Synthèse et filtrage robuste de la commande pour des système manufacturiers sûrs de fonctionnement

Pascale Marangé

UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Thèse
Présentée pour obtenir le grade de

Docteur de l'Université de Reims Champagne-Ardenne

Spécialité
GÉNIE INFORMATIQUE, AUTOMATIQUE ET TRAITEMENT DU SIGNAL

Par Pascale MARANGÉ

Synthèse et filtrage robuste de la commande pour des systèmes manufacturiers sûrs de fonctionnement

---

Soutenue publiquement le 05 novembre 2008, devant le jury composé de

Rapporteurs :	
M. Jean Louis FERRIER	Professeur, LISA, Université d'Angers
M. Jean François PÉTIN	Maitre de conférences HDR, CRAN, Université de Nancy
Examinateurs :	
Mme Véronique CARRÉ MÉNÉTRIER	Professeur, CReSTIC, Université de Reims
M. Yvon GARDAN	Professeur, CReSTIC, Université de Reims
M. Hervé GUÉGUEN	Professeur, IETR, Supélec de Rennes
M. Bruno VIGARIO	Directeur technique de Real Games, Porto
Encadrant de thèse :	
M. François GELLOT	Maitre de conférences, CReSTIC, Université de Reims
Directeur de thèse :	
M. Bernard RIERA	Professeur, CReSTIC, Université de Reims

Remerciements

Les travaux de thèse présentés dans ce manuscrit ont été réalisés dans le groupe Automatique et Systèmes Hybrides du CReSTIC (Centre de Recherches en STIC) de l'Université de Reims Champagne-Ardenne. Je remercie, Monsieur le Professeur Janan ZAYTOON, directeur du CReSTIC, de m'avoir accueillie au sein du laboratoire, et d'avoir eu un mot d'encouragement dans mes recherches.

Je remercie vivement les rapporteurs de ce mémoire de thèse, Monsieur le Professeur Jean-Louis FERRIER du Laboratoire d'Ingénierie des Systèmes Automatisés (LISA) et vice président du conseil scientifique de l'Université d'Angers et Monsieur Jean-François PÉTIN du Centre de Recherche en Automatique de Nancy (CRAN). Je tiens à leurs exprimer mon extrême gratitude d'avoir accepté d'examiner et consacrer du temps à étudier mon travail. Nos multiples rencontres ont toujours été intéressantes en remarques et suggestions.

Ces quelques lignes me permettent de remercier deux personnes qui ont eu un rôle important dans cette thèse pour la mener jusqu'au bout. Mes « *deux Papas de thèse* » l'un qui m'a parfois fait pleurer mais a toujours tout fait pour tirer le meilleur de moi-même et l'autre qui a toujours été là pour sécher mes larmes en me poussant et me montrant que j'en étais capable. J'exprime donc une grande reconnaissance à Monsieur le Professeur Bernard RIERA qui a dirigé mes travaux, de m'avoir accordée sa confiance, de m'avoir orientée dans mes recherches, et de m'avoir poussée au maximum. Bien que sa grosse voix m'a beaucoup et m'impressionne toujours, je le remercie pour ces qualités scientifiques et le Gentil qui se cache en lui. Il a été agréable de pouvoir réfléchir ensemble sur de nombreux problèmes. Je remercie aussi Monsieur François GELLOT qui a grandement encadré cette thèse.. Il a toujours été là pour me soutenir, me protéger, prendre le temps d'écouter mes idées et relire mes ébauches de documents. Bien qu'il soit impressionnant, ces trois ans de thèse m'ont permise de découvrir une personne généreuse, humaine et toutes les vraies qualités qui se cachaient derrière « l'ours » (je devrais écrire le nounours). Qu'ils sachent tous les deux que je leurs suis extrêmement reconnaissante et qu'un jour, promis, je les tutoierai. Un petit mot aussi à leur épouse et leurs enfants, merci d'avoir été aussi compréhensif pour le temps que leur mari et papas m'ont consacrée.

Un merci particulier pour chacun des autres membres du jury. « Obrigado » à Monsieur Bruno VIGARIO, Directeur technique de Real Games de m'avoir fait l'honneur de faire le déplacement du Portugal pour participer à mon jury de thèse, et d'avoir pris le temps d'examiner mes travaux. Un grand merci aussi à Monsieur le Professeur Hervé GUÉGUEN à Supelec de Rennes pour les différents échanges que nous avons pu avoir sur mes travaux, et d'avoir examiner mon travail. Un merci à Monsieur le Professeur Yvon GARDAN, Directeur de l'ERT Gaspard Monge pour sa participation à mon jury et pour les différentes discussions sur l'ingénierie collaborative.

Un merci tout particulier pour le dernier membre du jury, Madame le Professeur Véronique CARRÉ-MÉNÉTRIER ; elle m'a fait le plaisir de participer à mon jury de thèse et m'a soutenue durant

toutes mes années d'étude à l'Université. Je ne la remercierai jamais assez pour ces grandes qualités scientifiques et humaines. Elle a toujours été là pour des conseils scientifiques ou des discussions de « filles ». Elle a tenu le rôle de « *Grande Sœur de thèse* » à qui l'on essaye de ressembler.

Durant ces trois ans de thèse, j'ai eu plaisir de me voir confier des enseignements au département GCE de l'IUT de Reims et au département EEA de la Faculté des Sciences Exactes et Naturelles. Je vous remercie, David, Anthony, Bernard, de m'avoir fait confiance. C'est l'occasion aussi de remercier les enseignants de collège, de lycée et universitaires qui m'ont donnée l'envie de cette thèse. À David CARTON, Jean-Claude CHAUDRON, Antonio DELUCAS, Olivier DUBOIS, François GELLOT, Thierry MOREL, Pierre TREBBIAS et beaucoup d'autres, un grand merci en espérant un jour devenir une merveilleuse enseignante comme vous, qui savez transmettre votre passion.

Je tiens également à remercier les membres de l'équipe SED – Supervision (Alexandre, Benoit, Bernard, François, Laurent, Moamar, Nadhir, Véronique) ; et les autres membres du CReSTIC. Vous avez tous participé à me construire une bulle, où je me suis sentie bien, en sécurité et que j'ai beaucoup de mal à quitter. J'en ai déjà remercié quelques uns, j'ai eu aussi la chance d'avoir des « *Grands Frères de thèse* » qui m'ont accueillie et soutenue : Alexandre, Cyril, Damien, Fabien, vous avez toujours eu la phrase pour me faire sourire ou rougir ; ainsi que des « *Petits Frères de thèse* » : Benoit et Laurent, contente d'avoir partagé ces moments de galère avec vous. J'ai eu aussi des « *Oncles et Tantes de thèse* » qui ont toujours été là pour un sourire, un petit moment de discussion : Dominique, Florence, Kevin, Lynda, Marie-Ange, Nicolas, Olivier, Sébastien, Stéphane, Sylvie. Et enfin des « *Cousins de thèse* » qui ont eu la patience d'essayer de m'apprendre à jouer au tarot et à la belote : Alex, Benoit, Fabien, François, Laurent, Régis… Je sais, ce n'est pas gagné !

Bien sûr, je ne remercierais jamais assez mes amis de m'avoir toujours supportée dans les moments de doute. Julien, Manu, Nathalie, et Richard, pour nos vacances, pour nos soirées de fous rires avec la sacrée bébête violette. Radia, copine de master et grande amie de thèse, pour nos longues discussions et les bons moments partagés avec la petite Maïssa. Vaness qui, de ma Haute Marne, m'a toujours téléphonée pour me transmettre sa bonne humeur. Chrystelle, merci pour ton écoute, ton calme, et pour les moments partagés avec ton petit Gabriel ; son sourire m'a redonnée de l'énergie dans la période de rédaction.

Bien entendu, tout ça n'aurait jamais été possible sans le soutien inconditionnel de mes parents qui ont toujours cru en moi. Les valeurs que vous m'avez transmises, ont fait que nous avons réussi ensemble cette thèse. Merci à mon petit frère, bien que plus grand que moi, d'être fier de sa grande sœur et qu'il sache qu'elle l'est tout autant, de sa réussite.

Et je finirai par toi, mon P'tit Loup, car bien que plus ensemble, tu as continué à croire en moi et à supporter mes angoisses.

À tous ceux à qui je tiens et à qui je ne le dis pas assez souvent.

« Et d'envoyer ailleurs valser les bagues et les cœurs en collier car quand on s'aime très fort c'est comme un trésor et ça, et ça vaut de l'or. »

Zazie

Table des matières

Introduction générale

Les travaux présentés dans cette thèse font suite à un financement par le Conseil Régional de Champagne-Ardenne qui porte un intérêt particulier aux formations supérieures professionnalisantes en sciences et techniques. Dans ce cadre, il a été proposé de permettre l'utilisation de Systèmes Automatisés de Production (SAP) par des apprenants, tout en assurant la sécurité des hommes et de l'installation.

La problématique de la sécurité des systèmes automatisés se retrouve bien évidemment aussi dans les industries. En effet, la complexité des systèmes automatisés actuels, ainsi que la demande croissante, de la part des industriels, pour que les équipements de la Partie Opérative (PO) soient disponibles et ne soient pas sources potentielles de dangers, font que la sûreté de fonctionnement des systèmes de contrôle-commande est une préoccupation de plus en plus présente dans le monde industriel.

Les installations automatisées sont aujourd'hui facilement commandables à distance grâce aux Technologies de l'Information et de Communication. Partant de cette constatation, une des possibilités envisagée par la Région Champagne-Ardenne, est la mutualisation inter et intra établissements d'enseignement supérieur des équipements pédagogiques lourds, de type atelier flexible, pour la formation en EEA (Électronique, Électrotechnique et Automatique). L'utilisation à travers le web de Parties Opératives réelles par des automaticiens expérimentés ou non et susceptibles de commettre des erreurs de conception de la commande amplifie les problèmes de sécurité. Cette thèse tente d'apporter des éléments de solutions en répondant aux deux questions suivantes :

- *Comment garantir la sécurité du système ?* Pour cela, la norme CEI 61508 propose un ensemble de méthodes pour la conception sûre des systèmes électriques, électroniques, et électroniques programmables assurant des fonctions de sécurité, ce qui englobe nombre de systèmes de contrôle-commande. Les méthodes préconisées par cette norme sont relatives tant aux composants matériels qu'aux logiciels. Lorsqu'un système a été développé selon les préconisations de cette norme, il peut être alors certifié. Contrairement aux composants matériels dont les fautes sont aléatoires, un logiciel de contrôle-commande ne peut générer que des fautes systématiques ; la faute d'un composant logiciel provient en effet d'une mauvaise interprétation des spécifications ou d'une erreur de conception ou de codage, et se produit chaque fois que ses conditions d'apparition sont réunies. C'est seulement les erreurs provenant de la partie logicielle du système de contrôle-commande qui ont été traitées dans cette thèse.
- *Comment prendre en compte le concepteur ?* Compte tenu du cas d'étude, il nous semble nécessaire de considérer le niveau de compétence et de connaissance de l'automaticien. De notre point de vue, cela peut se faire à deux niveaux. D'une part, lors de la phase de conception de la commande, le cahier des charges doit être en adéquation avec les savoirs et savoir-faire de l'automaticien. D'autre part, il est important de lui fournir des explications compréhensibles en cas d'erreurs de conception dans la commande.

Ce travail de recherche passe nécessairement par la vérification et la validation du programme de l'Automate Programmable Industriel (API) proposé par le concepteur. De nombreux travaux ont porté sur des méthodes formelles de vérification. Cependant, leur utilisation en milieu industriel, et en particulier dans le domaine de la conception des systèmes de contrôle-commande pour les systèmes manufacturiers reste peu utilisée. Ceci est dû à plusieurs problèmes : les langages formels devant être mis en œuvre (langages de description de systèmes de transitions, logiques temporelles…) sont délicats à maîtriser, les résultats fournis en cas de preuve négative sont difficiles à interpréter, et, par-dessus tout, ces méthodes d'analyse exhaustive sont souvent sujettes à des problèmes d'explosion combinatoire, même pour des systèmes de taille relativement modeste.

L'objectif de cette thèse est de proposer une approche, implantable dans un API, permettant d'éviter la détérioration de la PO en cas d'erreurs dans le programme de

commande, tout en tenant compte des caractéristiques de l'automaticien concepteur. Les travaux se sont intéressés aux trois points suivants :

- La validation de la commande peut se faire avant l'implémentation ou pendant l'exécution du programme. Dans le premier cas, nous parlerons de validation hors ligne et dans le second cas de validation en ligne de la commande. Les travaux présentés dans cette thèse proposent une approche pour chacun des cas.
- L'écriture des spécifications doit être vérifiée avant de les utiliser dans l'une ou l'autre des deux approches. En effet si les spécifications sont mal définies, la validation qui sera faite par la suite sera erronée et pourra conduire à la détérioration du système alors que les spécifications seront toutes validées.
- Enfin, le concepteur de la commande doit être inclus dans la boucle de validation pour prendre en compte ses savoirs et savoir-faire.

Ce manuscrit est composé de 5 chapitres.

Le premier chapitre présente le contexte de travail. Il est montré que la mise à disposition de systèmes automatisés réels pour des automaticiens au niveau d'expertise variable, d'une part nécessite une approche originale de l'automatisation prenant en compte le concepteur, et d'autre part soulève des problèmes théoriques originaux de validation et de vérification de la commande qui peuvent avoir des retombées possibles pour le monde industriel manufacturier. En effet, l'automaticien qu'il soit expérimenté ou novice est susceptible de commettre des erreurs aux conséquences parfois graves pour la sécurité. La commande et la sécurité des systèmes manufacturiers sont donc présentées sous l'angle industriel. Les voies de progrès proposées concernent la sécurisation de l'installation et l'adaptation du système au concepteur (et non le contraire). Il est mis en évidence l'importance pédagogique de conserver une vision globale de l'installation. Il est plus intéressant pour un apprenant de travailler sur le système entier que seulement sur un élément de la PO. L'étude bibliographique réalisée sur les travaux en synthèse de la commande, en vérification/validation et en surveillance, nous a conduits à retenir, pour la sécurité, une approche hors ligne par synthèse et une approche en ligne par filtre.

Le second chapitre s'appuie sur des travaux précédents de l'équipe « Système à Événements Discrets et Supervision » du groupe « Automatique et Systèmes Hybrides » du CReSTIC. L'approche proposée en synthèse pour l'obtention d'une commande sûre, robuste et sans blocage a été adaptée pour assurer d'une part que les modèles de PO et de

spécifications soient justes, et d'autre part que la commande proposée respecte le comportement maximum admissible par le système compte tenu des spécifications.

Le troisième chapitre propose une approche en ligne par filtre qui consiste à n'envoyer vers la PO que des commandes validées. La principale difficulté de ce type d'approche réside dans la spécification du filtre et dans son implémentation. Nous proposons une structure de filtre permettant de sécuriser le système, de vérifier le respect du cahier des charges et de prendre en compte le niveau de compétence de l'automaticien. Le travail présenté dans cette thèse porte principalement sur la partie sécurité du filtre. Les spécifications de sécurité (ce que le système ne doit pas faire indépendamment du cahier des charges) sont représentées au moyen de contraintes logiques. Un canevas de contraintes utilisable par l'expert est proposé. Les spécifications fonctionnelles (ce que le système doit faire) représentent le cahier des charges qui est lié à la vision de la PO donnée par l'expert à l'automaticien concepteur. Nous proposons de conserver une vision globale du système au moyen d'une encapsulation des fonctions.

Le quatrième chapitre propose une méthode pour assurer que les spécifications de sécurité du filtre sont correctement définies. En effet, une erreur au niveau de la définition des spécifications va entraîner une validation de la commande erronée et ainsi mettre le système en danger. L'approche proposée consiste, en considérant la commande la plus permissive possible, à vérifier formellement au moyen du model-checker UPPAAL que les contraintes sont nécessaires et suffisantes pour éviter toutes situations de détérioration de l'installation (éléments de PO et produit). En d'autres termes, cela revient à garantir un filtrage « robuste » sécuritaire des commandes.

Le cinquième chapitre présente deux applications du filtre « robuste » de commande. La première concerne un système réel de conditionnement de médicaments. L'approche de sécurisation de la PO et d'adaptation au concepteur est mise en œuvre. Celle-ci a été testée avec des expérimentations menées avec des automaticiens novices et plus expérimentés. Nous avons ainsi pu montrer son intérêt. La seconde application concerne une PO simulée. Il s'agit d'un magasin automatisé. L'idée est dans ce cas d'utiliser le filtre, non pas pour éviter les détériorations, mais pour apporter des explications. Cet exemple permet aussi de mettre en avant les limites de notre approche.

Le manuscrit termine en proposant des perspectives de recherche aussi bien sur le plan théorique que sur le plan pratique, en particulier dans le domaine industriel.

Chapitre 1 :
Commande et sécurité des systèmes automatisés : contexte industriel et pédagogique

1 Introduction

Le problème initial qui a conduit à ce travail de thèse concerne l'utilisation « sûre » de Systèmes Automatisés de Production (SAP) par des élèves automaticiens. L'apprentissage de l'automatisme passe par l'acquisition de savoirs (théories et connaissances) et de savoir-faire (compétences). Dans le cadre des formations professionnalisantes, le savoir-faire doit être en adéquation avec les réalités et les besoins du monde industriel. Pour cela, il est important que l'apprenant soit confronté à de vrais problèmes de commande et utilise donc des Parties Opératives (PO) «réelles» pilotées par des Automates Programmables Industriels (API). Toutefois, l'utilisation de systèmes automatisés «réels» pose plusieurs problèmes :

- les risques de sécurité et de casse sont importants,
- l'entretien exige du personnel qualifié,
- le coût.

De plus, l'utilisation des Technologies de l'Information et de la Communication (TIC) qui est aujourd'hui une réalité dans le domaine des automatismes, fait qu'il est simple techniquement de programmer un API et de superviser une installation à distance à travers le WEB (Riera et al, 2005 b, 2005 c ; 2006 b). Il est donc tout à fait possible d'utiliser de manière distante ou non, du matériel professionnel (API et PO) et des progiciels. Cela n'est

pas sans poser des interrogations d'ordre pédagogique lorsqu'il s'agit d'étudiants, mais aussi des problèmes de sécurité intéressants et originaux que nous avons tentés de résoudre dans cette thèse. En effet, d'une part l'apprenti automaticien est susceptible de commettre toutes les erreurs possibles dans sa commande et elles peuvent éventuellement entraîner des dégradations de la PO. D'autre part, l'utilisation d'un SAP à des fins pédagogiques dépend de la formation. Un novice qui découvre le monde des automatismes ne peut bien entendu pas faire ce que l'on demandera à un étudiant en master spécialisé en EEA (Électronique, Électrotechnique et Automatique).

Nous sommes partis du constat que l'automaticien, qu'il soit novice ou expert est susceptible de commettre des erreurs lorsqu'il conçoit et implémente une commande. Par conséquent, il nous a semblé plus pertinent de s'intéresser à la conception de la commande dans le monde industriel plutôt que de travailler sur des modèles d'apprentissage utilisés dans les Sciences de l'Éducation qui, de plus, ne relèvent pas de notre compétence.

Dans ce premier chapitre, le domaine de la commande et de la sécurité des SAP, et en particulier des systèmes manufacturiers, est analysé. Ce travail va permettre de mettre en évidence que la mise à disposition de systèmes réels pour des automaticiens en formation présente des spécificités qui, d'une part nécessite une approche originale de l'automatisation prenant en compte l'automaticien, et d'autre part soulève des problèmes théoriques de validation et de vérification de la commande.

Nous montrerons ainsi qu'un travail initialement orienté vers des applications pédagogiques, soulève des problèmes théoriques de recherche avec des retombées possibles pour le monde industriel.

La première partie du chapitre est une introduction à la commande des systèmes manufacturiers industriels et vise à préciser le domaine de notre étude. L'amélioration de la productivité des installations repose entre autres sur une meilleure disponibilité des équipements. Celle-ci s'obtient en évitant les dysfonctionnements qui peuvent relever d'une mauvaise prise en compte de l'analyse du risque et de la sécurité lors de la phase de conception de la commande. Ce point fait l'objet de la deuxième partie du chapitre.

La partie suivante est consacrée à l'utilisation de PO réelles ou simulées dans la formation des automaticiens et à la difficulté intrinsèque d'un problème de commande. Les originalités de la problématique vis-à-vis du contexte industriel et les voies de progrès sont présentées à partir d'une approche systémique de l'automatisation en considérant le système

Homme-Machine dans sa globalité (apprenant, enseignant, PO et partie commande). L'approche proposée vise à ne pas retirer les opérateurs de la boucle. Elle conduit à sécuriser le système mais aussi à l'adapter à l'apprenant en modifiant le degré d'automatisation de la PO et ses actions possibles. Cette approche permet, quel que soit l'automaticien, qu'il soit novice ou expert, de travailler sur le système complet. La fin de ce premier chapitre propose une synthèse bibliographique sur les approches possibles pour obtenir une commande sûre. Dans le cadre de ce travail de thèse, nous avons retenu les approches par synthèse et par filtre.

2 La commande des systèmes automatisés : contexte industriel

Le rôle d'un Système Automatisé de Production (SAP) est de *« transformer, sous certaines conditions un élément initial en élément final, afin que sa valeur, par rapport à certains critères, soit augmentée »* (De Bonneval, 1993). L'utilisation de ces systèmes dans le milieu industriel doit répondre à des objectifs imposés et peut être sujette à des dysfonctionnements. La compétitivité à l'échelle mondiale impose aux SAP d'être de plus en plus performants tant au niveau de leur conception que de leur exploitation. Pour cela, les cadences de production doivent être augmentées, les temps de maintenance réduits, et les systèmes flexibles pour s'adapter rapidement à la production de nouveaux produits. La productivité est liée entre autres à la qualité de la commande. Si celle-ci n'est pas sûre et robuste, elle peut entraîner des aléas de fonctionnement aux conséquences fâcheuses pour la sécurité des hommes et des équipements.

2.1 Les systèmes automatisés de production

Un système est un ensemble d'éléments en interaction organisé dans un environnement avec lequel il interagit pour réaliser une fonction qui lui est attribuée (Le Moigne, 1999). De manière générale, un SAP est constitué de deux parties essentielles (figure 1.1.) :

- La partie commande (PC) : qui assure l'envoi des ordres vers la PO et permet la communication avec l'opérateur
- La partie opérative (PO) : qui transforme d'une part à partir des ordres envoyés par la PC, la matière d'œuvre en lui additionnant une valeur ajoutée, et d'autre part renvoie un compte-rendu à la PC

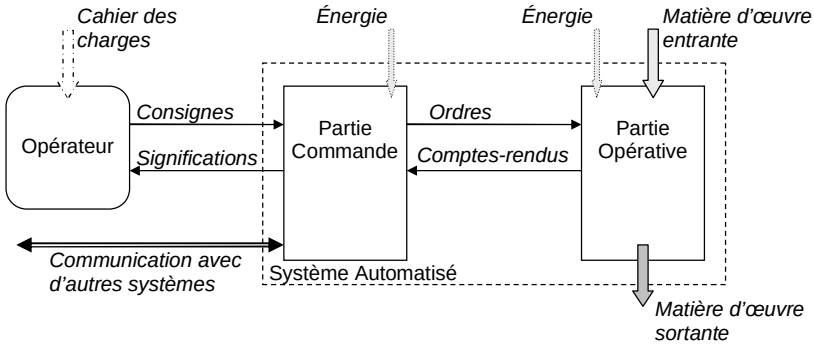


Figure 1.1. Structure et informations échangées dans un SAP (Théron, 2005)

La caractérisation d'un SAP peut se faire selon 3 points de vue utilisant des outils différents (Perrin et al., 2006) :

- Soit par la fonction à assurer qui exprime le besoin que le système doit satisfaire.
- Soit par la structure qui exprime le lien entre les différents composants matériels ou/et logiciels.
- Soit par la dynamique qui exprime le fonctionnement du système, c'est-à-dire la succession de mouvements, transformations à effectuer.

L'étude d'un SAP peut se faire en se plaçant du point de vue de la PO, ou de la PC. Comme le montre la figure 1.2., les informations en entrées et en sorties sont caractérisées par le point de vue retenu. Dans le cadre de nos travaux, le point de vue retenu est celui de la PC.

