based on event occurrence rules and precedences, IEEE Transactions on Robotics and Automation, pp. 785-794, Vol. 16, number 6, 12/2001

[Chandra, 2002] Chandra V., Kumar R., A event occurrence rules based compact modeling formalism for a class of discrete event systems. Mathematical and computer modeling of dynamical systems, 2002

[Combacau, 2000a] Combacau M., Berruet P., Charbonnaud F., Khatab A., Zamai E., supervision and monitoring of production systems, proc. of MCPL'2000, Grenoble, 07/2000

[Combacau, 2000b] Combacau M., Berruet P., Charbonnaud F., Khatab A., Réflexions sur la terminologie : Surveillance – Supervision, Groupement pour la Recherche en Productique, Systèmes de Production Sûrs de Fonctionnement, 03/2000.

[Combacau, 2001] Combacau M., Approche discrète de la surveilllance des systèmes de production, traité IC2 Hermès, Maiîtrise des risque de sûreté de fonctionnement des systèmes de production, Chap. 11, 09/2001

[Darkhovski, 2003] Darkhovski B., Staroswiecki M., A game-Theoretic Approach to Decision in FDI, IEEE Transactions On Automatic Control, Vol. 48, N°5, 2003.

[Dimitrova, 2005] Dimitrova D., Frey G., Batchkova I., Formal Approach for Modeling and Verification of IEC 61499 Function Blocks. Proceedings of the International Conference AMTECH 2005 "Advanced Manufacturing Technologies", University of Russe, Bulgaria, Volume 44, Book 2, pp. 731-736, 09/ 2005.

[Endsley, 1998] Endsley M.R., Situation awareness global assessment technique (SAGAT), Aerospace and Electronics Conference, Vol. 3, pp. 789-795 Dayton, OH, USA, 1988

[Fabian, 1998] Fabian M., Hellgren A., PLC-based Implementation of Supervisory Control for Discrete Event Systems, Proceedings of the 37th IEEE Conference on Decision and Control, Tampa, Florida, USA, 1998

[Flaus, 1997] Flaus J-M., Les systèmes dynamiques hybrides : approche de base et potentiel pour la commande de procédés, 6ème Congrès Français de Génie des Procédés, Récents progrès en génie des procédés, Simulation des procédés et automatique, N°56, vol. 11, pp.91-96, Paris, France, 1997

[Frey, 2000] Frey G., Litz, L., IEEE conference on systems man and cybernetics SMC 2000, Nashville, 10/2000

[Frey, 2002] Frey G., Formal methods in PLC control demonstrated at a flexible manufacturing line. Proceedings of the 5th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS2002), Cancun, Mexico, pp. 501-508, 09/2002.

[Holloway, 1994] Holloway L-E., Krogh B-H., Controlled Petri nets: A tutorial survey, 11th Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems, volume 199 of LNCS, 1994

[Hopcrott, 1979] Hopcroft J. E. Ullman J. D., Introduction to Automata Theory, Languages and Computations, Addison-Wesley, 1979

[Huang, 2002] Huang Z., Chandra V., Jiang S., Kumar R., Modeling Discrete Event Systems with Faults using a Rules Based Modeling Formalism, Mathematical and Computer Modeling of Dynamical Systems, 2002

[Huang, 2003] Huang Z., Chandra V., Jiang S., Kumar R., Modeling Discrete Event Systems with Faults using a Rules Based Modeling Formalism,. Mathematical and Computer Modeling of Dynamical Systems , pages 233-254, volume 9, number 3, 2003.

[Huang, 2005] Huang J., Kumar R.. Nonblocking Directed Control of Discrete Event Systems, IEEE Transactions on Automation Science and Engineering, 08/2005

[Huang, 2006] Huang J., Kumar R.. Directed Control of Discrete Event Systems: Optimization based Approach, IEEE Transactions on Systems, Man, and Cybernatics Part A, 01/2006

[IEC-61508] International Electrotechnical Commission 61508 : Funtional safety of electrical/electronic/ programmable electronic safety-related systems, 02/2003

[Inan, 1989] Inan K. M., Varaiya, algebras of discrete event models, Proc. of IEEE, Vol.77, n°1, 01/1989

[ITS_PLC] ITS PLC software, REALGAMES (http://www.realgames.pt)

[Klein, 2003] Klein S., Frey G., Minas M., PLC Programming with Signal Interpreted Petri Nets. Proceedings of the ICATPN 2003, Eindhoven (The Netherlands), LNCS 2679, pp. 440-449, Springer Verlag, 06/2003.