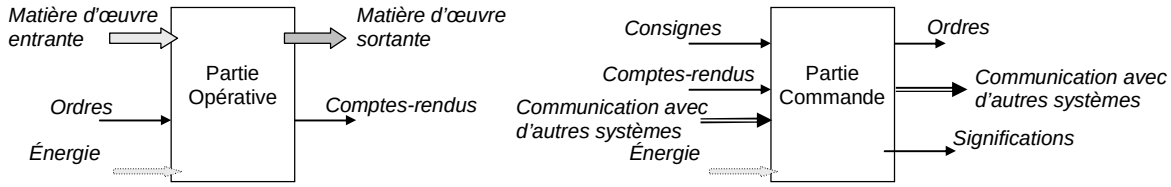


Figure 1.2. Entrées/Sorties en fonction du point de vue

Un SAP doit réaliser une ou plusieurs fonctions dans un environnement précis et il est nécessaire de prendre en compte un ensemble de contraintes visant à assurer l'intégrité des biens et des personnes. Ces contraintes, généralement décrites en termes de sécurité et d'écologie, sont imposées et définies par des normes (Mendez et al., 2003). Celles-ci peuvent être très nombreuses. Elles sont fortement dépendantes du procédé de fabrication utilisé (continu, batch, manufacturier) et des matières d'œuvre utilisées (alimentaires, corrosives, explosives, …).

Dans le cadre de cette thèse, nous considérons principalement les procédés manufacturiers qui présentent aujourd'hui un degré de complexité de plus en plus important dû au nombre important d'Entrées/Sorties et d'équipements ou éléments de PO inter connectés.

2.2 Les objectifs industriels

Les systèmes industriels actuels doivent assurer une meilleure productivité avec un coût moins élevé. Les automaticiens doivent donc concevoir des systèmes de plus en plus flexibles permettant un changement rapide de la production et des temps d'arrêt (entretien, maintenance) de moins en moins longs. Les SAP doivent donc répondre à des objectifs économiques, humains, techniques, et de pilotage (Pourcel, 1986) :

- Les objectifs économiques s'intéressent aux bénéfices financiers de l'entreprise et portent sur la diminution des coûts de fabrication et des stocks, la recherche de productions optimales,…
- Les objectifs humains s'intéressent à améliorer les conditions de travail de l'opérateur. En effet, les systèmes de production contiennent souvent des tâches dangereuses, répétitives, pénibles, et l'automatisation tente de réduire l'intervention humaine de ces tâches.
- Les objectifs techniques sont liés à la conception et à la transformation des processus de production. Les recherches portent sur :
    - La diminution de la durée du cycle de production en prenant en compte les durées administratives, d'approvisionnement, de transformation…
    - L'augmentation de la qualité des produits en assurant le niveau de qualité attendu par le client tout au long de la production
    - L'augmentation de la flexibilité, en permettant au système de s'adapter à un changement de production,
    - L'augmentation de la disponibilité du système de production en diminuant entre autres les temps de maintenance.
- Les objectifs de pilotage s'intéressent à la maîtrise des coûts de production et de stockage et à l'amélioration des temps de réaction en cas de perturbation.

Les travaux présentés dans ce manuscrit s'intéressent aux objectifs techniques en proposant une approche pour diminuer les dysfonctionnements et ainsi diminuer les interventions de maintenance. Le paragraphe suivant présente les différents dysfonctionnements pouvant survenir sur un SAP au niveau de la PO et au niveau de la PC.

2.3 Les dysfonctionnements possibles

Les dysfonctionnements sur un SAP sont liés à un événement non prévu qui vient modifier le fonctionnement de celui-ci (Pujo et al., 2002). Les dysfonctionnements peuvent être de deux types : externes ou internes.

Les dysfonctionnements externes peuvent venir :

- du client, qui peut modifier sa commande (changement de spécification sur le produit, diminution du délai de production, …),
- de la matière première, si les caractéristiques de celle-ci ne sont pas conformes,
- de l'environnement dû par exemple à une panne d'alimentation du SAP.

Les dysfonctionnements internes sont dus :

- à la partie matérielle du système de commande, c'est-à-dire une panne du calculateur, une mauvaise communication, …
- à la PO, en effet l'usure du système peut entraîner des détériorations (panne d'un capteur ou d'un actionneur par exemple),
- à la partie logicielle (commande) suite à une erreur de programmation ou de codage, …

Ces travaux de thèse se focalisent sur ce dernier point en essayant de détecter et d'éviter des dysfonctionnements de commande. Les dysfonctionnements liés à la partie logicielle sont différents de ceux liés aux composants matériels. En effet, les programmes de commande peuvent être modifiés par exemple lors d'interventions de maintenance. En revanche, ils ne sont pas sujets à des effets de vieillissement. L'origine des défauts logiciels est donc toujours humaine et peut se produire tout au long du cycle de vie du logiciel. Les moyens pour atteindre la sûreté logicielle se focalisent sur des méthodes et des démarches de spécification et de conception de logiciels (Laprie et al., 1995), (Frey et Litz, 2000). Nous verrons par la suite, que nos travaux proposent de sécuriser le système sans imposer une démarche au concepteur de la commande et reste applicable lors de modifications partielles du code de la

commande par des agents de maintenance par exemple. Les défaillances au niveau de la commande peuvent venir :

- d'un mauvais comportement de l'opérateur qui fait évoluer le système vers un état dangereux,
- d'une mauvaise conception du programme de commande. Celle-ci peut avoir été modifiée pendant le cycle d'exploitation du système de production. En d'autres termes, une commande « sûre » au démarrage d'une installation ne l'est plus forcément au bout de quelques années de production.

Les erreurs de l'opérateur peuvent provenir : d'une mauvaise vision du système, d'une mauvaise compréhension de ce qu'il doit faire, ou elles peuvent être intentionnelles. Au niveau des erreurs de conception de commande, les erreurs peuvent provenir d'une mauvaise vision du système, ou de la non-prise en compte de certaines spécifications.

Dans le cadre industriel, la problématique de détection des erreurs de commande se retrouve donc au niveau de la conception de la commande, mais aussi au niveau des tâches de maintenance ou encore d'amélioration du système, où l'agent de maintenance peut être amené à modifier une partie du programme, sans connaître complètement le système. Les TIC permettent une intervention à distance sur les systèmes et soulèvent plus fortement les problèmes d'erreurs dues à une mauvaise vision (ou perception) du système. Le paragraphe suivant présente l'utilisation des TIC dans le cadre de la maintenance et l'évolution de la télé maintenance vers l'e-maintenance.

2.4 Les technologies de l'information et de la communication et automatismes

Les TIC permettent d'accéder simplement à distance à des systèmes, aussi bien au niveau de la PO qu'au niveau de la PC. Ces nouvelles possibilités permettent d'une part d'avoir des informations sur le système distant et d'autre part d'agir sur celui-ci.

De plus en plus d'entreprises utilisent des techniques de contrôle et de maintenance à distance, pour surveiller l'état de fonctionnement de leurs systèmes. Ce concept de télémaintenance fait l'objet de la norme AFNOR (AFNOR, 2001) : *Maintenance d'un bien exécuté sans accès physique du personnel au bien.* Un système de télémaintenance est toujours composé d'au moins deux parties distinctes (Ivanov et al., 2003). Le centre expert de maintenance, et les sites à maintenir. Les équipements de plusieurs sites sont surveillés et, en cas de panne, le centre expert affecte la meilleure compétence en fonction de la maintenance à

réaliser. La télémaintenance a donc pour objectif de permettre d'effectuer, rapidement et à distance, un grand nombre d'opérations. Cette gestion « réactive » des tâches de maintenance a pour but de maximiser la disponibilité des équipements (Adjallah et al., 2005).

La e-maintenance est une forme évoluée de télémaintenance intégrant des fonctions avancée de supervision (détection, diagnostic, surveillance, reprise, reconfiguration, …) (Sefiane et al., 2005). Ainsi, tout au long du processus de maintenance, il est possible d'accéder à de l'information et non plus uniquement à des données. Les travaux de (Chatelet et al., 1999 ; Chatelet, 2000) ont présenté deux architectures possibles pour la e-maintenance :

- La première est composée d'un ordinateur doté d'un serveur web ou d'un API possédant un coupleur web, appelé communicateur Internet. C'est le seul équipement qui peut dialoguer avec l'équipement distant. Il concentre toutes les informations provenant des capteurs contrôlant le processus.
- La deuxième solution consiste à donner l'accès direct à des équipements, ils deviennent alors des serveurs web où l'opérateur peut agir directement.

Il nous semble que la possibilité d'agir à distance sur le système amplifie les problèmes de sécurité qu'il convient de prendre en compte dans la phase de conception et de validation de la PC. Ce point fait l'objet du paragraphe suivant.

3 Analyse du risque et sécurité des systèmes manufacturiers

Pour définir la sécurité d'un système, nous avons repris la norme (ISO CEI Guide 51, 1999) qui définit la sécurité comme l'absence de risques inacceptables où le risque est la combinaison entre la probabilité d'un dommage et sa gravité. Les risques peuvent provenir d'une défaillance système ou dus à une faute occurrente à une erreur humaine (Laprie et al., 1995). Comme nous l'avons vu, la problématique posée dans ce manuscrit s'intéresse aux défaillances dues à une erreur au niveau de la conception du programme de commande.

La fiabilité, la maintenabilité, la disponibilité et la sécurité (FMDS) se regroupent dans le concept de sûreté de fonctionnement (SdF). L'évaluation de la sûreté de fonctionnement d'un système consiste à analyser les défaillances des composants pour déterminer leurs causes et estimer leurs conséquences sur le service rendu par le système. Ces évaluations sont

définies par des analyses inductives ou déductives des effets des pannes, des dysfonctionnements et des erreurs d'utilisation

3.1 Concepts de la sûreté de fonctionnement

Dans les travaux de (Villemeur, 1988) la sûreté de fonctionnement est définie comme un ensemble de méthodes capables de déterminer, évaluer, prévoir, mesurer et maîtriser les risques. Dans les travaux de (Laprie et al., 1995 ; Laprie et al., 2004), la sûreté de fonctionnement définit la confiance que l'on peut avoir dans le système.

La Sûreté de Fonctionnement est présente tout au long du cycle de vie d'un produit ou d'un système en vue de maîtriser au mieux les risques, les délais et les coûts pour obtenir une qualité conforme aux exigences.

Elle est intégrée dans les différents modèles conceptuels de gestion de projet, en particulier dans le domaine du développement logiciel. Le plus connu des modèles est le cycle en V proposé dans les travaux de (Forsberg et al., 1991) où des étapes de validation et de vérification sont introduites pour chaque activité de spécification et de conception. Le cycle en V est un modèle conceptuel de gestion de projet mis en place suite au problème de réactivité du modèle en cascade (Royce, 1970). Il a pour objectif, en cas d'anomalie, de limiter un retour aux étapes précédentes et non au début du cycle. Ces travaux sont prolongés par une approche « Dual Vee Model » (Forsberg et al., 2005) en proposant une approche à deux niveaux : l'un au niveau systèmes et composants et l'autre reprenant le cycle en V classique pour chaque élément du système.

Un second modèle plus général et plus centré sur l'analyse des risques est celui de (Boehm, 1986) appelé cycle en spirale. Chaque cycle de la spirale représente un pas de définition du système et à chaque cycle, les quatre étapes suivantes sont exécutées :

1. Développer une série de prototypes pour identifier les risques en commençant par le plus grand,
2. Utiliser un modèle en V ou en cascade pour implémenter chaque cycle,
3. Si un cycle concernant un risque a été achevé avec succès, évaluer le résultat du cycle et planifier le cycle suivant,
4. Sinon, si un risque n'a pu être résolu, terminer le projet.

On constate que quelle que soit l'approche de conception, il est nécessaire d'évaluer et de prendre en compte les risques potentiels.

3.2 L'évaluation des risques

Dans le domaine industriel, le SIL, niveau d'intégrité de sécurité (Safety Integrity Level), permet de spécifier les prescriptions concernant l'intégrité de sécurité des fonctions de sécurité à allouer aux systèmes relatifs à la sécurité qu'ils soient électriques (E), électroniques (E) ou encore électroniques programmables (PES).

Le SIL dépend de la sollicitation à laquelle le système est soumis. Dans le cas des équipements électroniques programmables (PES) des systèmes manufacturiers, c'est l'indice de forte sollicitation qui est utilisé. La figure 1.3. présente les 4 niveaux d'intégrité de la sécurité définis en fonction de la probabilité de défaillance d'un système.

Niveau d'intégrité de sécurité	Modes de fonctionnement	
	Faible sollicitation (probabilité de défaillance par sollicitation)	Forte sollicitation (probabilité de défaillance par heure)
4	$\geq 10^{-5}$ à $<10^{-4}$	$\geq 10^{-9}$ à $<10^{-8}$
3	$\geq 10^{-4}$ à $<10^{-3}$	$\geq 10^{-8}$ à $<10^{-7}$
2	$\geq 10^{-3}$ à $<10^{-2}$	$\geq 10^{-7}$ à $<10^{-6}$
1	$\geq 10^{-2}$ à $<10^{-1}$	$\geq 10^{-6}$ à $<10^{-5}$

Figure 1.3. Niveau d'intégrité de la sécurité (SIL)

Pour être certifié à un niveau SIL donné, le développement du système doit utiliser des méthodes de réduction ou de prévention des risques. La norme IEC 61508 (2002) indique une démarche :

1. Choix du niveau SIL en fonction de la gravité des défaillances système,

2. Identification de méthodes de réduction des risques (figure 1.4.). Soit aucune validation n'est nécessaire, soit elle est recommandée (R), soit elle est hautement recommandée (HR) en fonction du niveau SIL souhaité (Gourcuff, 2007),

	SIL1	SIL2	SIL3	SIL4
Diagramme de flux de données	R	R	R	R
Automates à états finis		R	HR	HR
Méthodes formelles		R	R	HR
Modélisation du fonctionnement		HR	HR	HR
Réseaux de Petri temporels		R	HR	HR
Prototypage/ animation	R	R	R	R
Diagrammes de structures	R	R	R	HR
Tests probabilistes		R	R	HR
Simulation / Modélisation	R	R	HR	HR
Tests fonctionnels et boites noires	HR	HR	HR	HR

Figure 1.4. Niveau d'intégrité de la sécurité (SIL)

3. Choix de la / des méthode(s),

4. Évaluation des résultats : Les méthodes retenues permettent d'identifier des défaillances systématiques possibles. Ces résultats sont analysés afin d'identifier la cause de ces défaillances.

5. Actions à entreprendre en cas de défaillance : Les causes des défaillances sont contournées, modifiées ou supprimées afin de ne plus apparaître.

Ensuite une certification du procédé est réalisée par un organisme de contrôle et de normalisation, comme le bureau Veritas ou le Technischer Uberwachungs-Verein (TUV). Celui-ci s'assure que la démarche de mise en place pour réduire les risques a bien été suivie et qu'elle répond au niveau SIL désiré.

Dans l'industrie, deux types de dispositifs sont proposés pour réduire les risques liés à l'exploitation. La sécurité peut être améliorée au niveau matériel et/ou logiciel. La sécurité au niveau matériel se fait à travers la mise en place de carters de protection, de barrières immatérielles, de commandes bimanuelles (Evrot, 2008). La sécurité au niveau logiciel est assurée par des API de Sécurité (APIdS). Il existe deux types d'APIdS. Les premiers assurent la disponibilité d'un processus malgré une défaillance interne (logicielle) au moyen d'une architecture redondante. Les seconds calculent de façon redondante la commande, et vérifient qu'il y a identité de résultats avant de l'envoyer sur le système réel. En cas de différence le système est mis dans une situation de repli.

Les travaux proposés dans cette thèse ne s'intéressent pas à une défaillance d'exploitation de la PC mais à des défaillances dues à des erreurs de conception dans la commande. L'utilisation des APIdS ne répond donc pas à nos attentes.

Pour comprendre et analyser les situations possibles de mauvais fonctionnement (i.e. défaillances), il est nécessaire au préalable de réaliser une analyse fonctionnelle pour définir, ordonnancer, et hiérarchiser les fonctions du système (AFNOR NF X 50-151, 1991). L'analyse fonctionnelle est une étape fondamentale dans les démarches de conception et de prise en compte des risques.

3.3 Analyse fonctionnelle

L'analyse fonctionnelle débute beaucoup de projets de création d'un système (produit, procédé, processus). Elle permet de définir les fonctions principales, les fonctions secondaires et les fonctions contraintes d'un système. L'analyse fonctionnelle s'effectue habituellement en deux étapes :

- Une dite externe permettant de définir les limites matérielles, les différentes fonctions et opérations réalisées, et les configurations d'exploitation du système.
- L'autre dite interne permettant de réaliser une décomposition arborescente et hiérarchique du système en éléments matériels et/ou fonctionnels. C'est à ce niveau que sont déterminées les fonctions présentes dans le système.

Les méthodes les plus connues de modélisation pour l'analyse, la conception et la spécification de systèmes automatisés sont SADT (Ross, 1977 ; Marca, 1988 ; IGL 2006), SA-RT (Hatley et al., 1991), MERISE (Chen, 1976 ; 1990 ; Tardieu, 2000),…

En parallèle de l'analyse fonctionnelle, le système peut être étudié de manière structurelle. L'analyse orientée objet (OOA en anglais) est une réflexion faite en amont de la création d'une application. Elle a pour but de modéliser l'ensemble des éléments d'un système, c'est-à-dire définir la structure. Le modèle d'un système, composé d'objets qui peuvent avoir des liens entre eux. Les méthodes les plus connues d'analyse d'un système par approche orientée objet sont OMT (Rumbaugh et al., 1993), UML (Booch et al., 1998 ; Muller, 2004), …

Suite à l'analyse fonctionnelle d'un système ou structurelle, il est possible d'effectuer l'analyse dysfonctionnelle pour déterminer les situations à risques et ainsi définir les priorités à respecter.

3.4 Analyse de dysfonctionnement et AMDEC

Pour déterminer le mauvais fonctionnement d'un système, il existe différentes méthodes : Analyse de Priorités des Risques, Arbre de défaillances, … Une des plus connues et des plus utilisées est l'AMDEC.

L'Analyse des Modes de Défaillance, de leurs Effets et de leur Criticité (AMDEC) est une méthode spécifique de la sûreté de fonctionnement adaptée à l'analyse de systèmes statiques (systèmes au sens large composés d'éléments fonctionnels ou physiques, matériels, logiciels, humains ...) s'appuyant sur un raisonnement inductif (causes - conséquences), pour l'étude organisée des causes, des effets, des défaillances et de leurs criticités.

Cette technique a été utilisée pour la première fois au milieu des années 60 pour l'analyse de la sécurité des avions (Villemeur, 1988) et son utilisation se limitait au début à des études de fiabilité sur le matériel (notamment dans le domaine de l'aéronautique pour tenter de limiter la fréquence et la gravité des accidents aériens). Elle est désormais extrêmement répandue dans tout ce qui a trait à la sécurité, la maintenance ou la disponibilité. Jadis cantonnée au «simple» domaine du matériel, elle sert désormais de référence en matière de système, de fonctionnel et de logiciel. Malgré sa popularité et son utilisation dans de nombreux domaines, l'AMDEC n'est pas sans inconvénients. Ainsi, la méthode n'est pas adaptée à l'analyse des défaillances multiples (systèmes redondants) (Faucher, 2004).

Comme dans toutes les méthodes de la Sûreté de Fonctionnement, la définition du système est fondamentale. Le principe de base de l'AMDEC est de s'intéresser aux effets des défaillances pour un « système » particulier. Par exemple, dans la construction automobile, l'AMDEC « Produit » s'intéresse aux effets des défaillances du véhicule pour le client final. En revanche, la vision est différente dans le cas des AMDEC « Moyens de Production » où on s'intéresse aux effets des défaillances des lignes de production pour le constructeur automobile.

3.5 Synthèse

Le monde informatique a adopté depuis longtemps les méthodes conceptuelles et les outils de la gestion de projet. En revanche, même si les outils évolués de conception d'automatismes existent, ils ne sont pas forcément utilisés. Certaines industries à risques comme le nucléaire ou les transports (Boulanger, 2006) ont recours soit à la génération

automatique de code avec des compilateurs certifiés, ou aux techniques de vérification formelle (méthode B (Abrial et al., 1996) par exemple) pour s'assurer que le code généré répond aux spécifications fonctionnelles décrites dans le cahier des charges.Il semble que cela ne soit pas encore le cas dans l'ensemble du secteur manufacturier (Peyrucat, 2005) où il reste « trop d'à peu près dans la réalisation des automatismes » (Peyrucat, 2003). Les coûts d'étude restent importants et la réutilisation du code difficile, alors que l'amélioration de la productivité impose des temps de développement de plus en plus courts, et des automatismes de plus en plus fiables en vue de réduire les temps d'arrêts.

Pourtant les plates-formes logicielles pour la conception et la validation des systèmes automatisés industriels et des outils performants existent en vue de réduire les temps de développement, améliorer la qualité et la sûreté des applications, maîtriser les risques liés à la mise en service d'installations et faciliter l'exploitation et la maintenance des applications. A titre d'exemple, ControlBuild[1] est une plateforme logicielle complète et homogène à destination des automaticiens et des informaticiens industriels. ControlBuild est basé sur une démarche model-based-design, et permet de modéliser, simuler, générer et valider des applications d'automatismes.

Nous constatons que le métier de l'automaticien est en train d'évoluer. Il doit maintenant passer plus de temps à traiter et tracer les spécifications de la commande et ses évolutions qu'à développer du code. Cela passe par une prise en compte plus importante de l'analyse fonctionnelle et l'utilisation d'outils évolués de conception d'automatismes permettant une conception assistée automatisée

La maîtrise de la complexité des systèmes manufacturiers passe désormais par l'utilisation d'outils logiciels permettant aux automaticiens de développer des programmes de plus en plus fiables et flexibles en optimisant la capitalisation et le transfert du savoir-faire.

La commande des systèmes manufacturiers a donc considérablement évolué ces 15 dernières années. Les TIC ont conduit à des solutions supprimant les barrières entre les différentes applications logicielles de l'entreprise (M.E.S : Manufacturing Execution System, GMAO, E.R.P : Enterprise Ressource Planning). Les formations au métier d'automaticien, pour suivre cette évolution, se doivent de développer d'une part les aspects conceptuels et

[1] ControlBuild développé par le groupe GEENSYS

d'autre part les aspects pratiques liés à l'utilisation de systèmes automatisés réels. Ce point fait l'objet du paragraphe suivant.

4 La formation aux automatismes et l'utilisation de Parties Opératives pour la formation

Automatiser, c'est aussi permettre d'accéder et de diffuser à la bonne personne, la bonne information au bon moment. Ces modifications du métier de l'automaticien sont prises en compte aujourd'hui dans la formation académique.

4.1 L'enseignement des automatismes

L'enseignement des automatismes n'est plus aujourd'hui uniquement centré sur la « programmation GRAFCET d'API » (Bajic, 2006). Les automatismes industriels sont aujourd'hui complexes, structurés en réseaux et ouverts. Comme le souligne (Bajic, 2006), ils reposent sur le triptyque Automatismes – Réseaux – Supervision (ARS), et il n'est plus possible d'aborder un élément d'analyse ou de réalisation d'un automatisme selon un de ces 3 axes sans faire une référence aux 2 autres. De plus, l'interconnexion automatisme/informatique est devenue une nécessité dans le mode de fonctionnement des entreprises. L'élaboration d'architectures intégrant le MES (Manufacturing Execution System) et l'ERP (Enterprise Ressource Planning) accroît le besoin de transfert bidirectionnel de données. Le développement des interfaces entre les différents niveaux de l'entreprise (de la production vers la gestion) permettant de diffuser les données en provenance du procédé relève des compétences de l'automaticien qui, contrairement à l'informaticien a la connaissance de la PO et de la PC.

Par conséquent, en plus d'être spécialiste en commande, l'automaticien doit avoir des connaissances et des compétences dans des domaines aussi variés que la communication industrielle et les réseaux, la supervision et les bases de données.

La formation de l'automaticien ne peut pas être que théorique (savoirs). Il est indispensable qu'il puisse mettre en application sur des Parties Opératives (réelles ou simulées) les concepts et acquérir aussi des savoir-faire. Cela est d'autant plus vrai qu'il est

nécessaire d'avoir une vision holistique des automatismes industriels selon le triptyque Automatisme – Réseaux – Supervision (ARS).