[Klein, 2005] Klein S., Identification of Discrete Event Systems for Fault Detection Purposes, Doctorat de l'Ecole Normale Supérieure de Cachan, Spécialité : Electronique Electrotechnique et Automatique 09/2005

[Lambert, 1998] Lambert M., Riera B., Martel G., Analysis, design and evaluation of a new supervision system for the control of a nuclear fuel reprocessing

system, Proceedings of the 7th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man–Machine Systems, Kyoto, Japan, pp.455–460, 1998

[Machado, 2006] Machado J., Influence de la prise en compte d'un modèle de processus en verification formelle des systems à évènements discrets, these de doctorat de l'Ecole Normale Supérieur de Cachan et de l'Université du Minho (Portugal), juin 2006

[Makungu, 1999] Makungu M., Barbeau M., Saint-Denis R., Synthesis of controllers of processes modeled as colored Petri Nets Discrete Event Dynamic Systems : theory and applications, vol. 9, pp. 147-169, Kluwer academic Publishers, 1999

[Meda-Campaña 1998] Meda-Campaña M.E., Identification using interpreted Petri nets, International Symposium of Robotics and Automation, pp. 353-357, Saltillo Coahuila, México, 12/1998.

[Meda-Campaña 2000] Meda-Campaña M.E., López-Mellado E., Asymptotic identification of discrete event systems, 39th IEEE Conference on Decision and Control, pp. 2266-2271, Sydney, Australia, 12/2000.

[Meda-Campaña 2003] Meda-Campaña M.E., López-Mellado E., Required transition sequences for identification of DES, 42st IEEE Conference on Decision and Control. Hawaii USA, 12/2003.

[Millot, 1998] Millot, P., Lemoine, M.P., An attempt for generic concepts toward human–machine cooperation, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, USA, 1998

[Music, 2002] Music G., Zupancic B., Matko D., Model based programmable control logic design, in proc. of the 15th Triennial IFAC World Congress, Barcelona, Spain, 2002

[Perrin, 2001] Perrin J., Concevoir l'innovation industrielle ; méthodologie de conception de l'innovation, CNRS Editions, 02/2001

[Philippot, 2006] Philippot A., Contribution au diagnostic décentralisé des SED : application aux systèmes manufacturiers, Thèse de doctorat de l'Université de Reims Champagne-Ardenne, 07/006

[Philippot, 2004] Philippot A. Tajer A, Gellot F., Carré-Ménétrier V., Synthèse de la commande spécifiée en Grafcet : application à un préhenseur pneumatique, colloque francophone sur la modélisation des systèmes réactifs, MSR'03, Metz, pp. 61-75, 10/2003

[Qiu, 2000] Qiu, B. and Gooi, H.B. Web-based SCADA Display Systems (WSDS) for access via internet, IEEE Transaction on power systems, Vol. 15, pp.681–686, 2000

[Ramadge, 1989] P. J. Ramadge, W. M. Wonham. The control of discrete event systems. Proc. IEEE, Special Issue on Discrete Event Dynamic Systems, 77 (1):81-98, 01/1989

[Raush, 1998] Raush M. Krogh BH., Formal verification of PLC programs, Proceedings of AAC'98 Philadelphia PA USA, 06/1998

[Riera, 1999a] Riera, B., Martel, G., Anguè, J.C, La simulation : outil pédagogique pour l'enseignement de l'automatique dans les formations professionnalisées, Sciences et techniques éducatives, vol. 6, n°1, pp 37-60, Hermes Sciences, 12/1999

[Riera, 1999b] Riera, B., Martel, G., Lambert, M. Des exemples d'initiation attractive à l'EEA, Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 99), CEPADUES EDITIONS, ISBN 2.85428.515.8. (pp.343-347). Montpellier, France, 11/1999

[Riera, 2001] Riera, B., Conreur, G., Chemla, J-P Une salle de TP d'API adaptée aux NTE, Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 01), Campus des Cézeaux, Université de Clermont-Ferrand, France, 10/2001

[Riera, 2003a] Riera B., Debernard S., Basic Cognitive Principles Applied to the Design of Advanced Supervisory Systems for process control, Handbook of Cognitive Task Design. Erick Holnagel (Ed.) Lawrance Erlbaum Associates, Publishers, pp 255-281. 2003

[Riera, 2003b] Riera B., Gellot F., Actes de la journée d'étude sur l'innovation technologique dans la supervision industrielle : Des outils aux applications, Editeurs, Reims, France, 10/2003