4.2 Utilisation de PO réelles et simulées

Le savoir-faire acquis par les apprenants doit être si possible en adéquation avec les réalités et les besoins du monde industriel. Pour cela, il est important que l'élève automaticien soit confronté à des problèmes « réels » et utilise donc des parties opératives «réalistes». Celles-ci peuvent être utilisées à tous les niveaux et dans les 3 axes ARS. Elles imposent aux apprenants de les manipuler comme ils le feraient dans une activité professionnelle (Riera, 2001). Le fait de travailler sur un système réel est souvent plus motivant (Riera et al., 2005 ; 2006a ; Marangé et al., 2006a).

L'utilisation de systèmes automatisés «réels» présente cependant plusieurs inconvénients aussi bien pour l'enseignant que pour l'apprenant, parmi lesquels nous pouvons citer :

- le coût élevé des maquettes,
- les risques de sécurité et de casse,
- l'entretien de l'installation exige du personnel qualifié,
- l'enseignant doit être vigilant (stress),
- la configuration est figée (difficulté de simuler les défaillances par exemple),
- les temps de préparation des travaux pratiques sont importants,
- le manque de disponibilité et de fiabilité,
- …

C'est pour toutes ces raisons que la simulation, d'un point de vue pédagogique, peut être complémentaire aux systèmes réels. Riera (1999) distingue la simulation « simple » et la maquette virtuelle. Cette dernière repose en plus sur un rendu réaliste de la structure, des fonctions et du comportement du système mais aussi sur un respect de la dynamique. Actuellement, la tendance est à la simulation d'environnements industriels 3D très réalistes. Nous utiliserons d'ailleurs un logiciel de ce type (ITS PLC) dans le chapitre 5 pour la validation expérimentale.

Il est à noter que l'utilisation de la simulation et de la 3D s'est particulièrement développée dans le domaine du PLM (Product Life Management) (Riera et al., 2008). A titre

d'exemple, le groupe Dassault Systèmes développe et commercialise des logiciels d'application PLM et des services qui anticipent les processus industriels de demain et offrent une vision 3D de l'ensemble du cycle de vie d'un produit, de sa conception à sa maintenance en cours de vie. La gestion du cycle de vie du produit (PLM) consiste à appliquer des solutions industrielles collaboratives au développement des produits, depuis la phase de conception jusqu'à la fabrication. Ainsi, le logiciel DELMIA permet d'élaborer numériquement les processus de fabrication permettant ainsi d'optimiser la production réelle. Avec DELMIA Automation, les programmeurs peuvent valider et déboguer leur code API sur toutes sortes de dispositifs – outillages, robots, systèmes de sécurité et appareils électriques – plusieurs mois avant de procéder à leur intégration physique. Il s'agit là de simulations «parfaites» qui relèvent davantage de la boite à outils de l'automaticien (cf. paragraphe 3.5). On constate aujourd'hui l'importance de plus en plus grande que prend la simulation dans le développement de produit et la fabrication industrielle.

Grâce à la simulation on dispose à moindre coût de Parties Opératives simulées aux nombreuses fonctionnalités pour : tester avant d'implémenter, faire des TD interactifs mais aussi du TP à distance. Pour les étudiants, la maquette virtuelle est attrayante car l'aspect ludique est indéniable. 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, le système réel va développer le sens physique de l'étudiant. Par conséquent, l'approche que nous préconisons consiste à ne pas remplacer le système réel mais à le rentabiliser, d'un point de vue pédagogique, au maximum.

Au niveau de l'enseignement, il est de plus en plus courant de voir la mise en place de TP à distance, de cours à distance que ce soit dans les domaines de la physique (Guillon, 1996 ; Beney, 1998), de l'électronique (Saliah, 1999), de l'automatique (Metzger, 2004) ou de l'automatisme (Ginestine, 1992). L'étudiant peut ainsi acquérir un savoir ou savoir faire via Internet.

Dans le domaine du contrôle automatique de processus continus, l'utilisation de laboratoires, réels ou virtuels, à distance est courante dans le cadre de l'enseignement. Par exemple les travaux de (Metzger, 2004) proposent l'enseignement de la commande de dispositifs distribués via Internet sur des systèmes virtuels. D'autres travaux de (Aliane et al., 2006) proposent un laboratoire à distance : LABNET permettant la formation sur la régulation et les asservissements sur des maquettes réelles. Dans le même domaine, nous pouvons aussi citer les travaux de (Lunt et al., 2000 ; Colace et al., 2004 ; Lottin et al.,

2005 ; Tauvel et al., 2005) qui proposent aussi le contrôle de systèmes réels à réguler à distance.

En revanche, la bibliographie présente peu d'articles concernant l'enseignement des Systèmes à Événements Discrets (SED) et l'utilisation locale ou distante de SAP réels ou simulés, pour l'apprentissage de la programmation des API. On trouve par exemple les travaux de (Hassapis, 2003) qui proposent d'utiliser des simulateurs de systèmes informatiques distribués et d'API intégrés dans un livre électronique interactif. Les travaux de (Bellmunt, 2006) visent à rendre disponibles des plateformes via Internet afin de permettre l'utilisation en TP via du e-learning. Toutefois, ces approches ne tiennent pas compte du problème de la sécurité du système et de la manière d'adapter l'utilisation du système réel au niveau de l'apprenant. En effet, ces systèmes sont généralement conçus avec des composants industriels et une erreur de conception dans la commande peut entraîner des problèmes de sécurité.

Dans le cadre de ce travail de recherche, nous nous sommes principalement intéressés à la problématique de la conception de la commande des SAP. En d'autres termes, nous considérons la situation où l'automaticien est un concepteur (novice ou expert), qui dispose d'un cahier des charges et qui doit réaliser le programme à implémenter dans un ou plusieurs API pour piloter le système manufacturier considéré.

Il est évident qu'un problème de commande (i.e. cahier des charges) doit être adapté au concepteur. L'analyse, les connaissances et les compétences requises ne sont pas les mêmes pour un apprenant qui débute, que pour un automaticien expert. Toutefois, dans tous les cas, l'erreur humaine est possible et peut conduire à une dégradation de la PO.

Le paragraphe suivant tente de définir et de formaliser la complexité d'un problème de commande.

4.3 Paramètres liés à la difficulté d'un problème de commande

Pour proposer des cahiers des charges adaptés au concepteur, il est nécessaire de définir sa difficulté pour pouvoir la moduler.

Le concept de difficulté est à relier au concept de complexité qui est caractérisé par le nombre élevé de variables, par l'interaction entre ses variables… La perception de la complexité du système, son analyse, et sa modélisation sont spécifiques aux objectifs que le

« modélisateur » s'est fixé. Les travaux de Lind (Lind, 1994) considèrent que les systèmes de production peuvent être définis et analysés selon deux axes de décomposition : « Moyens-But » et « Tout-Parties » (figure 1.5.). L'axe « Moyens-But » est utilisé pour représenter le système par des buts (du concepteur ou de l'utilisateur), des fonctions et des composants. Ensuite, les descriptions des buts, des fonctions et des composants peuvent être réalisées selon différents niveaux de l'axe « Tout-Parties ». On retrouve ici la notion de modélisation fonctionnelle hiérarchique.

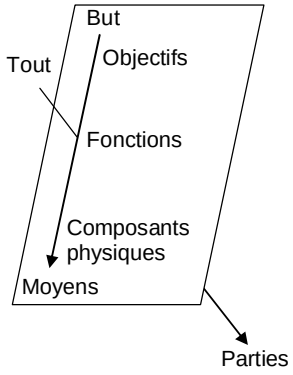


Figure 1.5. Vue d'un système par (Lind, 1994)

Il est possible de relier cette vision du système au niveau de difficulté (ou de complexité) d'une spécification de commande. Une PO est composée et perçue au travers d'un ensemble de capteurs et d'actionneurs fixés et difficilement modifiables. La solution la plus évidente pour simplifier le système et donc sa commande est de ne pas considérer certains éléments de PO. Toutefois, la complexité de la commande n'est pas directement liée à la dimension de la PO mais au cahier des charges. Nous considérons uniquement la difficulté ressentie et définie par l'expert (l'enseignant) et qui se base sur une solution considérée comme optimale. On notera que la commande proposée par le concepteur n'est pas forcément similaire à celle de l'enseignant et peut présenter un degré de difficulté plus élevé. Le travail d'analyse de la solution reste bien entendu à la charge de l'expert et ne fait pas l'objet de ce travail.

De notre point de vue, la difficulté d'une spécification de commande peut s'exprimer au travers de 3 paramètres, dépendants les uns des autres, et caractérisant la commande optimale qui, pour l'expert, répond au cahier des charges qui sont la dimension, la hiérarchisation et la synchronisation entre les éléments (Marange et al., 2007 a ; 2007 f).

4.3.1 Granularité ou dimension

La dimension de la commande est directement liée au nombre de sous-ensembles devant être commandés. Elle dépend au niveau le plus bas de la décomposition, du nombre de capteurs et d'actionneurs nécessaires pour concevoir la commande décrite par le cahier des charges. Plus la granularité est fine, plus l'effort d'analyse est important pour l'automaticien et plus le niveau de difficulté est élevé. De même, plus les compétences du concepteur dans le domaine de la commande sont importantes et plus il est capable de travailler au niveau le plus bas de l'axe « Moyens-But », c'est-à-dire : au niveau des capteurs et des actionneurs.

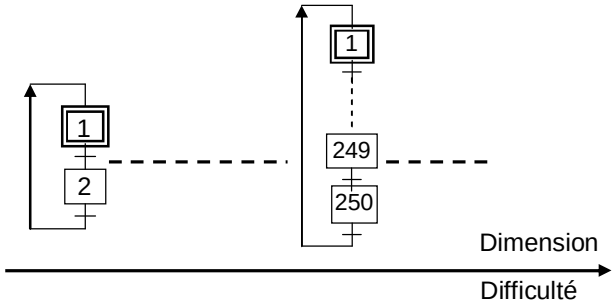


Figure 1.6. Premier paramètre de difficulté : dimension

4.3.2 Hiérarchisation

La hiérarchisation dans la commande d'un système apparaît lorsque l'on tient compte des différents modes de marche. En effet, la gestion d'un cycle « normal » sans tenir compte des divers modes de fonctionnement nécessite une commande séquentielle simple. En revanche, si le cahier des charges est « complet » et intègre plusieurs modes de marche, la solution nécessite une commande hiérarchisée. Dans la pratique, la hiérarchisation apparaît par exemple au travers des ordres de forçage de Grafcet (IEC, 2002 a).

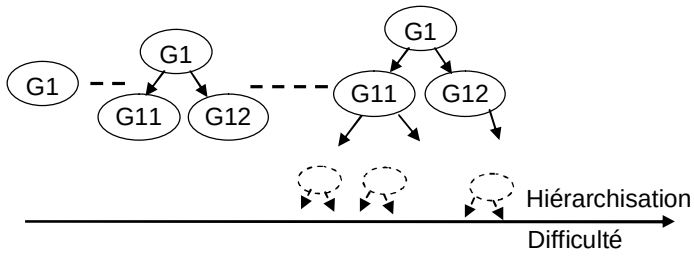


Figure 1.7. Deuxième paramètre de difficulté : Hiérarchisation

4.3.3 Synchronisation

Les deux paramètres précédents vont faire apparaître des synchronisations dues à la définition de la granularité du système (évolution simultanée d'événements) et à la hiérarchisation engendrée par des priorités entre les différents modes de fonctionnement. Toutefois, les problèmes de commande peuvent aussi nécessiter pour être résolus des structures particulières de gestion d'événements ou de ressources communes. La figure 1.8. indique, selon notre point de vue, quelques structures en Grafcet souvent rencontrées et leur niveau de difficulté théorique.

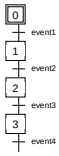
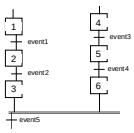
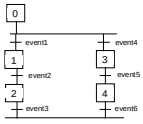
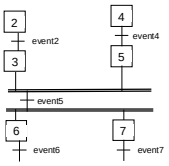
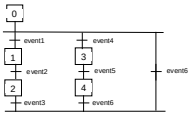
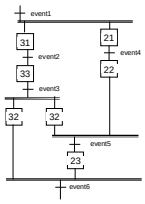
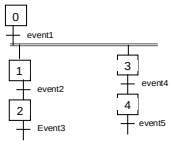
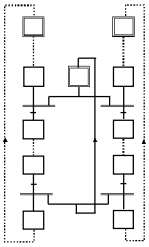
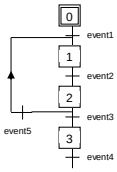
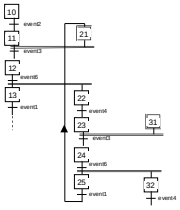
Cycle d'une séquence simple ***(très facile)***		*Synchronisation de séquences* ***(moyen)***	
Sélection de séquences ***(facile)***		*Synchronisation et activation de séquences parallèles* ***(moyen)***	
Saut d'étapes ***(facile)***		*Si événement Alors …* ***(difficile)***	
Activation de séquences parallèle ***(facile)***		*Gestion ressource commune* ***(très difficile)***	
Reprise de séquence ***(moyen)***		*Séquences alternées* ***(très difficile)***	

Figure 1.8. Troisième paramètre de difficulté : synchronisation

C'est sur ces trois paramètres que l'expert (i.e. le formateur) va pouvoir jouer pour adapter le niveau de difficulté du cahier des charges au niveau de compétence de le concepteur.

4.4 Synthèse

Il est intéressant de constater que les problèmes de conception de la commande des automatismes présentent des similitudes au niveau de l'industrie et de la formation. L'automaticien est dans les 2 cas le concepteur des programmes des API. Dans les deux domaines, des erreurs de conception de la commande peuvent entraîner des détériorations de

la PO, et le programme de l'API peut faire l'objet de modifications tout au long du cycle de vie de l'installation.

Aujourd'hui, les outils évolués de conception d'automatismes ne sont pas suffisamment utilisés dans l'industrie manufacturière et les modifications de la commande trop importantes pour garantir la sécurité des installations.

Toutefois, dans le monde professionnel, le concepteur, qui est éventuellement un intégrateur, est toujours considéré comme un expert. En revanche, dans la formation, l'expert est le formateur, et le concepteur est l'apprenant. Ce dernier peut dans certains cas être considéré comme un novice, et dans d'autres cas, comme un spécialiste.

Il nous semble nécessaire de tenir compte du concepteur lorsque l'on s'intéresse à la conception d'une commande « sûre » pour un système manufacturier. Pour cela, une autre approche de l'automatisation, tenant compte de la composante humaine, est utilisée pour pouvoir dégager des voies de progrès originales.

5 Problématiques et voies de progrès

Il y a différentes façons de voir l'automatisation et la commande (Riera, 2001). L'approche centrée technique considère qu'il faut automatiser à partir du moment où cela est possible. L'approche humaniste consiste à automatiser quand la tâche est pénible, ennuyeuse ou encore risquée pour un être humain. L'approche économiste consiste à automatiser lorsque cela revient moins cher. L'approche de l'automatique « humaine » est de considérer que l'automatisation ne consiste pas à retirer purement et simplement l'homme de la boucle de contrôle/commande (Bainbridge, 1983).

5.1 Automatisation et systèmes homme-machine

Selon le point de vue de l'automatique « humaine », automatiser consiste à concevoir un système qui accomplit partiellement ou totalement une fonction qui était ou pourrait être partiellement ou totalement réalisée par un opérateur humain (Parasuraman et al., 2000). Cette définition implique que l'automatisation peut varier entre différents niveaux allant du mode manuel au tout automatique et qu'un système automatisé est dans bien des cas amené à travailler avec un ou plusieurs opérateurs humains. La prise en compte de l'opérateur impose

une approche différente de l'automatisation orientée sur les Systèmes Homme-Machine (SHM).

Les travaux scientifiques sur les SHM s'intéressent à la modélisation des performances humaines, au contrôle de procédés industriels, à la place de l'opérateur humain dans le système, à la fiabilité humaine, et à la coopération Homme-Machine (Johannsen, 2007). Les originalités de l'approche SHM sont présentés ci-après (Riera et al., 2003).

- L'objectif principal est l'amélioration de la performance globale du SHM aussi bien au niveau du système qu'au niveau de l'opérateur,
- En conséquence, les effets induits de l'artéfact doivent être pris en compte. Pour cela, le SHM est étudié, en tenant compte aussi bien des caractéristiques humaines (nécessité de modèles décisionnels de l'homme) que des aspects techniques.
- Une phase d'évaluation est nécessaire, même si elle est difficile à réaliser parce que le comportement cognitif de l'opérateur n'est pas directement observable.

L'originalité des travaux en SHM se trouve dans une approche plus systémique qu'analytique de l'automatisation qui va être utilisée dans le cadre de notre problématique.

5.2 Axes de progrès retenus

Le SHM étudié est composé d'une PO, d'une PC, d'un concepteur (au niveau de compétence variable) et d'un expert (le formateur). L'objectif visé est de ne pas détériorer le système au moyen d'une mauvaise commande du concepteur.

Dans l'approche SHM, l'idée n'est pas de retirer systématiquement les opérateurs (i.e. le concepteur et l'expert), mais au contraire de leur fournir des outils nécessaires et performants pour la réalisation de leur tâche, respectivement, de conception et de vérification de la commande. Partant de ces constatations, nous proposons 2 voies de progrès : la sécurisation du système et son adaptation au concepteur.

5.2.1 Sécuriser le système

La sécurité est le premier point, qui doit être traité dans un système mis à la disposition de personnes pouvant commettre des erreurs. En effet, lors d'une mauvaise commande, il est nécessaire de détecter cette erreur pour éviter toutes détériorations.

La détection des erreurs de commande passe par la spécification des situations de détériorations, de dangers… Dans le cadre de cette thèse, nous considérons comme dangereuses pour les systèmes manufacturiers toutes les situations qui conduisent à une dégradation de l'installation. Nous avons uniquement pris en compte les problèmes de collisions entre des éléments de PO et/ou avec un produit.

De plus, il faut que l'approche proposée puisse s'appliquer à des exemples réels. Elle doit donc :

- être utilisable et compatible avec les contraintes temporelles liées à la PC,
- prendre en compte le fonctionnement de l'environnement matériel dans lequel évolue la commande du système. Dans le cadre de nos travaux, la commande est supposée être implémentée dans un API. Il faut donc tenir compte du retard de causalité dû au fonctionnement cyclique et synchrone de l'API,
- être indépendante de la commande implémentée. En effet, aucune hypothèse ne peut être faite sur la commande, car nous ne savons pas ce que peut proposer le concepteur de commande apprenant ou non.

La sécurisation de la commande peut s'envisager hors ligne ou en ligne. Dans le premier cas, la commande est vérifiée avant d'être implémentée. Dans le second cas, la commande est implémentée et la vérification se fait en temps réel.

L'autre voie de progrès concerne l'adaptation du système à l'apprenant.

5.2.2 Adapter le système au concepteur

La solution que nous proposons vise à donner au concepteur un champ d'actions sur le système qui soit adapté à son niveau, tout en pouvant conserver une vision globale de l'installation. Il nous semble en effet, que dans la formation, il est beaucoup plus valorisant pour l'élève automaticien de proposer une commande permettant de faire fonctionner une PO dans sa globalité plutôt que travailler sur un élément isolé du système. L'approche que nous proposons revient à percevoir le système à commander selon différents niveaux de l'axe « Moyens-But » du modèle de Lind (cf. paragraphe 4.3) tout en conservant une vision globale (Tout) de l'installation.

Pour cela, nous proposons que l'expert adapte le niveau de difficulté du cahier des charges au niveau de compétence du concepteur grâce à une modification du degré d'automatisation et du niveau d'autonomie du concepteur qui sera détaillée dans le chapitre 3.

Enfin, la prise en compte de la composante humaine dans le système, nous conduit à considérer la nécessité de fournir des explications au concepteur et/ou l'expert lorsqu'une erreur de commande est détectée.

5.3 Synthèse

Compte tenu des objectifs, il est nécessaire dans un premier temps d'assurer la sécurité du système vis-à-vis de la commande proposée par le concepteur. Dans le cas de l'applicatif en enseignement, la solution qui semble la plus simple, serait de comparer la commande de l'apprenant à celle de référence définie par l'enseignant. De cette manière l'enseignant est sûr de ne pas mettre en danger le système. Cependant l'inconvénient majeur qui nous fait rejeter cette solution, c'est que la commande est restreinte à une solution unique alors que plusieurs commandes peuvent répondre à un même cahier des charges. De plus, au niveau industriel cette approche est inapplicable.

La figure 1.9. reprend les problèmes auxquels nos travaux de thèse vont tenter d'apporter une réponse. Le concepteur propose une commande, celle-ci peut contenir des erreurs qu'il faut détecter pour éviter des détériorations au niveau du système (problématique 1 figure 1.9.).

Pour un système donné, défini par des entrées/sorties et un fonctionnement attendu, l'expert doit définir le degré d'automatisation et le champ d'actions du concepteur pour tenir compte des ses savoirs et savoir-faire et ainsi proposer un cahier des charges adapté (problématique 2 figure 1.9.). En cas d'erreur, il faut remonter l'information au concepteur et/ou l'expert (problématique 3 figure 1.9.).

Ces objectifs ainsi définis vont être utilisés comme critères dans le choix des approches utilisées par la suite pour vérifier le respect des spécifications par la commande proposée par le concepteur. Enfin, on retrouve les problématiques 1 (la sécurité) et 3 (les explications) aujourd'hui dans l'industrie manufacturière. Y apporter des éléments de réponses originaux peut conduire à l'amélioration de la productivité. En revanche la problématique 2 (la vision globale), qui va être maintenant développée, nous semble principalement intéressante dans le cas des applications pédagogiques.

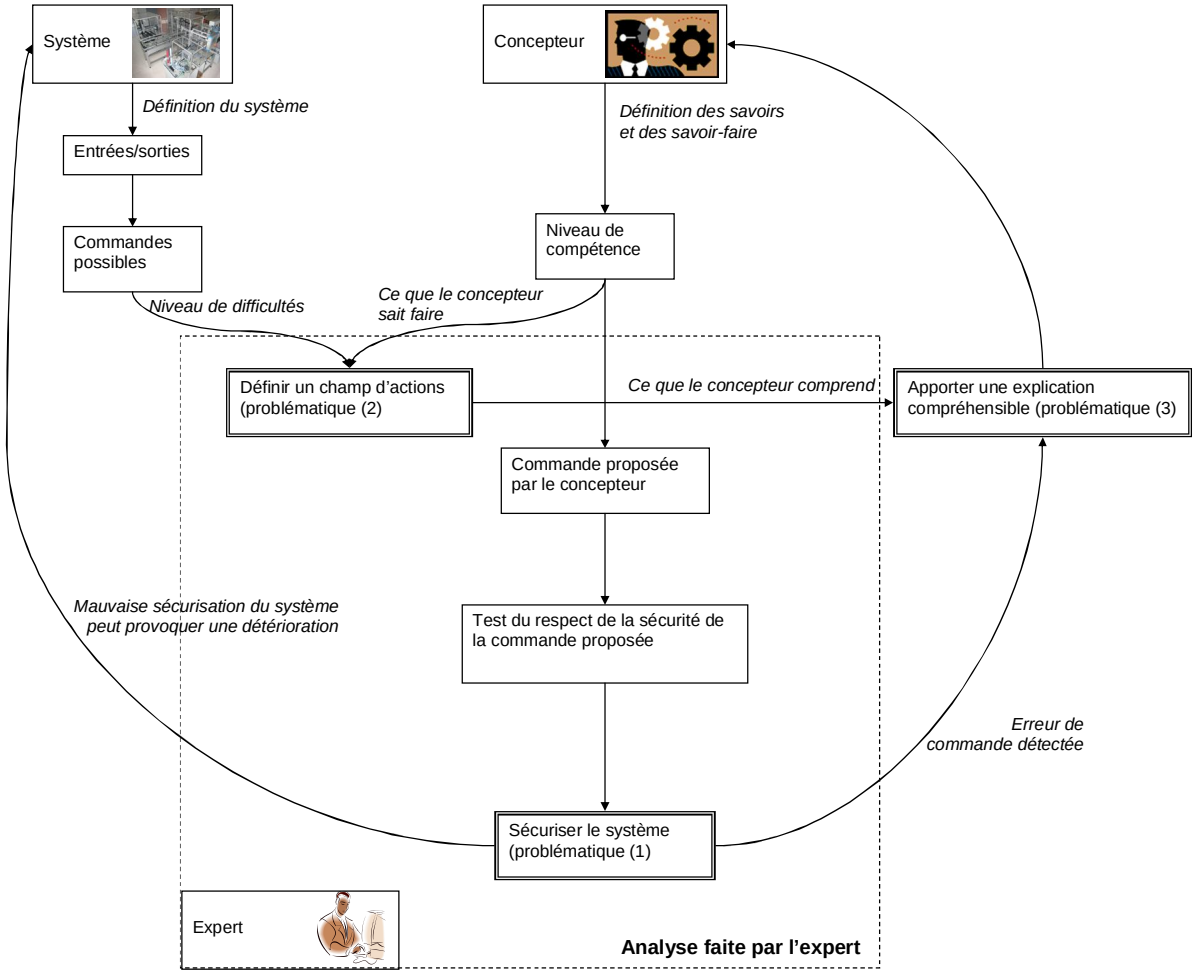


Figure 1.9. Synthèse de la problématique

L'idée est qu'à partir d'un système défini par des entrées/sorties, l'expert (i.e le formateur) définisse une vision du système et un cahier des charges adaptés aux connaissances du concepteur (figure 1.10.). Pour cela, l'expert doit étudier :

- Le système par une analyse fonctionnelle, dysfonctionnelle et ainsi en déduire les situations à risque,
- Le cahier des charges par l'analyse des attentes et ainsi connaître la structure de la solution,
- Le concepteur pour déterminer ses savoirs et savoir-faire. et ainsi définir un degré d'automatisation et adapter la perception du système.