[Riera, 2005] Riera B., Gellot F., Marangé P., Chemla J-P., Sayed Mouchaweh M., Un projet original en commande et supervision des systèmes automatisés : des enfants de 5 ans aux secours des animaux malades Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 05). Nancy, France, 10/2005

[Riera, 2008] Riera B., Vigario B., Chemla J-P., Correia L., Gellot F., 10 ans de maquettes virtuelles pour l'enseignement des automatismes de WINSIM en 1998 à ITS PLC Professional Edition en 2008, CETSIS, Bruxelles, Belgique, 10/2008

[**Rohee, 2005**] Rohee B., Répartition dynamique d'activité sur un automate programmable industriel à moniteur non préhemptif, Mémoire de DEA de Production Automatisé, LURPA, encadré par De Smet O., 06/2005

[**Rohee, 2006**] Rohee B., Riera B., Carré-Ménétrier V., Roussel J-M., A methodology to design and check a plant model, 3rd IFAC Workshop on Discrete Event System Design, DESDes06, Rydzyna, Poland, 09/2006

[**Rohee, 2007a**] Rohee B., Riera B., Carré-Ménétrier V., Manufacturing human machine design using plant models, IFAC/IFIP/IFORS/IEA Symposium-Analysis design and evaluation of human machine systems, Seoul, Korea, 09/2007

[**Rohee, 2007b**] Rohee B., Riera B., Carré-Ménétrier V., Roussel J-M., Outil d'aide à l'élaboration de modèles hybrides de simulation pour les systèmes manufacturiers, JN-JD MACS 2007, Reims, France, 07/2007

[**Rohee, 2007c**] Rohee B., Riera B., Du capteur à l'ERP : principes et mise en application des concepts MES sur une maquette pédagogique, Colloque sur l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS 2007), 10/2007

[**Rohee, 2008a**] Rohee B., Riera B. Des parties opératives pour enseigner l'automatisme, Revue d'enseignement Technologies et formations, Réseau SCEREN, à paraître

[**Rohee, 2008b**] Rohee B., Carré-Ménétrier V., Riera B., Proposition of completeness property to perform the plant modelling for manufacturing applications, 17th World Congress International Federation of Automatic Control (IFAC), Seoul, Korea, 07/2008

[**Rohee, 2009**] Rohee B., Riera B., Advanced supervisory control for manufacturing systems: from concepts to a separated monitoring system, International Journal of Intelligent Systems Technologies and Applications (IJISTA) special issue on intelligent system design and development, accepted, to be published in 2009

[Rossi, 2003] Validation formelle de programmes Ladder Diagram pour Automates Programmables Industriels, Thèse de Doctorat en EEA, Ecole Normale Supérieure de Cachan, 208p., 06/2003

[Roussel, 1994] Roussel J-M, Analyse De Grafcets Par Generation Logique De l'Automate Equivalent, thèse de doctorat, Ecole Normale Supérieure de Cachan, 12/994

[Roussel, 1996] Roussel J-M.., Lesage J-J., Validation And Verification Of Grafcet Using State Machine, Proceedings of IMACS-IEEE CESA'96 pp. 758-764, Lille, 07/1996

[Roussel, 2005] Roussel J-M., Giua, A., Designing dependable logic controllers using the supervisory control theory, 16th IFAC World Congress, CDROM paper n°04427, 6 pages, Prague (République Tchèque), 07/2005

[Roussel, 2006] Roussel J-M., Faure J-M., Designing dependable controllers using algebraic specifications, Control Engineering Practice, Volume 14, Issue 10, pp. 1143-1155, 10/2006

[Sheridan, 1988] Sheridan, T.B. Tasks allocation and supervisory control, Handbook of Human Computer Interaction, The Netherlands: Elsevier Science Publisher B.V., 1988

[Supremica] Supremica software (http://www.supremica.org)

[Tajer, 2005] Tajer A, Contribution aux approches formelles de synthèse de commande spécifiée par Grafcet, Thèse de doctorat de l'Université de Reims Champagne-Ardenne, 11/2005

[Toguyeni, 2007] Toguyeni A., Dangoumau N, Lee E-J., , Synthesis Approach for Reconfigurable Manufacturing Systems Design Based on Petri Nets, Studies in Informatics and Control (SIC), volume 16 (1), 2007

[Vicente, 1995] Vicente K.J., Christoffersen K., Pereklita, A. Supporting operator problem solving through ecological interface design, IEEE Transactions on Systems, Man and Cybernetics, Vol. 25, pp.529–545, 1995

[Wonham, 1987] W. M. Wonham, P. J. Ramadge, On the supremal controllable sublanguage of a given language, SIAM J. Control Optimization, 25, pp. 637-659, 1987