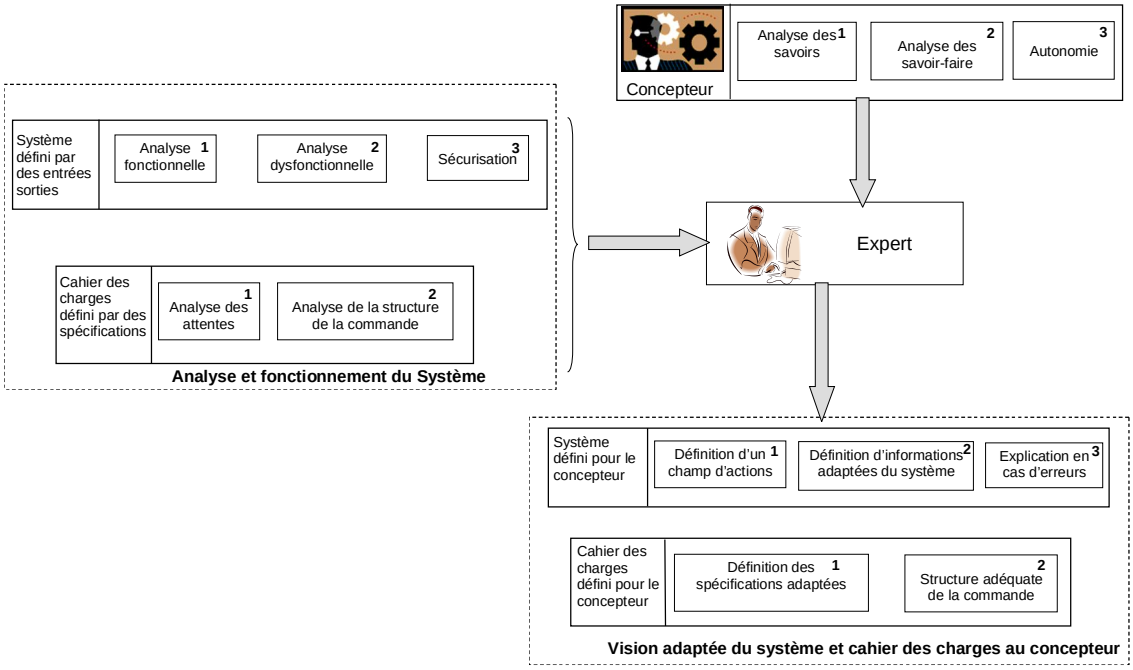


Figure 1.10. Démarche d'adaptation du système

De cette manière, l'expert peut définir un « nouveau » système adapté au concepteur, en gardant le même système mais avec des entrées/sorties du point de vue du concepteur (figure 1.11.).

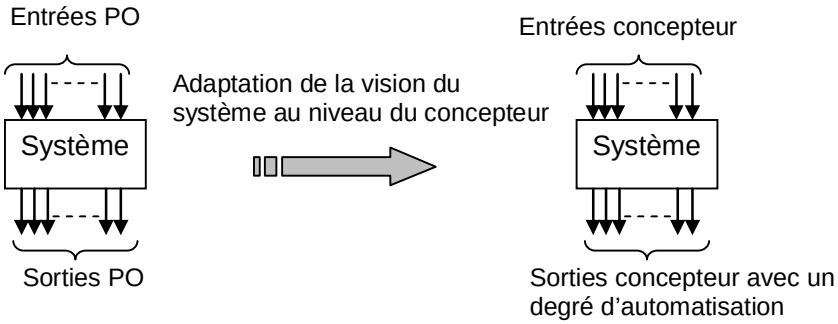


Figure 1.11. Démarche d'adaptation du système

Dans cette approche, qui sera détaillée dans le chapitre 3, les rôles de l'expert et l'apprenant sont conservés et valorisés car adaptés à leurs compétences. Nous proposons de finir ce premier chapitre en proposant un état de l'art bibliographique sur les différentes approches permettant d'obtenir une commande sûre. Ce travail orientera ainsi nos travaux sur la sécurisation de la PO.

6 Approches pour une commande sûre : synthèse bibliographique

Dans cette partie, des travaux permettant d'obtenir une commande sûre vont être présentés (Pétin, 2007). La première approche présentée concerne la sûreté de fonctionnement des systèmes réactifs en proposant une méthode de spécification allant de la vérification des spécifications du cahier des charges jusqu'à la validation des spécifications décrites par le Grafcet. La seconde approche montre l'élaboration de lois de commandes en utilisant les travaux de synthèse. La troisième approche présente les travaux portant sur la surveillance de systèmes.

6.1 Approche par validation/vérification

Avant d'introduire l'approche de validation/vérification, ces deux termes doivent être davantage définis. Nous sommes partis des définitions données dans l'ISO 8402 (ISO 8402, 1995). La validation est la confirmation que par examen et apport de preuves tangibles que les exigences particulières pour un usage spécifique sont satisfaites. Elle répond à la question « Construisons-nous le bon modèle ? ». La vérification est la confirmation par examen et apport de preuves tangibles que les exigences spécifiées ont été satisfaites. Elle répond à la question « Construisons-nous correctement le modèle ? ». En d'autres mots, la validation assure que les modèles de spécification, de conception, ou d'implémentation, sont conformes aux attentes du concepteur, en parallèle la vérification s'assure de la conformité des modèles élaborés au cours des processus de spécification et de conception par rapport aux exigences définies. La validation/vérification a pour but de définir un modèle de commande qui vérifie des propriétés de sécurité et de bon fonctionnement avant l'implémentation dans un système réel (figure 1.12.).

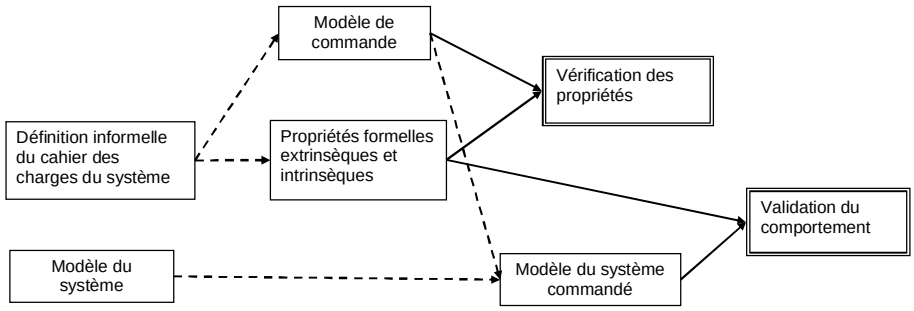


Figure 1.12. Validation/Vérification pour obtenir des lois de commande

Les travaux de validation et de vérification peuvent être regroupés en quatre classes (Frey, et al., 2000) : simulation, analyse d'atteignabilité, théorème Proving et model checking.

- Simulation : La validation/vérification se fait par exécution de scénarii de tests prédéfinis ou aléatoires. Bien que cette approche soit la plus utilisée dans le monde industriel, elle ne peut pas être exhaustive, demande beaucoup de temps en fonction du nombre d'entrées/sorties, et permet de détecter seulement les défaillances présentes dans les scénarii exécutés.
- Analyse d'atteignabilité : Par construction de tous les états atteignables d'un modèle de système, la validation/vérification se fait par analyse d'atteignabilité ou non d'états définis comme interdits (Kowalewski et al, 1996)
- Model-checking : L'approche est basée sur un modèle formel du système et des propriétés à vérifier. Le model-checking vérifie automatiquement les propriétés et détermine si elles sont vraies ou fausses, et dans le cas contraire, il fournit une trace conduisant au non respect (Schnoebelen et al., 1999 ; Frey et al., 2000 ; Machado et al., 2006 a, 2006 b ; Gourguff, 2007 ; Barragan, 2007)
- Théorème Proving : À partir des spécificités formelles, l'approche prouve mathématiquement les propriétés. L'approche nécessite une connaissance importante des systèmes formels (par un expert) et la démonstration globale est difficilement obtenue. Cependant cette approche a l'avantage d'éviter l'explosion combinatoire (Volker et al., 2002 ; Roussel et al., 2002 a, 2002 b ; Rushby, 2000)

Ces différentes approches se heurtent à la même difficulté : les risques d'explosion combinatoire et la complexité à définir correctement les propriétés du comportement du système.

6.2 Approche par synthèse

Une seconde méthode pour obtenir une commande sûre de fonctionnement est l'approche par synthèse s'appuyant sur les travaux de Ramadge et Wonham (Ramadge et al., 1989 ; Wonham et al., 1987). L'objectif est de définir l'ensemble des trajectoires qui correspondent aux spécifications souhaitées pour la commande. Le point de départ de cette approche consiste à spécifier l'ensemble des évolutions normales offertes par le procédé. Ces évolutions, encore appelées trajectoires, sont représentées souvent par des automates. La deuxième étape consiste à demander à l'utilisateur de spécifier le cahier des

charges : il s'agit de décrire l'ensemble des trajectoires souhaitées. Lors de la dernière étape, l'ensemble des séquences qui vérifient les propriétés établies dans le cahier des charges sont « extraites » du modèle du fonctionnement normal du procédé et des spécifications de commande (figure 1.13.).

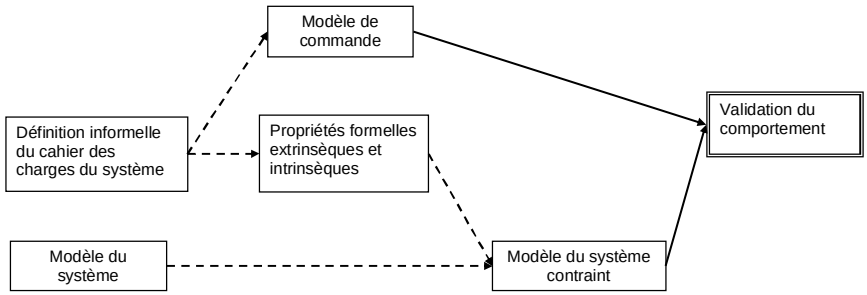


Figure 1.13. Synthèse formelle de la commande

Les travaux menés ont porté sur des extensions de cette théorie : stabilisation de Systèmes à Événements Discrets à structures vectorielles (Sarri, 1999), supervision par structure hiérarchique distribuée (Chafik, 2000 ; Chafik et Niel, 2000) et recouvrement de défaillances de SED temporels (Khatab, 2000). L'objectif de ces travaux est de proposer des solutions au problème de l'explosion combinatoire en réduisant la taille des modèles (Niel et al., 2001).

Des travaux menés (Kamach et al., 2002) portent sur l'étude multi-modèle du procédé. Cette démarche consiste à construire des modèles différents d'un même procédé lorsque celui-ci se trouve dans des configurations répondant à des objectifs différents. Pour un système de production, ces différents objectifs correspondent par exemple à des modes de marches différents. En ne décrivant que des comportements pertinents du point de vue de l'objectif visé, chaque modèle est de taille plus réduite qu'un modèle devant permettre l'étude de tous les objectifs. De plus, les contraintes ne sont plus regroupées sous la forme d'un superviseur centralisé, mais scindées en superviseurs propres à chaque modèle du procédé. Des mécanismes sont là aussi nécessaires pour sélectionner le superviseur actif et le changement cohérent de superviseur. De plus, les propriétés de blocage et de contrôlabilité nécessitent d'être redéfinies pour ces nouvelles structures. Les travaux développés visent donc à spécifier hors ligne toutes les lois de commande-supervision permettant de piloter un procédé de manière sûre (Nourelfath et Niel, 2000).

La commande est déduite a priori à partir du modèle du système et des modèles formels de spécification. Les approches par synthèse de la commande se heurtent elles aussi à des risques d'explosion combinatoire dans la définition de modèle de systèmes complexes et la mauvaise définition des spécifications conduit à la non-détection de certaines erreurs.

6.3 Approches par surveillance

Le but des travaux de surveillance est d'assurer la sécurité d'un système de commande et d'assurer la qualité d'un produit en exécution réelle, c'est-à-dire qu'en fonction de l'état du système, la surveillance doit réagir en cas de pannes, erreurs, et remonter les informations nécessaires à l'opérateur (figure 1.14.).

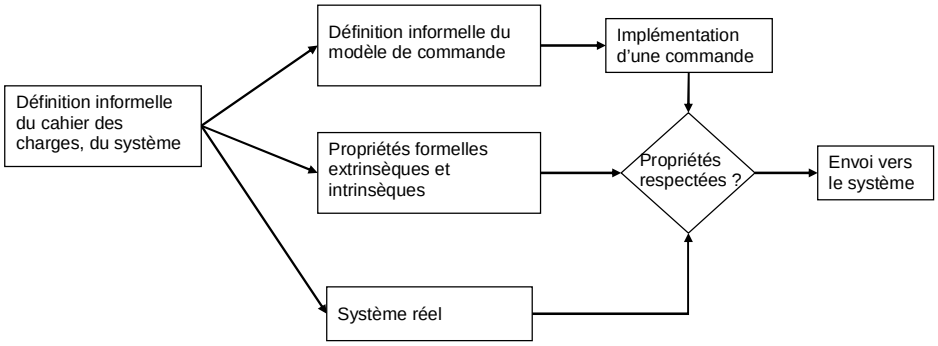


Figure 1.14. Surveillance de la commande

Pour cela, différentes approches sont mises en place : par référence, par émulation, ou par filtre. Une étude détaillée a été réalisée par Zamaï (Zamaï, 1997) :

- Surveillance par référence : Un modèle de procédé « modèle de référence » définit le comportement normal du procédé (Combacau 1991 ; Chaillet-Subias, 1995). Avant chaque émission d'une requête de la PC vers la PO, celle-ci est testée sur le modèle de référence. Si la requête n'est pas compatible une erreur de commande est détectée. Il y a possibilité de détecter des erreurs au niveau de la PO, en fonction des comptes rendus émis et l'état du modèle de référence.

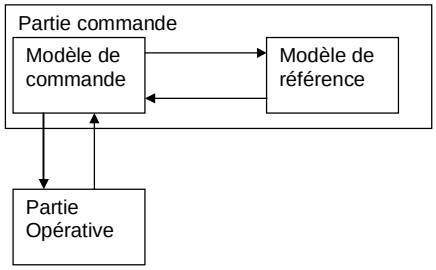


Figure 1.15. Approche par modèle de référence

- Surveillance par émulation : Par comparaison entre les informations provenant de la PO et l'émulateur (modèle des évolutions normales de la PO), toute incohérence révélera une défaillance au niveau du procédé (Holloway 1990, Toguyeni, 1992). Dans le cadre de nos travaux, cette approche n'est pas utilisable car elle ne permet pas de détecter les erreurs au niveau de la commande.

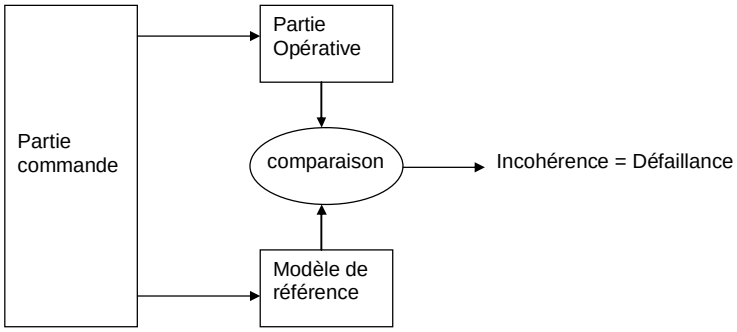
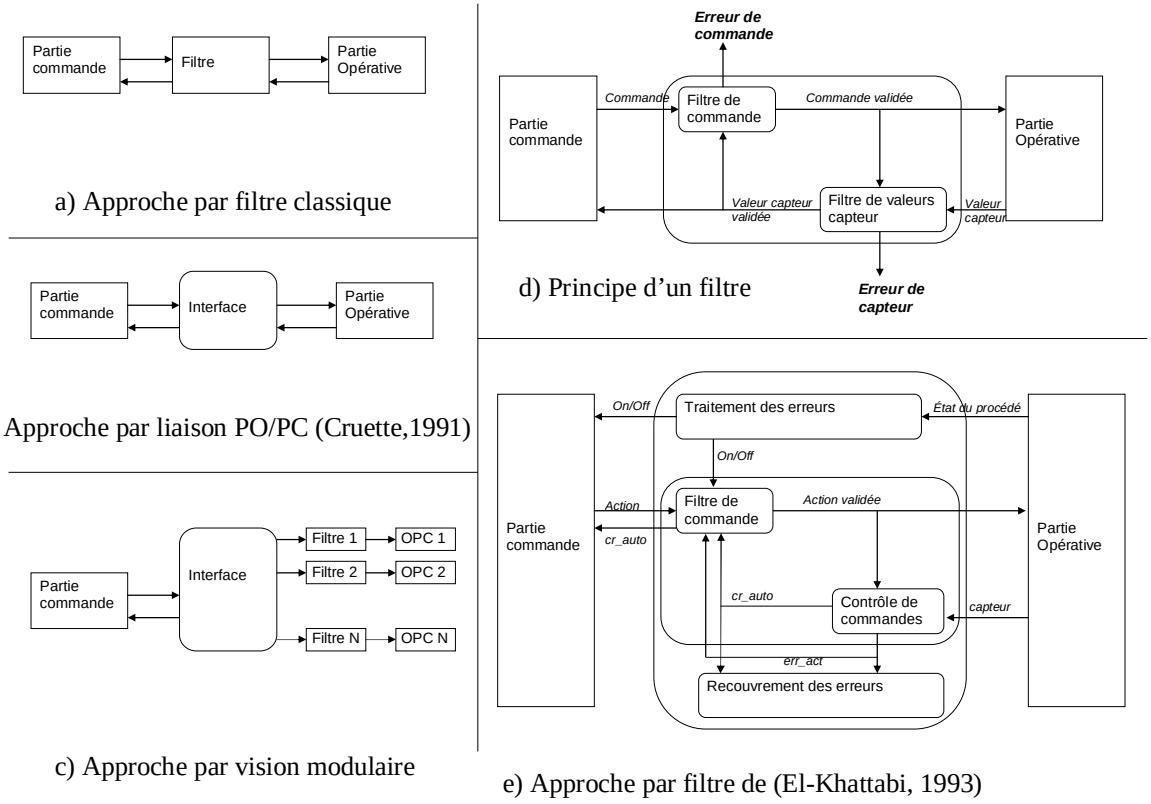


Figure 1.16. Approche par modèle en émulation

- Surveillance par filtre : Cette approche place un bloc de validation entre la PO et la PC. Ce bloc permet d'une part d'assurer la cohérence des commandes émises par rapport aux commandes prévues (défaillance de commande) et d'autre part de valider ou non la réalisation d'une commande (défaillance du procédé). Plusieurs travaux ont porté sur cette approche (Lhoste, 1994 ; Alanche et al., 1986 ; Cruette 1991 ; El-Khattabi, 1993 ; Toguyeni, 1992).



a) Approche par filtre classique

b) Approche par liaison PO/PC (Cruette,1991)

c) Approche par vision modulaire

d) Principe d'un filtre

e) Approche par filtre de (El-Khattabi, 1993)

Figure 1.17. Approches par modèle en filtre

6.4 Synthèse

Nous retenons de cette étude, l'approche de synthèse et l'approche par filtre. L'approche de synthèse a été retenue par rapport aux précédents travaux faits au sein de l'équipe SED et Supervision du CReSTIC qui peuvent être adaptée pour assurer que la commande soit sûre avant son implémentation (figure 1.18.). Dans cette approche, l'expert définit le modèle du système ainsi que les spécifications pour définir le système contraint par l'étape de synthèse. Ensuite, les évolutions de la commande proposée par le concepteur, doivent être cohérentes avec celles du modèle contraint. Si c'est le cas, le concepteur implémente sa commande en étant sûr de ne pas atteindre des situations dangereuses. Il s'agit donc d'une approche de vérification hors ligne.

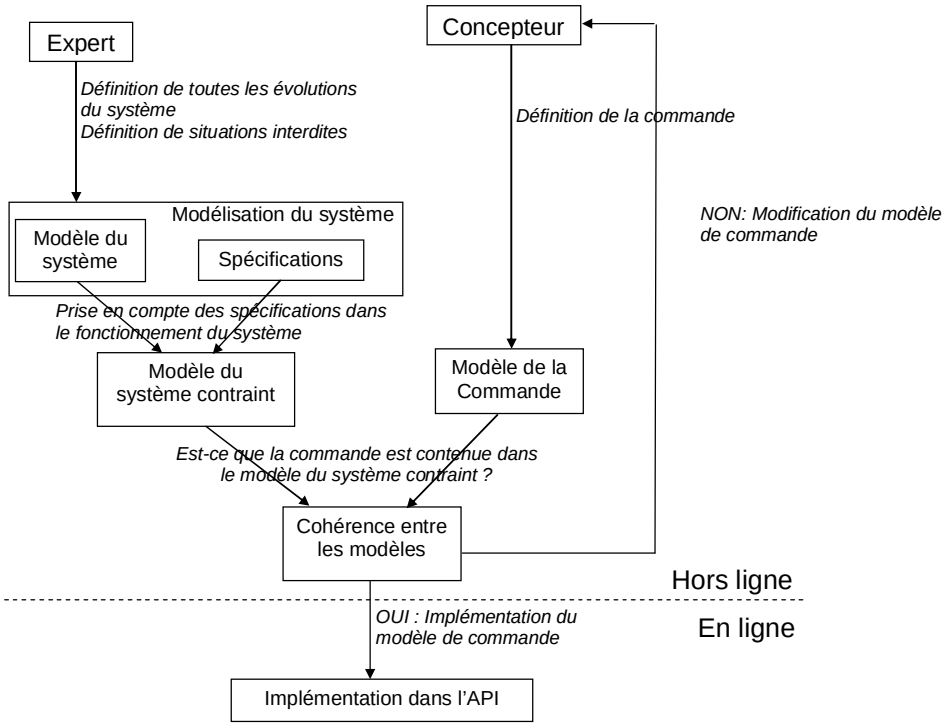


Figure 1.18. Sécurisation de la commande avant l'implémentation

L'approche par filtre permet a priori une implémentation aisée dans un API. Le principe (figure 1.19.) consiste à implémenter dans l'API, les spécifications définies par l'expert. De son côté, le concepteur, à partir du cahier des charges, réalise et implémente sa commande dans l'API. En ligne, les ordres sont envoyés vers la PO si les spécifications sont respectées. Habituellement, il y a une similitude entre les spécifications (expert) et le cahier des charges (concepteur). Nous proposerons une approche différente.

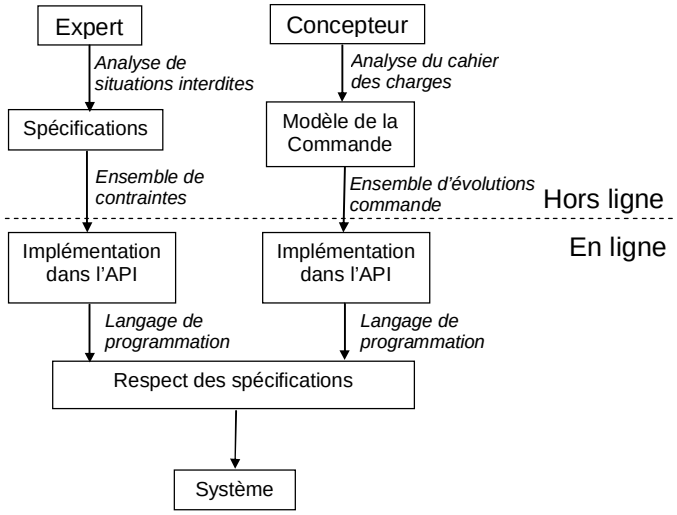


Figure 1.19. Sécurisation de la commande pendant l'exécution

La fiabilité des deux approches de validation de commande suppose que les spécifications sont correctement définies. L'un des objectifs est d'être utilisable sur un système réel, il est donc nécessaire de lever cette supposition pour être certain de ne pas détériorer le système. Il faut donc proposer une approche permettant de garantir que les spécifications sont suffisantes pour assurer la sécurité du système, sans faire d'hypothèse sur la commande.

7 Conclusion

Le problème initial concernait l'utilisation par des apprenants de systèmes industriels réels dans le cadre de leur formation d'automaticien. Nous avons montré que cette problématique est présente dans le monde industriel. On considère toutefois dans ce cas, le concepteur comme un automaticien spécialiste, mais qui reste susceptible de commettre des erreurs de commande.

Ce chapitre a permis d'introduire la problématique de la conception de la commande dans les systèmes manufacturiers. Si celle-ci n'est pas sûre, elle peut entraîner des dégradations de la PO. Nous avons montré que les outils évolués de conception d'automatismes sont à ce jour trop peu utilisés et les modifications du programme de commande trop nombreuses pendant la phase d'exploitation, pour pouvoir garantir la sécurité des installations manufacturières.

Des voies de progrès ont été proposées en prenant en considération la composante humaine du système étudié. Cela nous a conduits à définir 2 axes de recherche. Le premier concerne la sécurisation de l'installation et le second l'adaptation du système au concepteur. Nous avons proposé d'étudier 2 approches pour l'obtention d'une commande sûre. La première hors ligne est basée sur une approche par synthèse qui fait l'objet du chapitre suivant. La seconde en ligne repose sur une approche par filtre qui fera l'objet des chapitres 3 et 4.

Chapitre 2 :
Utilisation d'une approche de synthèse pour la validation hors ligne de la commande

1 Introduction

Dans ce chapitre, les travaux effectués sur la synthèse de la commande selon Ramadge et Wonham (Wonhan et al., 1987, Ramadge et al., 1989) sont présentés en vue de proposer une approche hors ligne de validation de la commande. Les recherches du groupe SED et supervision du CReSTIC (Marangé, et al., 2008 c) présentées dans ce chapitre font suite à de précédents travaux de l'équipe (Ndjab, 1999 ; Tajer, 2005) sur une proposition de synthèse de la commande spécifiée par Grafcet. Une méthodologie structurée pour modéliser la partie opérative (PO), ainsi qu'une approche de synthèse de la commande pour définir un superviseur et le comparer à une commande réalisée sous forme de Grafcet avaient alors été proposées. Cette approche permet lors d'une situation de blocage, d'éliminer l'état bloquant ainsi que les transitions associées, sans s'assurer au préalable que la commande implémentée corrigée correspond aux attentes du concepteur. Dans ce chapitre, cette approche est détournée pour effectuer la validation de commande. En cas de situation de blocage, celle-ci sera analysée pour apporter une explication au concepteur qui devra alors changer sa commande jusqu'à ce qu'il n'y ait plus de blocage. Cela assurera que la commande proposée ne conduise pas le système dans un état de détérioration ou de danger.

Comme cela a été montré dans le premier chapitre, la complexité des Systèmes Automatisés de Production (SAP), accroît les exigences des utilisateurs en termes de sûreté de fonctionnement et d'aide à la conception de programmes. Pour aider le concepteur sur ces aspects, deux approches sont possibles (Faure et al., 2001) : la première par validation et la seconde par synthèse. La première approche consiste à laisser le concepteur du système de commande développer des lois de commande basées sur le cahier des charges et à analyser ensuite automatiquement une représentation formelle de ces lois de commande. Une telle analyse se fonde par exemple sur l'une des techniques d'analyse développées pour les automates à états (model-cheking (Bérard et al., 1999, Machado, 2006 a, 2006 b)) ou des techniques de calcul symbolique (theorem-proving (Roussel et al., 2002 a, 2002 b)). Il existe également des méthodes de validation à base de langages synchrones qui apportent de bons résultats sans les inconvénients des méthodes à base d'automates à états (Gaffé, 2003 ; Delaval et al., 2007). La deuxième approche dite de synthèse (Ramadge et al., 1989 ; Makungun et al., 1999 ; Hellgren et al., 2002 ; Ghaffari et al., 2002 ; Gouyon et al., 2004 ; Achour et al., 2004), permet de déduire directement des lois de commande des spécifications et du procédé sans réelle participation du concepteur au processus d'élaboration de la commande. Cette approche nécessite une modélisation formelle du procédé et des spécifications pour manipuler les modèles formels représentatifs.

Dans des travaux précédents, des chercheurs du CReSTIC (Carré-Ménétrier et al., 2002; Zaytoon, 2001 ; Zaytoon et al., 1999) se sont intéressés au développement de techniques permettant la synthèse automatique d'une implantation correcte de la commande spécifiée au moyen d'un modèle Grafcet (IEC 1988 ; IEC 2002 b). Ce modèle a été choisi en raison de sa convivialité et de sa large utilisation dans la spécification de la commande des automatismes séquentiels. Il a été choisi d'asseoir cette démarche sur des outils/méthodes ayant des bases formelles, tels que les automates et la théorie de supervision des systèmes à événements discrets (SED) (Ramadge et al., 1989). Le travail a ainsi abouti au développement d'une approche globale de synthèse et d'implantation de la commande réactive la plus permissive vis-à-vis d'un Grafcet, d'un modèle du procédé à commander et des contraintes à respecter sur les évolutions du procédé. Cependant, cette approche possède deux inconvénients majeurs liés à la difficulté de modéliser le procédé et les spécifications ainsi qu'à l'explosion combinatoire inhérente à tous SED. Elle est de plus très sensible aux erreurs de modélisation. Cette démarche conduit à une construction automatique sans retour d'information vers le concepteur et, si elle permet la synthèse d'une commande sûre, déterministe, réactive et sans

blocage, elle ne permet pas de s'assurer que le résultat final soit en adéquation avec les souhaits initiaux du concepteur.

Suite à cette dernière remarque, les travaux présentés dans ce chapitre proposent d'étudier le comportement global du système pour faire de la validation de commande. L'idée est de conserver le principe de la démarche de synthèse et ses outils/méthodes en l'adaptant pour effectuer de la validation (figure 2.1.). L'expert a à sa charge la définition du modèle de PO et des spécifications à respecter, et le concepteur doit définir le modèle de commande. L'approche hors ligne de validation permet d'une part, à l'expert de valider le modèle de PO et les spécifications en supposant une commande juste, et d'autre part au concepteur de valider la commande en supposant les modèles de PO et les spécifications justes. Avant l'implémentation dans l'Automate Programmable Industriel (API), à partir des informations obtenues à la fin de la comparaison entre le modèle provenant de la synthèse et celui de la commande, il est possible d'informer l'expert ou le concepteur sur d'éventuelles erreurs, provenant de blocages, ou de contraintes non respectées. Lorsqu'il n'y aura plus de corrections à apporter, le concepteur peut implémenter la commande dans l'API.

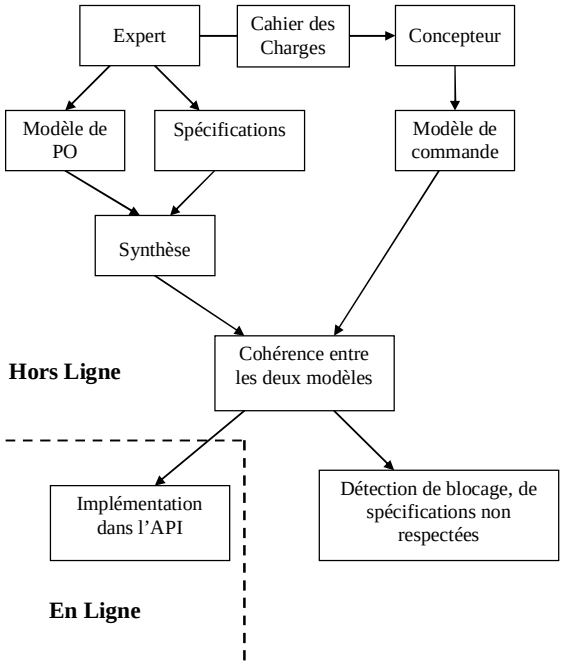


Figure 2.1. Démarche de validation par synthèse

Avant de présenter la démarche de validation hors ligne, il faut rappeler les précédents travaux permettant de rendre l'approche applicable à des systèmes réels en proposant :

- Pour la PO, une méthodologie de modélisation structurante à base d'automates booléens à travers une adaptation de l'approche de (Chandra et al., 2001).
- Pour les contraintes, un principe de modélisation par des équations logiques entre les entrées/sorties du contrôleur.

Ces deux points seront brièvement rappelés dans la première partie. L'approche de synthèse proposée par Tajer (Tajer, 2005) sera rappelée dans la troisième partie, ainsi que l'utilisation de celle-ci pour faire de la validation de commande. La quatrième partie présentera la méthode permettant de valider la commande, et de détecter des erreurs au niveau de la PO, ou au niveau de la définition des spécifications.

2 Principe de la modélisation de la partie opérative et des contraintes

La construction d'une commande correcte par synthèse automatique suppose l'élaboration de modèles pour la PO et les contraintes. Dans ces travaux, les spécifications liées à la commande sont scindées en deux catégories : celles concernant la commande proprement dite et celles représentant les contraintes de sécurité et de vivacité. Les contraintes de sécurité correspondent à ce qu'il ne faut pas faire et les contraintes de vivacité correspondent au séquencement que doit respecter le système complet. Différents travaux (Philippot et al., 2003) ont proposé une modélisation à partir d'automates rudimentaires et événementiels, cependant l'élaboration d'un modèle avec ces automates reste très intuitive, génératrice d'erreurs de modélisation et d'explosion combinatoire. Pour résoudre une partie de cette problématique, une construction modulaire et structurée des modèles est proposée : les modèles de commande, de PO et des contraintes étant explicites et indépendants entre eux. Cette section se compose de deux parties : la première est consacrée à la PO et la seconde aux contraintes. La commande est modélisée quant à elle en Grafcet, nous supposons qu'elle peut contenir des erreurs au niveau de l'interprétation du cahier des charges mais pas au niveau de la structure ou que ces erreurs sont détectées par le logiciel de saisie.

2.1 Modélisation de la partie opérative

La modélisation explicite de la PO est une tâche complexe qui se heurte non seulement à des problèmes méthodologiques de structuration et de composition de ses éléments mais aussi au choix de la granularité du comportement modélisé. Par conséquent, la méthode est basée sur des règles en s'appuyant sur des travaux de V. Chandra (Chandra et al., 2001, Chandra et al., 2002), pour aider l'expert dans sa tâche de conception des modèles de PO.

2.1.1 Règles d'occurrence et relations de précédence

Pour conserver le bénéfice du cadre formel proposé par la théorie de la supervision (Ramadge et al., 1989), la modélisation de la PO s'effectue sous forme d'automates décrivant les évolutions physiquement possibles en fonctionnement normal du procédé par des événements simples. Pour refléter les interactions entre la commande spécifiée par Grafcet et la PO, l'interprétation de S. Balemi (Balemi et al., 1993) est retenue, où les événements commandables Σ_c représentent les entrées du procédé et les événements non commandables Σ_u, ses sorties. Cette interprétation est spécialisée de manière à concilier la nature continue des actions et des variables d'entrées du Grafcet avec le caractère événementiel du modèle de la théorie. Ainsi, la correspondance suivante est retenue : un événement commandable correspond soit à l'activation $\uparrow z$ soit à la désactivation $\downarrow z$ d'un ordre du Grafcet, tandis qu'un événement non commandable est associé soit au front montant $\uparrow e$ soit au front descendant $\downarrow e$ d'une variable d'entrée du Grafcet. Les ensembles Σ_c et Σ_u s'écrivent alors $\Sigma_c = +Z \cup -Z$ et $\Sigma_u = +E \cup -E$. Les ensembles d'événements +Z et -Z, correspondent aux activations et désactivations des ordres, tandis que les ensembles d'événements +E et -E reflètent les ensembles des fronts montants et descendants des entrées. La modélisation retenue se place d'un point de vue commande, c'est-à-dire que les entrées représentent les capteurs, et les sorties, les actionneurs.

Pour structurer la modélisation de la PO, une modélisation à base de règles définissant les interactions entre les événements commandables et les événements non commandables, et des relations entre ces événements non commandables a été retenue (Philippot et al., 2004 a). Il s'agit dans le premier cas de règles dites d'occurrence correspondant donc à l'influence des événements commandables sur les événements non commandables, et dans le second cas de relations de précédence représentant la chronologie entre les événements non commandables conséquents à un même événement commandable. Ces règles définies par l'utilisateur sont

ensuite traduites en automates à états. La définition des règles est obtenue de la manière suivante :

- La définition des conditions initiales des entrées et des sorties dans la phase de modélisation de la PO. Ces états initiaux correspondent soit aux états actifs, soit aux états inactifs des entrées/sorties.
- L'interaction entre les sorties et les entrées permet de déterminer des règles d'occurrence de type cause/conséquence. Il s'agit en fait de déterminer s'il existe des événements non commandables α du système qui sont générés suite à l'occurrence d'un événement commandable σ, en tenant compte de l'état du système β (état des entrées et des sorties). Formellement, l'écriture de cette proposition est la suivante : $\forall\sigma \in \beta\sum_c$ avec $\sum_c = +Z \cup -Z$, $\beta \in E \cup Z$ et si $\exists\alpha\in\sum_u$ avec $\sum_u = +E \cup -E$ tel que α est une conséquence de σ alors on définit la règle d'occurrence :

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

- Un événement commandable peut conduire à générer un ou plusieurs événements non commandables. Pour séparer ces événements non commandables en terme de précédence, pour chaque règle d'occurrence ayant la même cause, une relation appelée « relation de précédence » est définie, permettant de préciser la chronologie liant les événements non commandables successifs, tel que :

$\forall\sigma\in\sum_c$ si $\exists$ $(\alpha_1, \alpha_2) \in\sum_u \times \sum_u$ et $\sigma \Rightarrow \alpha_1$, $\sigma \Rightarrow \alpha_2$ alors on définit une relation de précédence entre α_1 et α_2 par :

$$\alpha_1 \rightarrow \alpha_2 \quad ou \quad \alpha_2 \rightarrow \alpha_1$$

A partir des règles d'occurrence et des relations de précédence, nous proposons de générer automatiquement un modèle automate à états à travers une méthode de construction dite booléenne.

2.1.2 Construction de l'automate de partie opérative

Les premiers travaux de Ramadge et Wonham (Wonham et al., 1987) sur la théorie de supervision préconisaient l'emploi d'automates événementiels et rudimentaires mais peu explicites car les états portent peu d'informations sur le système. De nombreux travaux ont

proposé d'autres modèles pour représenter le système tels que les State-Based Discrete-Event introduit par (Wong, 1990), les Vector Discrete-Event System de (Li, 1991), les Boolean Discrete Event System qui sont un cas particulier des Vector Discrete-Event System avec une structure d'état à vecteur booléen (Wang, 2000).

Pour prendre en compte les informations sur l'état de la PO et sur l'état de la commande, nous avons donc choisi la dernière structure en utilisant au niveau de chaque état de l'automate, une structure à base de vecteurs d'entrées et de sorties. Par conséquent, la définition de l'automate à états booléens retenue pour ces travaux est la suivante :

Un automate à états booléens est défini par un 4-uplet $G = (P, \sum, \delta, p_0)$, où P est l'ensemble des états de G, $\sum = \sum_u \cup \sum_c$, δ la fonction de transition définie par $\delta : P \times \sum \rightarrow P$, l'état initial p_0 de l'automate G. Les états de P sont définis par un vecteur d'états booléens à n dimensions. Chaque composante du vecteur d'état peut prendre la valeur 0 ou 1. Le nombre d'états maximum du modèle automate est donné par le nombre de variables d'états, c'est-à-dire 2^n états au maximum pour n variables. A chaque état p de l'automate G est associé le vecteur d'entrées $E = [e_1,..., e_m]$, le vecteur de sorties $Z = [z_1,..., z_s]$ où m est le nombre d'entrées de la commande et s le nombre de sorties de la commande. Dans l'algèbre de Boole, l'ensemble IB est défini par $(\{0,1\}, \wedge, \vee, \neg)$. L'ensemble P des états de G est donc défini par :

$$P = \{[[e_1,\ldots, e_m] , [z_1,\ldots, z_s]] / e_i , z_j \in \{0,1\}*\{0,1\}, i=1,\ldots,m, j=1,...,s \text{ et } m+s=n\} \subseteq IB^n.$$

La construction de l'automate de la PO est basée sur les règles et les relations et s'effectue selon les 4 étapes suivantes (un exemple est traité dans l'annexe 2) :

1. Construire la table avec les 2^n états (n est le nombre d'entrées/sorties du système) décrivant toutes les combinaisons possibles entre les entrées/sorties du système. Cet ensemble correspond à l'ensemble des états P.

2. Supprimer les incohérences logiques entre les entrées. Ici, nous ne nous intéressons pas aux incohérences logiques liées aux événements commandables mais à celles liées aux conséquences (événements non commandables) car seules les entrées de la PO sont considérées. Après la suppression de ces incohérences logiques correspondant aux états p_{inc}, l'ensemble des états P est donc défini par : $P = P - \{p_{inc}\}$

3. Construire l'automate équivalent à partir du diagramme des évolutions commandables. Ce diagramme est déterminé à partir de la table de vérité

représentant les états restants après la suppression des incohérences logiques. Pour chaque combinaison, l'occurrence d'un événement commandable $\uparrow z$ ou $\downarrow z$ de $\sum_c$ permet de changer la variable d'état de sortie ce qui conduit à un nouvel état. Le principe consiste à construire à partir d'un état p_k de P ($p_k=(E_k, Z_k)$ avec $Z_k = [z_{1k},\ldots, z_{sk}]$ et $E_k= [e_{1k},\ldots, e_{mk}]$) l'état atteignable p_{k+1} suite à l'occurrence d'un événement commandable $\uparrow z_{jk}$ ou $\downarrow z_{jk}$ avec j=1,…,s et k=1,…,2n.

4. Compléter l'automate avec les évolutions non commandables en utilisant les règles d'occurrence, les relations de précédence et les conditions initiales. Suite à l'occurrence d'un événement commandable, les règles d'occurrence permettent de déterminer l'état atteignable dans P. Les relations de précédence permettent de définir la séquence d'états atteignables par des événements non commandables, suite à l'occurrence d'un même événement commandable.

Cette méthode de modélisation est appliquée à chacun des éléments du système. Le modèle complet de la PO est obtenu alors par la composition asynchrone de tous les modèles élémentaires. Nous obtenons un nombre fini d'états dans le modèle de PO qui est égal au maximum à 2^{nc+na} où nc et na représentent respectivement le nombre de capteurs et d'actionneurs.

2.2 Modélisation des contraintes

Dans cette approche de synthèse, les contraintes à modéliser sont des contraintes de sécurité et/ou des contraintes de vivacité. Intégrer des contraintes consiste à inhiber des actions et/ou à agencer et séquencer l'exécution des commandes envoyées au procédé. Dans les SED, les contraintes s'expriment classiquement par des automates à états qui ne sont pas sans poser des problèmes de conception et d'interprétation. Pour faciliter l'écriture des contraintes sans utiliser des automates à états, une alternative est proposée et basée sur l'utilisation d'équations logiques qui sont fonction de l'ensemble des entrées et de l'ensemble des sorties du Grafcet. L'utilisation des équations logiques permet de réduire l'explosion combinatoire lors de la génération de la commande. Cependant, l'utilisation d'équations logiques ne nous permet pas de représenter une séquence d'événements bien précise, sans passer par des reconstructeurs d'informations.

Il existe quelques travaux utilisant les équations logiques dans l'algèbre de Boole dont ceux de Roussel (Roussel et al., 2003) qui utilisent une approche algébrique pour développer

une méthode de synthèse formelle. Cette approche permet d'obtenir, pour un système logique, une commande conforme aux spécifications. Elle repose sur la formalisation des spécifications sous forme d'assertions dans une algèbre de Boole adaptée aux signaux binaires appelé Algèbre II (Roussel, 1994) (Merle et al., 2007). L'algèbre II permet de décrire des comportements intégrant des aspects temporels, en manipulant cette fois des signaux binaires en fonction du temps et des variables booléennes. Les travaux présentés sur la modélisation des contraintes s'effectuent dans l'algèbre de Boole classique sans aspect temporel.

Le cadre formel de la modélisation des contraintes est basé sur l'algèbre de Boole des propositions pour laquelle les éléments manipulés sont des valeurs binaires 0 ou 1 qui représentent les valeurs des entrées E et des sorties Z du Grafcet de spécification. Ces éléments sont représentés par des fonctions définies dans IB. La définition de ce cadre formel est développée dans (Tajer, 2005).

L'emploi des équations logiques booléennes a pour avantage d'exprimer les contraintes de manière explicite et flexible, dans un langage bien connu de l'expert. L'écriture de ces contraintes se détermine simplement en utilisant le connecteur d'implication « Si …Alors… » entre les événements. Cette implication va permettre d'exprimer les contraintes de sécurité et de vivacité uniquement sous forme statique dépendant seulement du vecteur d'entrées sorties. Cela ne permet donc pas d'exprimer des contraintes temporelles ou des séquences d'événements. Par exemple, dans notre démarche, la dynamique des vérins telle que la vitesse d'avancement n'est pas considérée.

La partie suivante présente comment cette approche de synthèse proposée dans les précédents travaux de l'équipe, est détournée pour faire de la validation de commande.

3 De la synthèse automatique de la commande vers une approche de validation de la commande

3.1 Rappel de la démarche de synthèse automatique

La démarche de synthèse automatique hors ligne développée dans les précédents travaux (Carre-Ménétrier, 2000 ; Tajer, 2005 ; Tajer et al., 2006 ; Marangé, et al. 2006 b), a pour objectif la construction d'une commande sûre, sans blocage, déterministe et réactive. Elle comprend six étapes présentées figure 2.2.

1. L'étape de modélisation (étape 1) est décomposée en trois phases : (i) l'élaboration du Grafcet correspondant à la spécification de la commande, (ii) la modélisation de la PO et (iii) la spécification des contraintes de sécurité et de vivacité.

2. L'opération de synthèse (étape 2) consiste à partir des contraintes et du procédé, modélisés précédemment, à générer l'automate du superviseur *SUP* selon l'algorithme de synthèse proposé par (Kumar, 1991). Cet automate correspond au comportement commandable maximal admissible du procédé par rapport aux contraintes.