[Wonham, 2002] W. M. Wonham – Notes on the control of discrete event systems, Systems Control Group, Dept. of electrical and computer engineering, University of Toronto, 07/2002

[Wonham, 2004] Wonham W.M., Supervisory control of discrete-event systems, ECE 1636F/1637S, 2004

[Zaytoon, 1997] Zaytoon J., Lesage J-J., Marce L., Faure J.-M., Lhoste P., Vérification et validation du Grafcet, JESA, 31(4), pp. 713-740, 1997

[Zaytoon, 1999] Zaytoon J., Ndjab Hagbebell C., Carré-Ménétrier V., Grafcet et graphes d'états : synthèse hors-ligne de la commande, APII JESA, Vol. 33, n°7, pp 783-814, 1999

[Zaytoon, 2001] Zaytoon, J. and V. Carre-Menetrier Synthesis of control implementation for discrete manufacturing systems, International Journal of Production Research 39(2), 329–345, 2001

[Zecevic 1998] Zecevic, G., Web based interface to SCADA system, Proceedings of the Power System Technology POWERCON'98, Vol. 2, Beijing, China, pp.1218–1221, 1998

RESUME

Mots-clés : Systèmes à Evènements Discrets, Modélisation, Système manufacturier, Système Homme-Machine, Partie opérative

Quelque soient les applications développées dans le cadre des systèmes à évènements discrets, l'analyse et la représentation du comportement de la partie opérative par un modèle formel est nécessaire. L'analyse bibliographique montre que, suivant les applications et les auteurs, la modélisation de la partie opérative prend des formes très différentes suivant les objectifs à atteindre. Le modèle utilisé est généralement proposé empiriquement et il est supposé faire partie du travail préliminaire aux travaux présentés. Bien que située à la base de tous les travaux, la modélisation en tant que telle, ne fait pas beaucoup l'objet de travaux de recherche.

Nous nous sommes intéressés dans ce mémoire à l'obtention d'un modèle de partie opérative selon deux approches : la première consiste à mettre à l'épreuve formellement un modèle de partie opérative de bas niveau pour tester si tous les changements de commandes sont considérés. La confiance dans le modèle de partie opérative empirique est augmentée en vérifiant partiellement certaines propriétés sur celui-ci. La seconde approche s'intéresse à la construction du modèle de partie opérative par analyse de la composition en constituants physiques de la partie opérative. Chaque composant est modélisé indépendamment suivant un niveau de précision plus ou moins important avant que le modèle de partie opérative ne soit assemblé pour être utilisé dans des applications.

D'abord basés sur les outils de représentation classique de la théorie des langages, les résultats de la modélisation de la partie opérative présentés dans ce mémoire sont ensuite appliqués à la supervision industrielle d'un atelier flexible d'emballage et de conditionnement. Cette modélisation montre ses limites dans la prise en compte du temps et l'aspect continu du comportement de la partie opérative. La démarche de modélisation est ensuite étendue aux Réseaux de Petri hybrides pour pouvoir représenter le comportement discret et continu de la partie opérative et leurs interactions. Cette approche hybride permet enfin d'apporter un cadre formel pour la simulation de parties opératives réalistes intégrée dans le logiciel ITS_PLC Professional Edition.

ABSTRACT

Keywords: Discrete Event System, modeling, manufacturing system, Human Machine Interface, plant part

Whatever applications in the field of Discrete Event System, analysis and plant part behavior modeling represented by a formal model is required. Our bibliographical analysis shows that, according to authors and applications, operative part modeling can take very different forms depending on goals. Used models are generally empirical and are supposed to be defined before the contribution. Although modeling is needed in all the research contribution, modeling does not provide a lot of contribution dedicated to it.

In this thesis, we are interested in how to obtain the plant model with two approaches: the first one consists in putting to the test formally an empirical low-level plant model to test if all the control changes are considered. Confidence in the empirical plant model is improved by verifying partially on this one. The second approach consists in proposing to build the plant model by analyzing the components that compose the plant. Each material component is modeled independently with a precision level more or less important before that the plant model is assembled to be used in applications.

First, the representing tool that is used is based on the language theory. The obtain results on the modeling of the plant parts are applied to the supervisory control of a packaging flexible cell. The modeling tool shows its limits in the taking into account of the time and the continuous aspects of the plant behavior. Then, our approach is extended to hybrid Petri nets to take into account the discrete and continuous behavior of the plant and their interaction. This hybrid approach enables to supply formal context to simulate realist plant models integrated in the software ITS_PLC Professionnal Edition.