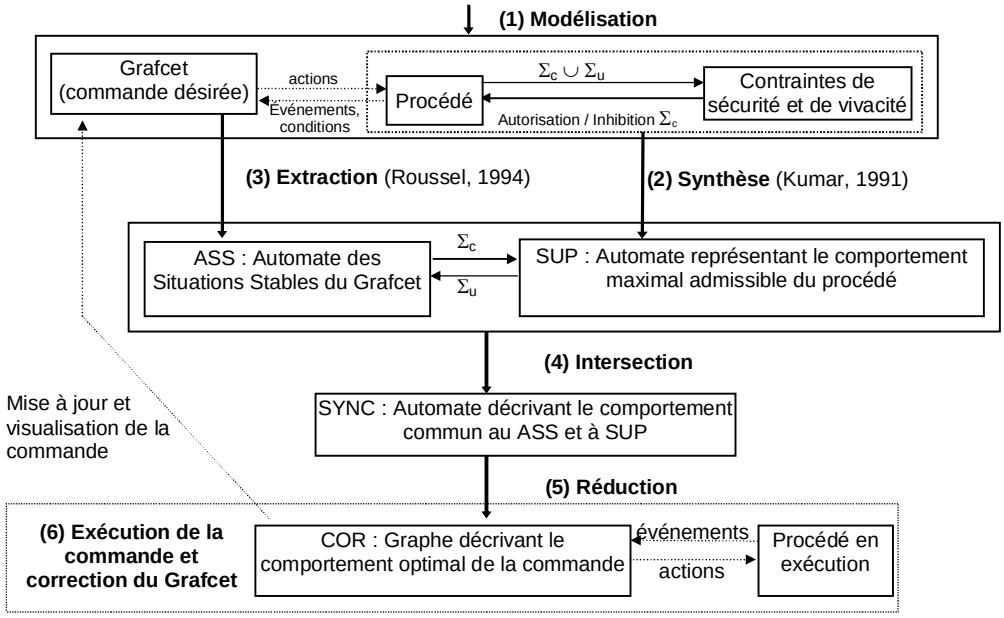


Figure 2.2. Les étapes de l'approche de synthèse hors ligne

3. L'extraction (étape 3) consiste à appliquer l'algorithme de (Roussel, 1994) afin de traduire le modèle Grafcet en automate équivalent noté *ASS* (Automate des Situations Stables). L'extraction est définie jusqu'à présent en respectant la norme Grafcet (IEC, 1988). Les concepts introduits par la révision technique générale de la norme (IEC, 2002 b) (événement d'entrée, événement interne, assignation, affectation, forçage, macro-étape et encapsulation) ne sont pas encore pris en compte et leur modélisation reste à étudier.

4. L'intersection (étape 4) a pour objectif de générer l'automate nommé *SYNC* qui décrit le comportement commun aux automates *SUP* et *ASS* de manière à ne retenir, dans le modèle de commande, que les évolutions autorisées par le

superviseur. L'algorithme d'intersection parcourt les automates *SUP* et *ASS* et construit *SYNC* à partir des événements admissibles par les deux automates. Le principe d'intersection est particulier car chacun des automates *ASS* et *SUP* repose sur une sémantique propre (Carré-Ménétrier et al., 2002). Pour *ASS*, un état est muni de l'ensemble des actions actives lors de la situation correspondante du Grafcet, et les transitions sont décrites par une fonction composée des entrées du Grafcet et de fronts sur ces entrées. Quant à *SUP*, il correspond à un automate rudimentaire, où les transitions sont uniquement décrites par des événements.

L'automate *SYNC* se construit en tant que modèle asynchrone dont les transitions correspondent à des événements simples. Les ordres qui doivent être émis en parallèle dans une situation du Grafcet, sont représentés dans *SYNC* par une structure arborescente composée de transitions décrivant les activations et les désactivations successives autorisées par le superviseur. Les états reliés par ces transitions constituent une région correspondante à la dite situation du Grafcet. Ainsi, l'automate *SYNC* est constitué d'un ensemble de régions (figure 2.3). Les transitions intra-région correspondent aux événements commandables, et les transitions inter-régions aux événements non commandables.

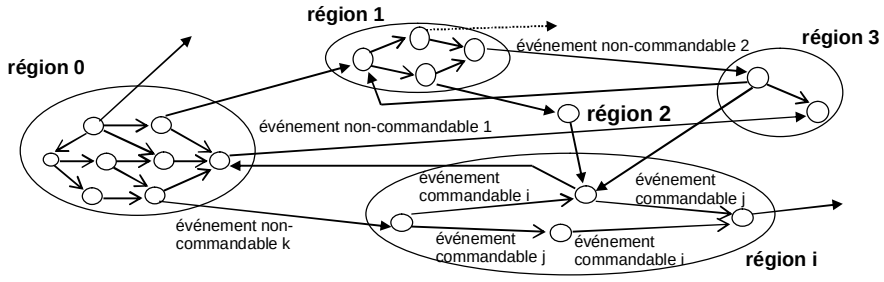


Figure 2.3. Modèle sémantique de l'automate SYNC

5. La réduction (étape 5) s'effectue en trois phases : dans la première phase, les états bloquants de SYNC sont rendus non atteignables par suppression de leurs transitions amont, la deuxième phase dite d'optimisation consiste à retirer de l'automate *SYNC* modifié lors de la première phase, les transitions qui sont non atteignables durant l'exécution réelle et enfin, la troisième phase procède à l'agrégation des situations instables, de manière à aboutir à une commande réactive qui évolue, suite à l'occurrence d'un événement non commandable, d'une situation stable à une autre. L'automate résultant de l'étape de réduction

(*COR*) est ensuite implanté dans l'API et correspond à la restriction minimale du comportement du Grafcet qui garantit la conformité par rapport aux contraintes spécifiées et le non-blocage du système.

6. La dernière phase (étape 6) concerne l'exécution de la commande optimale implantée (*COR*).

La commande ainsi obtenue est sûre, sans blocage, déterministe et réactive, mais par la suppression des états bloquants dans l'étape 5, il n'est pas certain que la commande implantée corresponde aux attentes du concepteur. Ce point sera repris dans la suite de ce chapitre pour montrer l'intérêt de la validation. Avant cela, la démarche pour obtenir le comportement maximum admissible de la PO par rapport aux contraintes définies est présentée.

3.2 Détermination du comportement maximal admissible du procédé vis-à-vis de contraintes pour des SED booléens

3.2.1 Algorithme

Pour formaliser l'algorithme support de notre démarche (étape 2, figure 2.5), le principe de la commandabilité définie par Ramadge et Wonham en l'adaptant aux SED booléens, est retenu. Cette adaptation s'exprime de la manière suivante : pour un procédé G défini par le 4-tuplet (P, Σ, δ, p_0), et un ensemble de spécifications K, il faut que la proposition suivante (Wang, 2000) soit vérifiée :

$\forall p \in P$, tel que p vérifie K

Si $\forall \alpha \in \Sigma u$ tel que $\delta(p, \alpha)$ vérifie K

Alors K est commandable, et K est l'ensemble des spécifications à respecter par le système.

En d'autres termes, la spécification K est commandable si et seulement si l'occurrence d'un événement non commandable α à partir d'un état vérifiant K ne conduit pas le procédé hors de la spécification K, c'est-à-dire vers un état qui ne vérifie pas K. Pour faciliter la mise en pratique de cette proposition, l'algorithme de synthèse (Tajer, 2005) utilise la notion d'états défendus et faiblement défendus, où les états défendus représentent des états ne respectant pas la spécification K lors de l'occurrence d'un événement non commandable α, et les états faiblement défendus représentent des états conduisant vers un état défendu sous l'occurrence d'un événement non commandable α.

L'algorithme consiste, à partir du modèle global du procédé et des contraintes modélisées selon les principes précédents, à générer l'automate à états booléens du superviseur *SUP*. L'automate *SUP* est décrit par 6-uplet (Σ, Q, Δ, E, Z, q_0), avec Σ l'ensemble des événements, Q l'ensemble des états, q_0 l'état initial et Δ une fonction de transition partielle Δ: $Q \times \Sigma \rightarrow Q$, un vecteur d'entrées E, et un vecteur de sorties Z. Une transition de l'automate *SUP* est définie par un triplet (q, α, q'), q étant l'état de départ, α l'événement correspondant à la transition et q' l'état d'arrivée. La fonction de transition peut être étendue aux séquences d'événements de la manière suivante : Soit Σ^* l'ensemble des séquences d'événements de longueur finie, ε la séquence vide et w une séquence finie d'événements :

$\Delta : Q \times \Sigma^* \rightarrow Q$, $\Delta(q, \varepsilon) = q$ et $\Delta(q, w\alpha) = \Delta(\Delta(q,w), \alpha)$

Cet algorithme reçoit en entrée un automate à états booléens G représentant le modèle global du procédé, un ensemble K de spécifications logiques représentant les contraintes de sûreté et de vivacité ainsi que la structure initiale de l'automate résultant donnée par (Σ, $\{q_0\}$, δ, q_0). Un état q de *SUP* est caractérisé par un état de G, un état du vecteur E, et un état du vecteur Z ainsi l'état initial s'exprime par $q_0 = (p_0, E_0, Z_0)$. Cet algorithme est basé sur trois étapes :

La première étape (figure 2.4.a) traite les événements commandables liés à l'activation ou à la désactivation d'une sortie de la commande ($\downarrow$z ou $\uparrow$z). Ces événements décrivent les ordres envoyés de la commande vers la PO. Les contraintes sont structurées en fonction des événements commandables Σ_c. Ainsi, pour chaque événement commandable $\sigma \in \sum_c$ est associé un ensemble de contraintes K_σ, $K_\sigma \subset K$. Par conséquent, une itération consiste, pour une évolution commandable ((q, σ, q'), ($\sigma \in \sum_c$)), à vérifier l'ensemble des contraintes K_σ. Cette vérification est obtenue par la valeur du vecteur d'entrées E_q, du vecteur de sorties Z_q et de l'événement *s* (l'occurrence de l'événement commandable *s* permet de changer la valeur de l'élément correspondant dans le vecteur Z pour obtenir $Z_{q'}$).

- Si dans l'ensemble K_σ, les équations logiques associées à σ sont vraies alors l'évolution correspondante n'est pas défendue et la transition correspondante est ajoutée à l'ensemble des transitions Δ. En effet, cet ordre est prévu par la commande et autorisé par le superviseur *SUP*. L'état aval q' atteint depuis q, suite à l'occurrence de σ, se distingue de l'état amont q par le vecteur de sorties $Z_{q'}$. L'état q' est ensuite ajouté à l'ensemble des états Q s'il n'en fait pas déjà partie.

- Si l'ensemble des contraintes K_σ n'est pas vérifié (les équations ne sont pas vraies), alors l'évolution (q, σ, q') est défendue.

a- Première étape :

$\forall q \in Q$ faire :

Si $\exists\ \sigma \in \sum_c$ tel que $\Delta(q,\sigma) = q'$ **Alors**
  Si K_s est vrai **Alors**
  - $\Delta = \Delta \cup \{(q, \sigma, q')\}$;
    $Q = Q \cup \{q'\}$ Si $q' \notin Q$.
  Sinon
    (q, σ, q') est une évolution défendue

b- Deuxième étape :

1er pas : **Si** $\exists\ \Delta(q, \alpha) = q'$ ($q' \in Q$, $\alpha \in \sum_u$) tel que q' ne vérifie pas K
  Alors q est défendu

2ème pas : **Si** $\nexists\ q \in Q$ tel que q un état défendu
  Alors Fin.
  Sinon $\forall q \in Q$
    Si
    - q n'est pas défendu,
    - $\exists\ w \in \sum^*$ tel que ($\forall \alpha \in w$, $\alpha \in \sum_u$) et ($q_w = \Delta(q, w)$ et q_w est défendu)

    Alors q est faiblement défendu.

c- Troisième étape :

$\forall q \in Q$ tel que q est un état défendu ou faiblement défendu.
- $\forall\ (q, \alpha, q') \in \Delta$, $\Delta = \Delta - \{(q', \alpha, q)\}$.
- $\forall\ (q', \alpha, q) \in \Delta$, $\Delta = \Delta - \{(q, \alpha, q')\}$.
- $Q = Q - \{q\}$.
- $\forall\ q' \in Q$ tel que $\nexists (w \in \sum^*$ et $q' = \Delta(q_0, w))$ faire $Q = Q - \{q'\}$

Figure 2.4. Algorithme de synthèse (Tajer, 2005)

La deuxième étape (figure 2.4.b) consiste à construire s'il n'existe pas déjà, à partir d'un état courant q, les évolutions de *SUP* caractérisées par des événements non commandables. Ces événements non commandables correspondent aux réactions de la PO par rapport aux ordres de la commande. Dans cette étape, il y a deux pas :

- Le premier pas concerne l'identification des états défendus conduisant vers des états à interdire où les contraintes K ne sont pas vraies. Pour cet état q, il existe un événement non commandable α possible à partir de l'état correspondant du procédé q' tel que l'état atteignable ne vérifie pas K. D'après la définition de la commandabilité, si le comportement du système ne contient aucun état défendu alors il est commandable et peut être considéré en tant que superviseur.
- Le deuxième pas sert à identifier les états faiblement défendus. Ce sont tous les états à partir desquels il existe une séquence d'événements *w* non commandables permettant d'atteindre un état défendu.

La troisième étape (figure 2.4.c) consiste à retirer dans un premier temps tous les états défendus et faiblement défendus ainsi que toutes les transitions amont et aval qui leurs sont

associées. Dans un second temps, les états non atteignables à partir de l'état initial sont retirés de l'automate obtenu.

3.2.2 Discussion

D'après la théorie de supervision, le superviseur obtenu par synthèse doit être commandable. Les travaux de Wang (Wang, 2000) vérifient la commandabilité par les deux propriétés suivantes :

- $(\forall q \in Q, \alpha \in \Sigma_u)$ si q vérifie l'ensemble des contraintes K et s'il existe une fonction de transition $\Delta(q, \alpha)$ alors $\Delta(q, \alpha)$ doit vérifier l'ensemble K. Cela signifie que l'occurrence d'un événement non commandable ne doit pas conduire dans un état ne vérifiant pas les contraintes.
- $(\forall q \in Q)$ si q vérifie l'ensemble des contraintes K alors $(\exists q_1,\ldots, q_n \in Q$, et $\alpha_1,\ldots, \alpha_n \in \Sigma)$ tels que $q_n=q$ et $\Delta(q_{i-1}, \alpha_i) = q_i$ doit vérifier l'ensemble K pour i variant de 1 à n. En d'autres termes, à partir de n'importe quel état vérifiant l'ensemble des contraintes, tous les états par lesquels l'état q_n est atteint, doivent vérifier l'ensemble des contraintes K.

Pour montrer que l'algorithme proposé respecte aussi la commandabilité, il suffit de montrer que les deux propriétés précédentes sont vérifiées :

- La première condition de commandabilité est vérifiée de manière triviale par la construction définie dans l'algorithme. Par l'intermédiaire du pas n°2 (figure 2.4.b), si un état est accessible par un événement non commandable et qu'il ne vérifie pas l'ensemble des contraintes K, alors l'état qui conduit vers cet état est défini comme défendu ou faiblement défendu, et éliminé ainsi que les transitions associées à chacun de ces états.
- La seconde condition se détermine de la même manière, la construction de l'automate du superviseur par l'algorithme proposé permet d'éliminer les états ne respectant pas les contraintes et ainsi obtenir un ensemble d'états atteignables qui respectent l'ensemble des contraintes. Donc, une séquence d'événements commandables ou non commandables va toujours conduire vers un état vérifiant l'ensemble des contraintes.

L'algorithme proposé répond par conséquent, au critère de commandabilité de la théorie de supervision. En utilisant les notions d'états défendus et faiblement défendus, l'algorithme

proposé peut être vu comme une adaptation aux automates booléens de l'algorithme de Kumar (Kumar, 1991). Dans le cas de cet algorithme, la complexité en nombre d'états s'exprime dans le pire cas, par $M^k 2^{\Sigma c+\Sigma u}$ où M est le nombre d'états de l'automate des contraintes et k le nombre de contraintes. En conservant cette notation, l'algorithme proposé génère $2^{\Sigma c+\Sigma u}$ états. Cette importante diminution vient du fait qu'aucun produit synchrone n'est effectué entre le modèle du procédé et celui des contraintes.

3.3 Démarche de validation

L'avant dernière étape (étape 5 figure 2.2.) de l'approche de synthèse hors ligne, retire les états bloquant ainsi que les transitions qui lui sont associées. Cette dernière étape assure que la commande est sûre et sans blocage mais n'assure pas qu'elle corresponde aux attentes du concepteur. Il faut donc un véritable concept de validation et non plus de synthèse automatique de la commande, certaines fonctionnalités de ces étapes doivent être modifiées et nous proposons dans l'étape 5 de procéder à l'analyse de l'intersection entre la commande et le comportement maximal admissible du procédé. En effet, les informations contenues dans cette intersection sont extrêmement riches et peuvent renseigner le concepteur ou l'expert sur d'éventuels blocages ou des évolutions qui ne respectent pas certaines contraintes.

Cette démarche de validation par synthèse peut permettre soit, à l'expert de valider ses modèles de PO ainsi que l'ensemble de contraintes, soit au concepteur de valider sa commande.

Dans le premier cas, l'expert cherche à s'assurer que les modèles d'éléments de PO (ElPO) et les contraintes sont correctement définis (figure 2.5). Pour cela, il utilise un modèle de commande dont il est sûr et effectue les étapes de validation par synthèse. Lors de l'analyse de l'intersection, les erreurs qui peuvent apparaître se trouve seulement au niveau des modèles d'ElPO ou des contraintes.

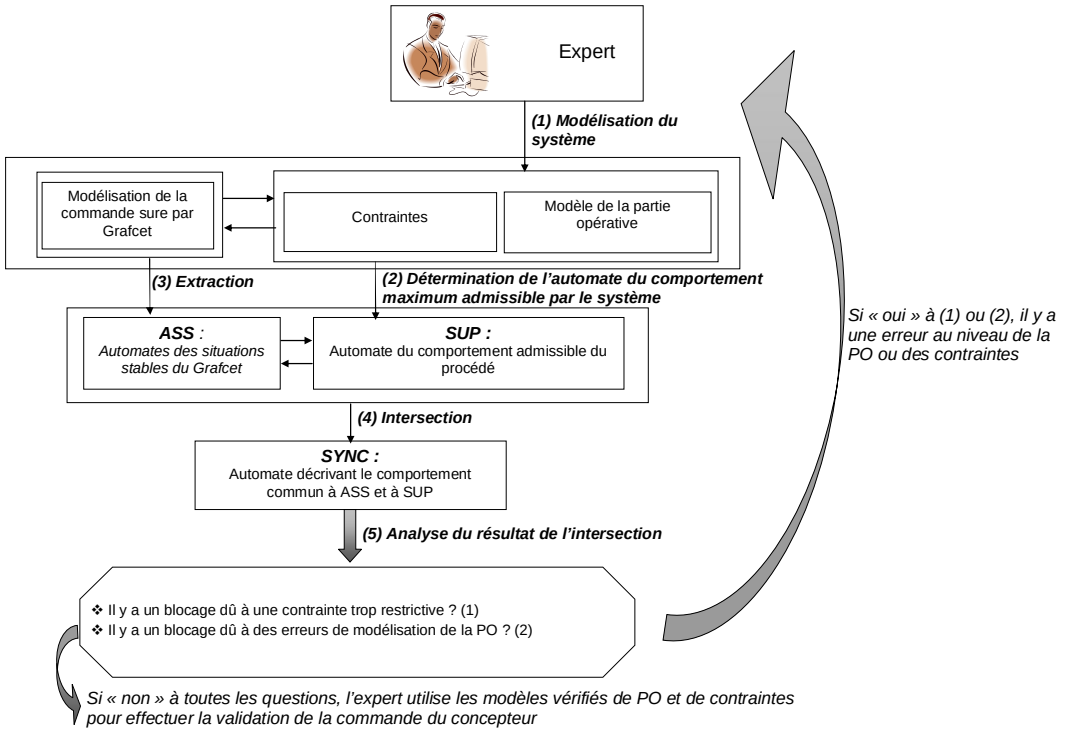


Figure 2.5. Phase de validation des modèles d'ElPO et des contraintes par l'expert

Dans le second cas (figure 2.6.), le concepteur valide sa commande à partir des contraintes et des modèles d'ElPO vérifiés par l'expert au préalable. Lors de cette seconde étape, les erreurs possibles sont seulement au niveau de la commande.

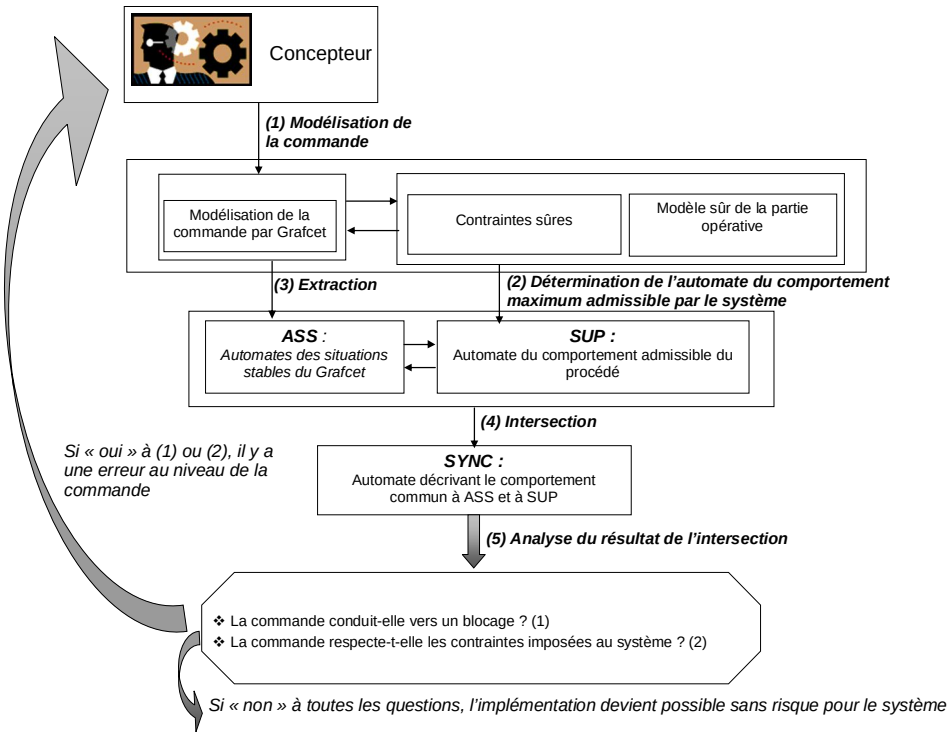


Figure 2.6. Phase de validation de la commande par le concepteur

A l'issue de la phase de validation, nous pouvons proposer au concepteur comme dans la démarche de synthèse, d'implanter l'automate d'intersection obtenu à la fin de la phase de validation. Si le concepteur a correctement intégré toutes les informations de la phase de validation, il ne devrait plus y avoir de blocage et d'évolution ne respectant pas les contraintes. Le concepteur peut soit implémenter directement la commande qu'il a défini sous forme de Grafcet, soit il implémente l'automate COR obtenu après l'étape 5 figure 2.2. L'implémentation est faite dans un langage de programmation d'automate programmable industriel, tels que le Ladder, le texte structuré ou la liste d'instructions (IEC, 2003). Dans le second cas, la commande implantée résulte donc de la phase de validation et est, par conséquent, différente du Grafcet initial fourni par le concepteur en tout début de la phase de validation. Cette méthode d'implémentation nous permet de vérifier le principe de WYPIWYE[2] (Berry, 1989) car le modèle qui est validé est le même qui nous permet de générer le code pour l'API.

4 Aide à l'élaboration de la commande

Les informations sur les blocages en exécution réelle et les corrections éventuelles qui pourraient porter sur des inhibitions d'ordre au niveau de la commande par la prise en compte des contraintes sont donc visibles au niveau de l'opération d'intersection entre *ASS* et *SUP*. Ainsi, à l'issue de l'opération d'intersection, l'analyse de l'automate résultant peut conduire à deux cas :

- Soit aucun blocage, aucune correction ne sont visibles au niveau de l'intersection ce qui signifie que la commande respecte les spécifications de sécurité et de vivacité imposées au système. Dans le cas de l'utilisation de l'approche par l'expert, il peut être sûr de la bonne définition des modèles d'ElPO et des contraintes. Dans le cas de l'utilisation par le concepteur, il peut implanter la commande dans un API en sachant qu'il n'y aura aucun risque pour le matériel opératif ou les produits.

[2] WYPIWYE : What You Prove Is What You Execute

- Soit des corrections ou des blocages sont détectés, la commande ne respecte donc pas les spécifications imposées au système :
  - S'il s'agit d'une situation de blocage, que ce soit dans la phase de validation des modèles du système ou dans la phase de validation de la commande, l'expert ou le concepteur analyse la situation bloquante grâce aux données élaborées à partir des informations contenues dans les régions de l'automate d'intersection et agit en conséquence.
  - S'il s'agit de corrections proposées sous forme d'inhibitions d'ordres, ceci signifie que les contraintes imposées au système ne sont pas respectées. La liste des contraintes non respectées est remontée pour en déduire d'éventuelles modifications. Dans la phase de validation des modèles, l'expert peut agir sur la définition des contraintes qui peuvent être trop restrictives. Dans la phase de validation de la commande, le concepteur sait que la commande est erronée par rapport aux contraintes qu'il doit respecter.

4.1 Présentation synthétique de l'information proposée en cas d'erreur

4.1.1 Automate agrégé

Pour aider le concepteur ou l'expert dans la compréhension des erreurs et dans les éventuelles modifications, les informations sont présentées à travers un automate agrégé. Cet automate regroupe les informations sur la situation de la commande, l'ensemble des étapes actives, les ordres autorisés et ceux interdits. Ses états correspondent aux régions de *SYNC*. Les évolutions intermédiaires représentant les états instables d'une région, sont remplacées par un état décrivant la situation stable de la commande. Cet état est ainsi muni de trois ensembles : l'ensemble des étapes de la situation correspondante du Grafcet, l'ensemble des ordres autorisés par le superviseur et l'ensemble des ordres interdits par le superviseur. Les ordres à autoriser et à interdire sont déduits des transitions commandables de la région de l'automate *SYNC*. L'automate agrégé (figure 2.7) ainsi généré est donné par des états qui sont reliés par des transitions non commandables reliant les régions de *SYNC*. L'automate agrégé obtenu à partir de *SYNC* est défini par le 5-uplet (Σ_u, ET, ϕ, $etat_0$, $etat_b$), avec :

- Σ_u l'ensemble des événements non commandables

- ET l'ensemble des états correspondants à une région de *SYNC*. A chaque état et∈ET est associé le 5-uplet (ORD_{et}, $PROH_{et}$, Sit_{et}, E_{et}, Z_{et}) où :
  - ORD_{et} : représente l'ensemble des ordres autorisés dans cet état,
  - $PROH_{et}$: représente l'ensemble des ordres interdits dans cet état,
  - Sit_{et} : représente l'ensemble des étapes du Grafcet correspondant à cet état,
  - E_{et} : représente le vecteur d'entrée de la commande correspondant à cet état.
  - Z_{et} : représente le vecteur de sortie de la commande correspondant à cet état.
- $\phi : \Sigma_u \times ET \rightarrow ET$ est une fonction partielle représentant les transitions de l'automate agrégé.
- $état_0$ est l'état initial.
- $état_b$ est l'état de blocage.

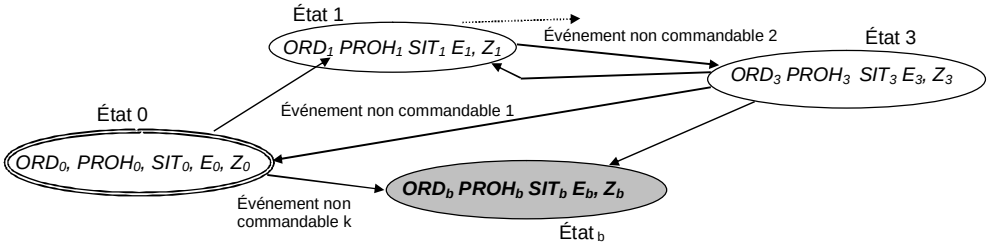


Figure 2.7. Modèle de l'automate agrégé

4.1.2 Démarche d'analyse

Pour structurer la démarche d'analyse, il est proposé à l'expert et au concepteur de procéder étape par étape.

Dans le cas de l'analyse du modèle d'intersection pour valider les modèles du système, l'expert analyse dans :

- La 1ère étape, les paramètres ORD_{et} et $PROH_{et}$ de l'automate agrégé qui déterminent si les contraintes sont respectées ou non dans l'état concerné. Il peut être nécessaire dans cette étape d'afficher aussi le superviseur *SUP*. L'expert peut dans cette première étape s'apercevoir qu'une contrainte est mal définie ou est trop restrictive.

- La 2ème étape, les ElPO mis en cause dans le blocage. L'expert peut visualiser le ou les ElPO associés au blocage. Lors de cette seconde étape, il peut se rendre compte d'une erreur dans la modélisation d'un ElPO.

Dans le cas de l'analyse du modèle d'intersection pour valider la commande par le concepteur, celui-ci analyse dans :

- La 1ère étape, la spécification Grafcet proposée en connaissant la situation complète du système grâce aux informations Sit_{et}, E_{et} et Z_{et} de l'automate agrégé. Cette analyse permet de comprendre les différentes évolutions possibles au niveau de la commande, c'est-à-dire, les différentes transitions franchissables. Ainsi par cette première analyse, il peut s'apercevoir d'une mauvaise réceptivité associée à une transition ou un mauvais envoi de commande au niveau de l'étape.
- La 2ème étape, les paramètres ORD_{et} et $PROH_{et}$ de l'automate agrégé qui déterminent si les contraintes sont respectées ou non dans l'état concerné. Ici aussi le superviseur *SUP* peut être utilisé. Lors de cette seconde étape, il peut constater que la commande ne respecte pas les contraintes et la modifier en conséquence.
- 3ème étape : les ElPO pour avoir confirmation de la cause d'une évolution système ou la conséquence d'une action.

4.2 Étude de cas

En vue d'illustrer la démarche de validation hors-ligne proposée, l'exemple du tri de caisses tiré de (Bossy et al., 1989) et traité par Tajer (Tajer, 2005) est proposé. Seule la partie sur l'interprétation d'un blocage est détaillée ici, la construction du modèle de PO, l'obtention du superviseur et de l'intersection de l'ASS et du superviseur sont illustrées brièvement dans l'annexe 1.

L'objectif de cette application est de trier automatiquement des caisses de deux tailles différentes (figure 2.8.). Le processus industriel se compose du convoyeur 1 qui apporte les caisses les unes après les autres, d'un axe composé de deux vérins double effet V_1 et V_2, de deux vérins simple effet V_3 et V_4, et de deux convoyeurs (convoyeur 2 et 3) permettant l'évacuation des caisses. Selon la taille de la caisse, l'axe composé de V_1 et de V_2, place les caisses devant le vérin V_3 ou devant le vérin V_4.

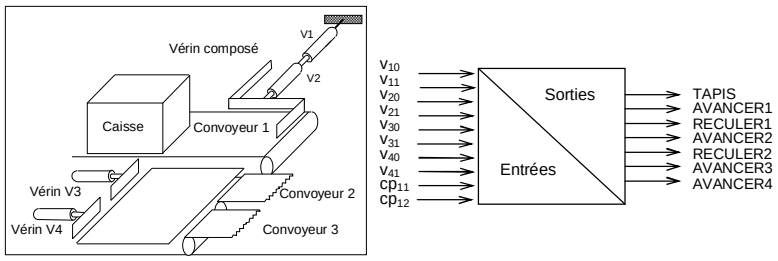


Figure 2.8. Système du tri de caisses

La notation est la suivante : v_{i0} correspond à la position rentrée du vérin V_i, v_{i1} à la position sortie du vérin V_i, cp_{11} correspond au capteur de petite caisse et cp_{12} au capteur de grande caisse. Il est supposé que l'information sur le type de caisses est mémorisée jusqu'à l'arrivée d'une nouvelle caisse. Les sorties du système sont définies par : *TAPIS* correspond à l'ordre d'activer les trois convoyeurs, *AVANCER1* (*AVANCER2*) à l'ordre de sortie du vérin V_1 (resp.V_2), *RECULER1* (*RECULER2*) à l'ordre de rentrée du vérin V_1 (resp. V_2), *AVANCER3* (*AVANCER4*) à l'ordre de sortie du vérin simple effet V_3 (resp. V_4). Le système est mis en marche par l'action du bouton *dcy*. Du point de vue de la commande, les vecteurs d'entrées et de sorties correspondant au système de tri de caisses, sont définis de la façon suivante : E = {*dcy*, v_{10}, v_{11}, v_{20}, v_{21}, v_{30}, v_{31}, v_{40}, v_{41}, cp_{11}, cp_{12}}, Z = {*TAPIS*, *AVANCER1*, *AVANCER2*, *RECULER1*, *RECULER2*, *AVANCER3*, *AVANCER4*}.

Les contraintes posées pour cette application sont les suivantes :

- Interdiction d'avancer et de rentrer le vérin V_1 :

$$\text{RECULER1} \wedge \text{AVANCER1} = 0 \quad (2.1)$$

- Interdiction d'avancer et de rentrer le vérin V_2 :

$$\text{RECULER2} \wedge \text{AVANCER2} = 0 \quad (2.2)$$

- Interdiction d'avancer V_3 si le vérin V_1 est en position v_{11}

$$\text{AVANCER3} \wedge v_{11} = 0 \quad (2.3)$$

- Interdiction d'avancer V_3 si le vérin V_2 est en position v_{21}

$$\text{AVANCER3} \wedge v_{21} = 0 \quad (2.4)$$

- Interdiction d'avancer le vérin V_2, si une petite caisse est présente

$$\text{AVANCER2} \wedge cp_{11} = 0 \quad (2.5)$$

- Interdiction d'avancer les vérins V_3 et V_1 en même temps

$$AVANCER3 \wedge AVANCER1 = 0 \quad (2.6)$$

- Interdiction d'avancer les vérins V_3 et V_4 en même temps

$$AVANCER3 \wedge AVANCER4 = 0 \quad (2.7)$$

- Interdiction d'avancer le vérin V_4 si pas de grandes caisses

$$AVANCER4 \wedge \neg\, cp12 = 0 \quad (2.8)$$

Dans cette partie, nous analysons deux types d'erreurs : absence d'envoi d'un ordre à la PO et erreur au niveau de la modélisation de la PO. La première erreur correspond à une erreur pouvant être faite par le concepteur, alors que la deuxième est une erreur pouvant être faite par l'expert. Pour chacune de ces erreurs, l'intérêt de la démarche d'analyse proposée pour aider le concepteur ou l'expert à déterminer ces erreurs sera montré.

4.2.1 Absence d'un ordre envoyé à la partie opérative

Lors de la phase de validation de la commande, le concepteur peut proposer une commande (figure 2.9) qui contient une absence d'ordre envoyé vers la PO sachant que les modèles de PO et les contraintes ont été préalablement validés. Une fois qu'une caisse est détectée, une temporisation est lancée avant de la placer devant V_3 ou V_4, pour permettre à la caisse d'être devant le vérin composé de V_1 et V_2. Enfin la caisse est évacuée.

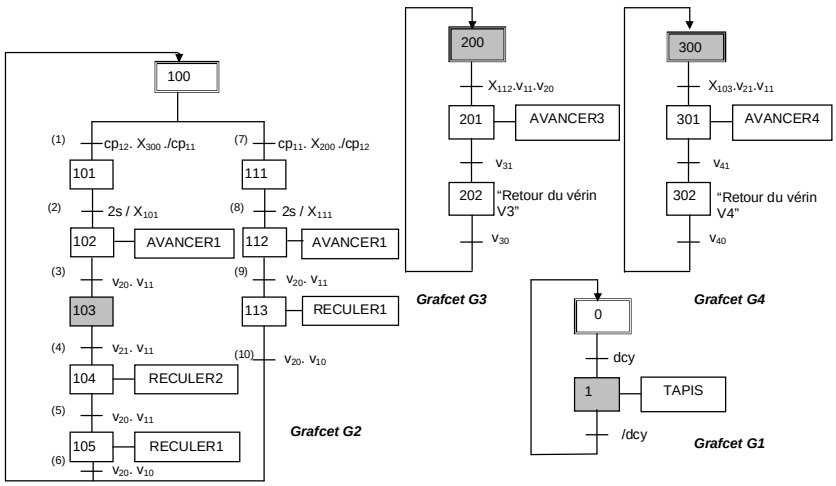


Figure 2.9. Grafcet de commande de l'exemple de tri de caisses

La figure 2.10 montre que la commande conduit à un blocage. La trace conduisant au blocage est proposée au concepteur pour lui permettre de visualiser et d'analyser l'erreur commise.

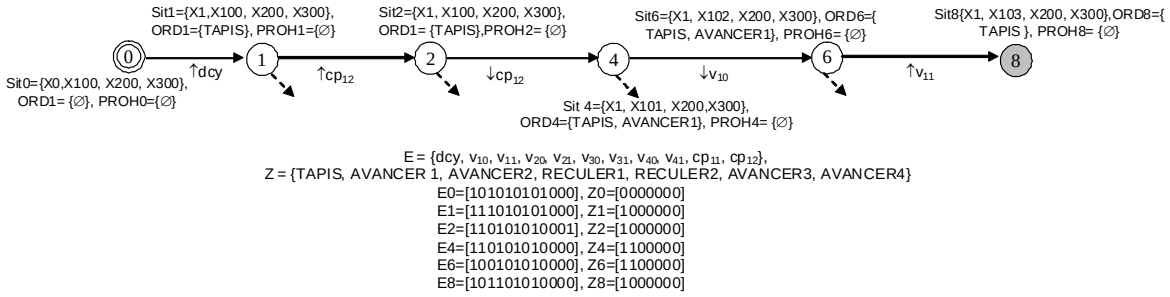


Figure 2.10. Visualisation de la séquence de blocage[3]

La figure 2.10 illustre la situation de blocage car depuis l'état 8, aucune évolution n'est possible. Dans cet état, la commande (figure 2.9) se trouve dans la situation {X1, X103, X200, X300} du Grafcet. Quant à la PO, les deux vecteurs de sorties et d'entrées E_8= [1,0,1,1,0,1,0,1,0,0,0,0] et Z_8= [1,0,0,0,0,0,0] indiquent que le système est en marche (*dcy = 1*) le vérin V_1 est en position sortie, et les vérins V_2, V_3, et V_4 sont en position rentrée. Au niveau des sorties, seul l'ordre *TAPIS* est envoyé.

1ère étape : Analyse du Grafcet de commande

Par l'analyse du Grafcet, le concepteur peut en déduire que les évolutions possibles au niveau de la commande sont soit le passage à 0 de *dcy*, soit la validation et le franchissement de la transition notée (4) dans le Grafcet G2 dont la réceptivité est $v_{21}.v_{11}$.. Pour que la réceptivité $v_{21}.v_{11}$ soit vraie, le système se trouvant dans la configuration où le vérin V_2 est rentré, il faut que l'ordre *AVANCER2* soit envoyé pour faire sortir le vérin V_2 et donc, par conséquent, activer la position v_{21}. A ce stade, soit le concepteur détermine directement l'erreur (il manque l'ordre *AVANCER2* dans l'étape 103) soit, il a besoin d'informations supplémentaires lui permettant de vérifier que cet ordre n'est pas interdit par les contraintes.

2ème étape : Analyse du superviseur

Le concepteur peut analyser les informations provenant du superviseur. L'état du superviseur correspondant à l'état 8 de l'automate agrégé (figure 2.10), est l'état q_{16} où les ordres autorisés sont : *AVANCER2, …*

[3] l'état initial est modélisé par un double cercle

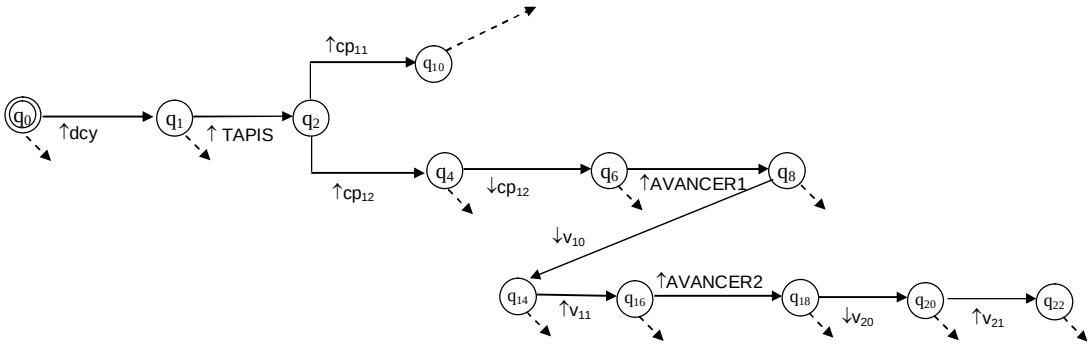


Figure 2.11. Extrait de l'automate superviseur

L'ordre *AVANCER2* n'est pas interdit au niveau de l'automate de la PO contrainte (figure 2.11). Donc, le concepteur détermine que l'erreur vient de l'absence de l'ordre *AVANCER2*. Ceci lui est confirmé avec l'information de l'ElPO associée à l'événement non commandable v_{21}.

3ème étape : Analyse de la PO.

Pour être certain de son raisonnement, le concepteur peut visualiser l'ElPO qui l'intéresse. A partir de cette dernière information (figure 2.12), le concepteur a confirmation que pour avoir l'occurrence de v_{21}, il faut l'envoi de *AVANCER2*. Le concepteur peut donc en conclure que pour que la réceptivité $v_{21}.v_{11}$ soit vraie, il faut que l'ordre *AVANCER2* soit envoyé vers la PO, qu'aucune contrainte ne contredit à ce stade l'envoi de l'ordre et que seul un oubli de l'envoi de l'ordre au niveau de la commande génère le blocage.

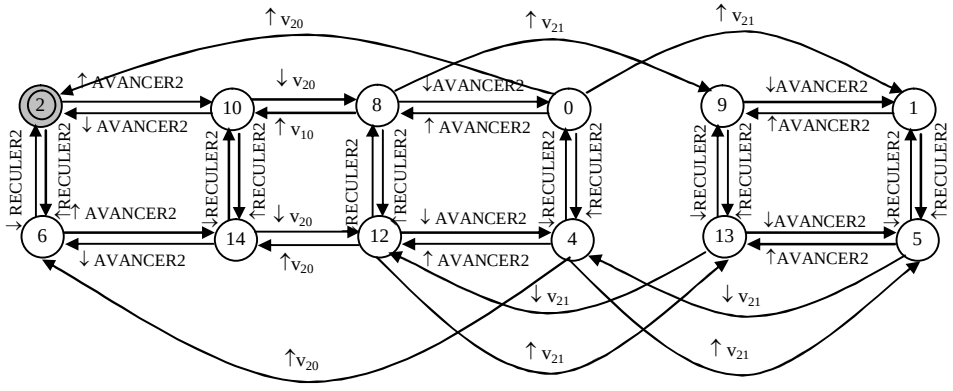


Figure 2.12. Modèle du mouvement du Vérin V2

4.2.2 Erreur au niveau de la partie opérative.

Lors de la phase de validation des modèles du système, l'expert propose la commande sûre de la figure 2.13, pour détecter si les modèles de PO sont correctement définis.

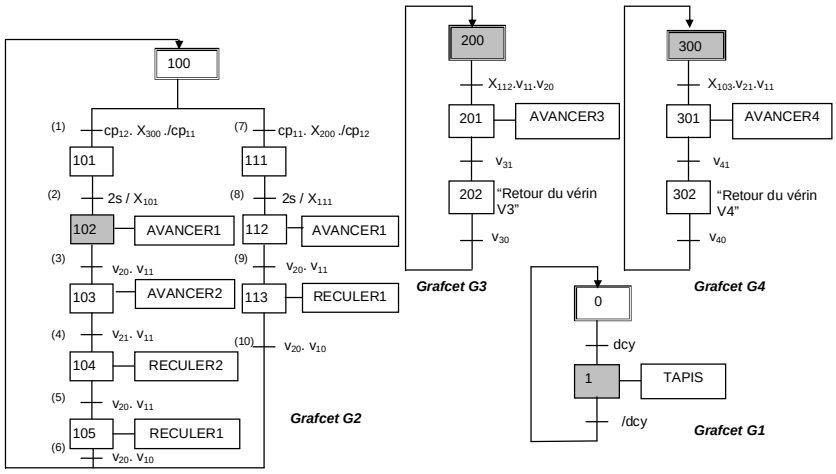


Figure 2.13. Grafcet d'une commande sûre du tri de caisses

La trace conduisant au blocage est présentée à l'expert en figure 2.14. Depuis l'état 6, aucune évolution n'est possible. Dans cet état, les Grafcets (figure 2.13) se trouvent dans la situation {X1, X102 X200, X300}. Les deux vecteurs de sorties et d'entrées E_6= [1,0,1,1,0,1,0,1,0,0,0,0] et Z_6= [1,1,0,0,0,0,0] indiquent que le système est en marche (*dcy* = 1), le vérin V_1 est en position sortie, et les vérins V_2, V_3, et V_4 sont en position rentrée. Au niveau des sorties, les ordres *TAPIS* et *AVANCER1* sont envoyés vers la PO. L'analyse de la situation des Grafcets permet à l'expert d'avoir l'information sur la situation du système et de la commande.

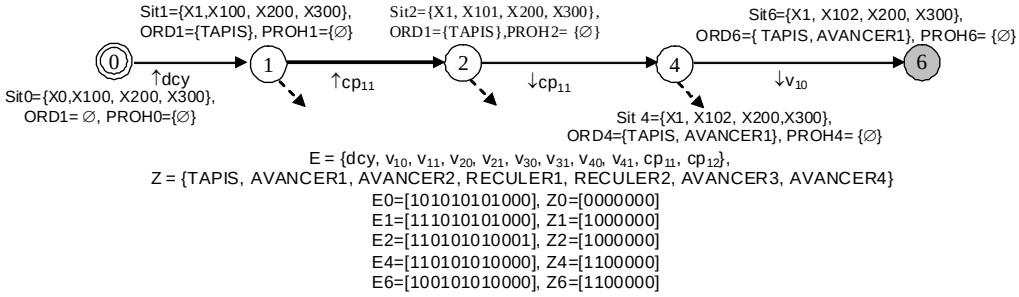


Figure 2.14. Visualisation de la séquence de blocage

1ère étape : Analyse du superviseur

Pour déterminer l'origine du blocage, l'expert analyse le superviseur (figure 2.15). Il peut constater que l'ordre *AVANCER1* est bien autorisé par le superviseur. A ce niveau, aucune contrainte n'est donc violée et même si la suite d'événements $\downarrow v_{10}$ puis $\uparrow v_{21}$ suite à l'occurrence de *AVANCER1* ne paraît pas logique, il faut analyser les ElPO pour pouvoir conclure.

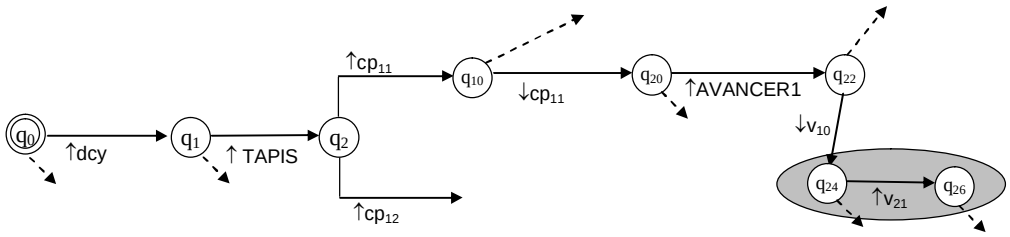


Figure 2.15. Extrait de l'automate superviseur

2ème étape : Analyse de la PO.

L'expert visualise les ElPO du vérin V_1 (figure 2.16) et du Tapis car ce sont les ElPO mis en jeu lors du blocage.

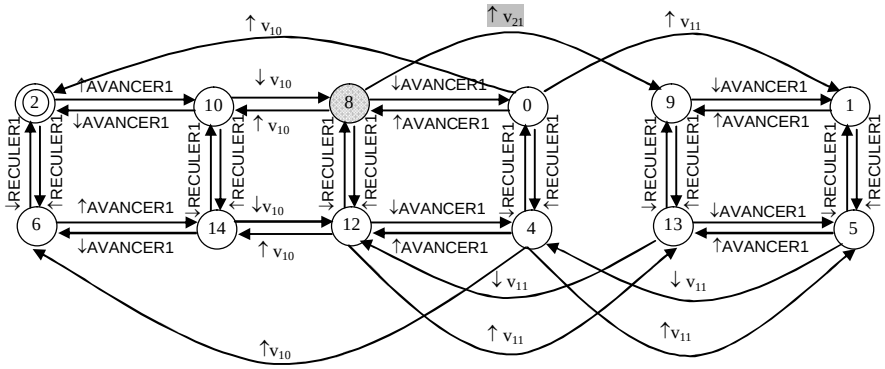


Figure 2.16. Modèle du mouvement du Vérin V1

L'analyse des figures 2.16 et 2.12, montre que v_{21} appartient au modèle du mouvement V_1 et au modèle du mouvement V_2, ce qui est logiquement impossible dans la configuration du système de tri de caisses. La figure 2.16 décrit la suite d'événements lorsque *↑AVANCER1* est autorisé : apparition de $\downarrow v_{10}$ puis apparition de $\uparrow v_{21}$. Avec ces informations, l'expert peut se rendre compte qu'il y a une erreur de modélisation de la PO du vérin V_1. Logiquement, l'événement attendu après l'état 8 de la figure 2.16, est $\uparrow v_{11}$ et non $\uparrow v_{21}$. En prenant en compte cette erreur de modélisation, la figure 2.16 va évoluer pour retrouver la suite logique $\downarrow v_{10}$ puis $\uparrow v_{11}$ suite à l'occurrence de *AVANCER1*.

5 Bilan par rapport à nos attentes

Cette partie fait un bilan sur l'approche de validation de la commande par une opération de synthèse par rapport à nos attentes fixées dans le chapitre 1 (§5). Le bilan se fait selon trois critères :

- Définition du modèle de PO et prise en compte du concepteur,
- Génération d'explications en cas d'erreur,
- Mise en œuvre et fiabilité de l'approche au niveau de la sécurité du système.

5.1 Définition de la partie opérative et prise en compte du niveau du concepteur

L'approche proposée dans ce chapitre est basée sur un modèle de PO défini par des états booléens de capteurs et d'actionneurs et, est définie par une opération de synthèse. Quelle que soit la méthodologie de modélisation de la PO, démarche globale ou démarche modulaire, et des modèles utilisés pour la PO, automates rudimentaires ou automates booléens, l'opération de synthèse prenant en compte les contraintes, sous forme d'automates ou d'équations logiques, engendre des problèmes d'explosion combinatoire, limitant ainsi la méthode à des systèmes de taille réduite. Cet aspect limite donc l'application de cette approche à des systèmes simples avec peu d'entrées/sorties. Cependant, des travaux proposent des démarches itératives et modulaires de synthèse (Vahidi et al., 2006 ; Endsley et al., 2006 ; Gouyon et al., 2004), permettant ainsi de diminuer les risques d'explosion combinatoire pour des systèmes complexes. Toutefois, toutes ces approches, comme celle proposée dans nos travaux, sont confrontées au problème de l'implémentation dans l'API.

Concernant la granularité du modèle étudié, notre approche se situe au niveau le plus bas, c'est-à-dire au niveau des capteurs/actionneurs. La granularité est donc dans ce cas fixe et non modifiable et ne permet donc pas de moduler la définition de la PO. Un des objectifs fixés est de pouvoir proposer une approche qui permet quel que soit le niveau de connaissances du concepteur de piloter un système dans sa globalité.

Les modèles obtenus ont leurs limites tant au niveau de la représentation du système, des notions temporelles, que des évolutions simultanées des événements. En effet, les modèles étant seulement fonction de l'état des capteurs et des actionneurs, ils ne permettent

pas de différencier toutes les situations. De plus, la possibilité d'évolutions simultanées dues à l'utilisation d'un API ne peut pas être prise en compte dans les automates à états booléens. L'écart entre le modèle de PO proposé et le comportement réel du système existe et est lié essentiellement au fait que l'aspect temporel, le fonctionnement cyclique de l'API et les technologies utilisées ne sont pas pris en compte. Actuellement, certains travaux s'orientent vers une méthodologie de modélisation bas niveau de la PO intégrant directement la technologie des pré-actionneurs, des actionneurs et des capteurs (Rohee et al., 2006 ; 2007), (Philippot, 2006). Nous verrons dans la suite de ce manuscrit, une proposition de modélisation de PO différente des précédentes prenant en compte l'environnement de calcul.

5.2 Définition des contraintes et génération d'explications

La définition des contraintes se fait intuitivement à partir du connecteur d'implication « Si…Alors… » modélisé par des équations logiques. En cas d'erreur, que ce soit pour la phase de validation de modèle système ou la phase de validation de la commande, les informations retournées au concepteur ou à l'expert sont fines et très riches. Elles nécessitent donc une très bonne connaissance du système global pour analyser correctement le problème. De plus l'approche utilise plusieurs outils de modélisation : automates à états, équations logiques, et la compréhension d'une erreur nécessite la connaissance de ces modèles. Dans cette approche, la compréhension des erreurs et les éventuelles modifications sont à la charge du concepteur dans le cas de la validation de la commande et de l'expert dans le cas de la validation des modèles du système qui détient toutes les informations et compétences. Pour ne pas remonter trop d'informations, l'analyse est découpée en plusieurs étapes.

Dans le cas de la validation des modèles du système sachant la commande sûre, l'expert connaît la valeur du vecteur d'entrées sorties du système et les étapes actives dans le Grafcet lors du blocage :

- La 1ère étape informe l'expert sur l'ensemble des ordres autorisés et interdits après l'étape de synthèse. Cette étape lui permet de détecter une erreur au niveau de la définition des contraintes
- La 2ème étape informe l'expert sur les modèles d'ElPO mis en jeu dans le blocage. Les éléments (capteurs et actionneurs) mis en cause dans un début d'analyse pourront être visualisés dans le ou les sous-modules auxquels ils appartiennent. Dans cette étape, l'expert peut détecter une erreur dans les modèles de PO.

Par l'analyse des deux étapes, l'expert peut détecter une erreur liée à un manque de précision du cahier des charges, ou une mauvaise modélisation de la PO. Ces deux types d'erreur ne font pas l'objet de ces travaux de thèse. De plus, il est à remarquer que si le modèle de commande change, il pourra révéler de nouvelles erreurs au niveau de la PO ou des contraintes.

Dans le cas de la validation de commande, le concepteur connaît la valeur du vecteur d'entrées sorties du système et les étapes actives dans le Grafcet lors du blocage :

- La 1ère étape consiste à analyser la spécification Grafcet proposée en connaissant la situation complète du système. C'est-à-dire que le concepteur est informé sur la valeur du vecteur d'entrées/sorties, ainsi que les étapes actives dans les différents Grafcets. Cette analyse permet de comprendre les différentes évolutions possibles au niveau de la commande, c'est-à-dire, les différentes transitions franchissables. Cette étape peut permettre au concepteur de détecter des mauvaises réceptivités ou des absences d'actions sur une étape.
- La 2ème étape informe le concepteur sur l'ensemble des ordres autorisés et interdits après l'étape de synthèse. Dans cette étape, le concepteur peut voir les contraintes que sa commande ne respecte pas.
- La 3ème étape confirme au concepteur l'erreur qu'il a déterminée, en analysant les modèles des ElPO.

L'analyse des trois types d'informations, doit permettre au concepteur de commande de détecter une mauvaise réceptivité associée à une transition, l'envoi d'une mauvaise commande. Cette analyse demande beaucoup de connaissances de la part du concepteur et nécessiterait d'être effectuée par un expert ! Il est préférable que le déblocage en cas d'erreur soit effectué par le concepteur sans aide extérieure d'un expert, en particulier si le système est utilisé de façon distante.

Nous le verrons dans le chapitre suivant, nos travaux (Marangé et al., 2007 d ; 2007 e et 2007 f) visent à améliorer la modélisation des contraintes en proposant un découpage en fonction de leurs types (sécurité ou vivacité) et de leurs conditions intrinsèques (statique, dynamique), ainsi qu'un filtrage à deux niveaux, l'un pour apporter des explications au niveau du concepteur et l'autre pour assurer un maximum de sécurité.

5.3 Mise en place et fiabilité de l'approche

Nous l'avons souligné dans le paragraphe précédent, la modélisation globale de la PO du système limite son utilisation à des systèmes simples avec peu d'entrées/sorties

De plus la démarche est assez lourde en modèles et en étapes de calcul :

- Transformer la commande spécifiée par Grafcet en automate à états,
- Déterminer le superviseur par synthèse entre le modèle de PO et les contraintes,
- Croiser le superviseur et l'automate de commande,
- Analyser les blocages pour déterminer les éventuelles erreurs.

L'approche repose sur des équations logiques pour diminuer l'explosion combinatoire par rapport aux travaux de Kumar (Kumar, 1991) en proposant une modélisation de contraintes par automates. Cependant, l'approche par synthèse reste difficilement implémentable dans des automates programmables industriels. La possibilité d'utiliser cette approche en ligne a été étudiée dans les travaux de (Philippot et al., 2004 b), mais l'approche se heurte à la définition de la taille de la fenêtre d'observation du système. Lorsqu'elle est trop grande, la construction de la commande s'éloigne du temps réel et si elle est trop petite, la détection du blocage n'est pas systématique. La difficulté à déterminer la taille de cette fenêtre a en partie orienté nos recherches vers la validation en ligne développée dans le chapitre suivant.

L'approche de validation de la commande par une opération de synthèse est basée sur des outils et des méthodes formels tels que les automates à états et la théorie de supervision des SED (Ramadge et al., 1989). Cela permet d'assurer une grande fiabilité à l'approche de validation de la commande. La difficulté de cette approche vient des points suivants :

- L'élaboration d'un modèle complet du système est une tâche difficile liée à la taille de celui-ci,
- L'élaboration des contraintes définissant le comportement du système n'est pas méthodique. Une erreur sur la modélisation des contraintes peut masquer d'éventuelles erreurs de commande. Dans l'exemple du tri de caisses, les huit contraintes proposées ne sont pas correctement définies. En effet, le système nécessite 13 contraintes plus 12 autres si l'on prend en compte la technologie des vérins (chapitre 3 § 5).

- De même, une erreur sur la modélisation de la PO peut masquer d'éventuelles erreurs de commande. La phase de validation des modèles d'ElPO et des contraintes par l'expert peut éliminer ce risque. Cependant cette phase suppose d'avoir une commande sûre.

La démarche pour obtenir les modèles nécessaires à la détection des erreurs n'est pas complètement implémentée. Le logiciel Optigraf7 (Ndjab, 1999, Tajer, 2005) permet d'obtenir le *SUP*, l'*ASS*, le modèle d'intersection, ainsi que les séquences de blocage. Cependant, c'est au concepteur dans le cas de la validation de commande ou à l'expert dans le cas de la validation des modèles du système de faire l'étude pour trouver la cause.

6 Conclusion

L'approche basée sur l'opération de synthèse, permet de détecter des erreurs qui peuvent conduire à des blocages en exécution réelle, des erreurs au niveau de la commande, des erreurs au niveau de la modélisation de la PO et même de révéler si des contraintes sont trop restrictives. Avec la prise en compte des nouveaux modèles et algorithmes, la démarche peut être utilisée aussi bien pour effectuer une synthèse automatique de la commande que pour effectuer une validation de la commande avant l'implantation. En fait, la démarche va beaucoup plus loin qu'une simple assistance à l'élaboration de la commande mais permet d'effectuer une validation du comportement global d'un SED. Cependant, par rapport aux objectifs posés dans le chapitre 1, il semble difficile d'utiliser cette approche pour permettre une validation de commande facilement implémentable où le concepteur peut comprendre seul ces erreurs. En fonction des objectifs fixés, les travaux de thèse n'ont pas été poursuivi dans cette direction, cependant, des pistes de réflexions sont données en perspectives. Nous proposons donc, dans le chapitre suivant une approche adaptée au niveau du concepteur, et à base de filtres pour assurer la sécurité du système permettant de répondre aux critères précédents.

Chapitre 3 :
Approche de validation en ligne de la commande adaptée à l'automaticien

1 Introduction

Le chapitre précédent a montré la possibilité d'utiliser des travaux de synthèse pour faire de la validation de commande. Cependant par rapport aux objectifs initiaux fixés, cette approche hors ligne ne répond pas complètement aux attentes. En effet, elle est sujette à des risques d'explosion combinatoire lorsque des exemples complexes sont traités, et la vision du système à commander est basée uniquement sur les entrées (capteurs) et les sorties (actionneurs) de la Partie Commande (PC). De plus, la détermination et la compréhension de l'erreur par le concepteur nécessitent de sa part la connaissance d'outils de modélisation tel que les automates à états. Le concepteur s'apparente de fait plus à un expert qu'à un apprenant. Enfin, la solution proposée ne tient pas compte du fonctionnement cyclique de l'Automate Programmable Industriel (API). En effet, l'utilisation d'un API peut entrainer des évolutions simultanées d'événements, des retards de causalité. Devant la difficulté de pouvoir implémenter les résultats issus d'une approche de validation de la commande hors ligne basée sur la synthèse de la commande, nous avons orienté nos travaux vers une approche en ligne par filtre pour valider la commande.

Le chapitre 1 a montré que l'approche proposée doit en tout premier lieu assurer la sécurité du système en interdisant les évolutions dangereuses. Ce point soulève la

problématique de comment définir les situations dangereuses, et comment éviter les détériorations qu'elles entraînent. Le chapitre 1, au travers des concepts des Systèmes Homme-Machine (SHM) a aussi mis en évidence la nécessité de prendre en compte l'automaticien (concepteur, expert) dans la proposition d'une démarche de validation de la commande. Cette approche de l'automatisation a conduit à deux idées :

- Adapter la vision du système à la compétence du concepteur,
- Donner aux outils de validation des capacités explicatives en permettant la remontée d'informations utiles pour le concepteur.

Nous proposons dans ce chapitre une approche de validation en ligne de commande avec prise en compte des connaissances du concepteur. Le chapitre est structuré en trois parties.

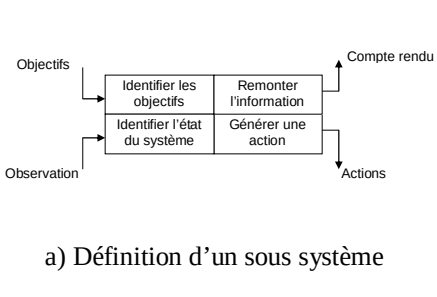
La première partie propose une approche permettant d'adapter une spécification (i.e. un problème de commande) aux compétences du concepteur, tout en gardant une vision globale de la Partie Opérative (PO). La seconde partie présente l'approche de validation par filtre intégrant la prise en compte du concepteur, qui a été développée. Le troisième point traite des difficultés à correctement définir les composants du filtre. En effet, toutes les informations nécessaires dans le filtre ne sont pas mesurées, il faut donc être capable de les reconstruire. Les contraintes peuvent représenter les spécifications liées à la sécurité et au cahier des charges. Pour ce dernier point, il est proposé à l'expert de suivre un canevas de contraintes après avoir découpé le système en éléments de PO (ElPO) et défini les interactions entre ElPO et produit. L'approche proposée est illustrée au moyen de l'exemple du tri de caisses présenté dans le chapitre 2.

2 Adaptation fonctionnelle de la PO au concepteur

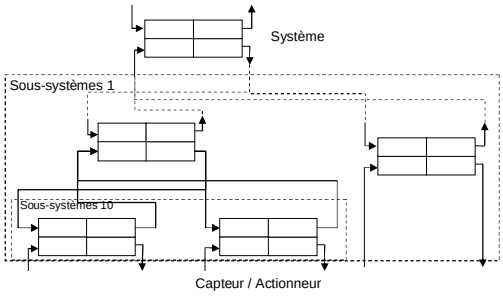
L'idée proposée dans le chapitre 1 est, nous l'avons vu, d'adapter le niveau de difficulté par une modification des spécifications au niveau « fonctionnel » de l'axe « Moyens-But ». Donc en modifiant le degré d'automatisation, il devient possible de conserver une vision globale du système. Pour cela, nous proposons d'adapter le niveau de difficulté en utilisant la dimension fonctionnelle du système et l'autonomie donnée au concepteur. Ainsi, il devient possible de limiter la perception de la PO et de donner plus ou moins de possibilités d'actions

à l'automaticien. De cette manière, l'automaticien va concevoir sa commande en utilisant des entrées/sorties comme si elles correspondaient aux entrées/sorties du système.

Les travaux de (Belhimeur, 1989) portant sur une description hiérarchique d'un système, proposent de définir des sous-systèmes (figure 3.1. b). Chaque sous-système reçoit du niveau supérieur des objectifs à réaliser. Le niveau supérieur attend en retour un compte rendu que le sous-système aura défini en fonction des observations provenant des niveaux inférieurs. Pour réaliser les objectifs, le sous-système envoie des actions aux niveaux inférieurs. Cette perception du système peut être reprise pour donner une « nouvelle » vision du système, adaptée au concepteur. C'est-à-dire que pour un niveau hiérarchique donné, les sous-modèles correspondraient au champ d'actions du concepteur. Cette idée est reprise dans la représentation de la fonction (figure 3.2.) mise à la disposition de l'utilisateur (Marange et al, 2007 b, 2007 f). Ainsi, la définition de la fonction va permettre d'encapsuler une séquence plus ou moins complexe, des synchronisations et/ou des structures difficiles.



a) Définition d'un sous système



b) Décomposition d'un système vers les capteurs/actionneurs

Figure 3.1. Définition hiérarchique d'un système par Belhimeur

Une fonction (figure 3.2.) est activée par le biais d'une demande d'activation (D_{acti}) de la part du concepteur et elle est désactivée lorsqu'une demande de désactivation ($D_{désacti}$) est faite. L'activation réelle de la fonction est effectuée seulement si les conditions d'activation ($CondFa_i$) sont présentes. Le principe d'exécution d'une fonction est le suivant : L'activation ou la désactivation d'une fonction ne sera effective que si les conditions d'activation ou de désactivation sont présentes. Si le concepteur fait la demande d'activation (D_{acti}) de la fonction dans l'intervalle I_1, où les conditions d'activation sont présentes, l'envoi est correct et la fonction exécute l'ensemble des « sorties système » ($\sum(S)$). Par contre, si la demande est faite à l'extérieur de I_1, une alarme ds_i est émise. Il en est de même pour l'alarme fs_i qui est émise, si la demande de désactivation ($D_{désacti}$) n'est pas faite dans l'intervalle I_2. Une variable $Fonc_i\_en\_exécu$ permet de savoir si la fonction est en cours d'exécution ou non. Il est à noter que la définition de la commande au niveau des sorties-systèmes est à la charge de l'expert.

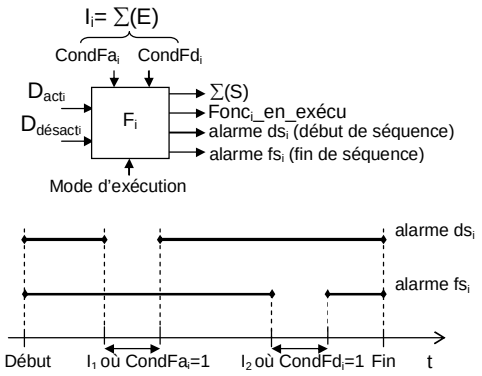


Figure 3.2. Définition des paramètres d'une fonction

À partir des connaissances du concepteur, la fonction peut être exécutée selon deux modes pour lui donner plus ou moins d'autonomie : mode semi-automatique, ou mode contrôlé :

- Dans le premier cas : mode semi-automatique, l'activation de la fonction est contrôlée par le concepteur et la désactivation est faite automatiquement par le système lorsque les conditions de désactivation deviennent vraies. Dans ce mode de fonctionnement, seules des erreurs au niveau de l'activation seront détectées.
- Enfin, dans le deuxième cas : en mode contrôlé, c'est le concepteur qui doit activer et désactiver la fonction lorsque les conditions sont remplies. Dans ce mode de fonctionnement, des erreurs peuvent être commises au niveau de l'activation et/ou de l'arrêt.

Pour ces deux cas, des alarmes (*dsi*, *fsi*) sont déclenchées si la demande ne coïncide pas avec l'instant où les conditions sont vraies. Dans le premier cas, il y aura seulement des alarmes sur une mauvaise activation des fonctions (*dsi*).

Une fonction ne peut pas être réactivée lorsqu'elle est en cours de fonctionnement. Pour cela, la variable $Fonc_i\_en\_exécu$ doit être à 0 pour activer la fonction. Il faut donc respecter la condition suivante :

$$\uparrow F_i \wedge Fonc_i\_en\_exécu = 0 \qquad (3.1)$$

Par cette approche, il devient possible de garder une vision globale du système adaptée au niveau de connaissance de l'utilisateur. Cela n'empêche pas le concepteur de commettre des erreurs pouvant mettre en danger la PO. La suite du chapitre propose une approche permettant de sécuriser le système et d'apporter une explication en cas d'erreur.

3 Approche de validation en ligne par filtre

Plusieurs comportements du système peuvent être distingués (figure 3.3.). Soit le comportement peut mettre le système en danger, soit il est sécurisé mais ne répond pas au cahier des charges, soit il assure la sécurité du système et répond aux attentes du cahier des charges.

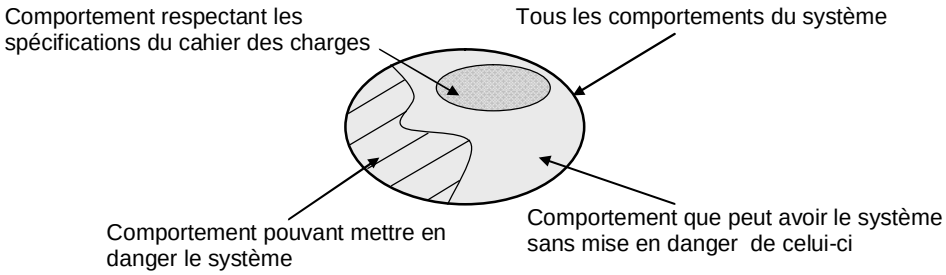


Figure 3.3. Différents comportement du système

Pour ne pas atteindre la zone où le comportement est dangereux pour le système, des contraintes de sécurité sont définies. Celles-ci définissent ce que ne doit pas faire le système. Une fois, le système sécurisé, les contraintes de vivacité définissent les attentes du cahier des charges. Ce second type de contraintes n'est pas traité dans ce manuscrit. Les travaux de thèse ont seulement recherché à sécuriser le système quelle que soit la commande proposée par le concepteur. La validation des spécifications du cahier des charges ne fait pas l'objet de ces travaux de thèse et fera partie de nos perspectives de travail.

Pour que l'approche de validation soit techniquement réalisable, une approche « en ligne » par filtre (Cruette, 1991) implémentée directement dans l'API est proposée. Cela permet de s'affranchir des problèmes d'asynchronisme et de ne pas calculer toutes les évolutions possibles. Dans cette approche de validation, l'idée est d'empêcher les évolutions pouvant mener le système à une situation de risques sans pour autant calculer toutes les commandes possibles. La première partie présente le modèle de filtre utilisé. La seconde partie présente la méthodologie pour mettre en place une validation de commande en ligne.

3.1 Proposition d'un modèle

L'utilisation d'un filtre pour effectuer de la surveillance de système n'est pas une idée nouvelle. En effet, le chapitre 1 §6 (Combacau, 1991 ; Cruette, 1994 ; Lhoste, 1994 ;

Zamaï, 1997) a montré que plusieurs travaux s'étaient intéressés à l'utilisation de filtres. La difficulté de ces approches réside dans la définition des spécifications et leurs implémentations. Dans le cadre de ce manuscrit, une approche est proposée pour accompagner l'expert dans la définition de ces contraintes. Il est proposé une décomposition du filtre au niveau du système, mais aussi au niveau des fonctions définies pour le concepteur dans le paragraphe 2.

Pour interdire toutes les évolutions dangereuses pour le système, un filtre est placé entre la PO et la PC (figure 3.4.a). Celui-ci ne filtre que les évolutions provenant de la PC, car il est supposé qu'il n'y a aucune défaillance du système au niveau matériel. Ce filtre est composé de plusieurs composants permettant d'une part de filtrer mais aussi d'adapter le système à l'utilisateur (figure 3.4.b) (Marangé et al., 2008a, 2007 a ; 2007 f) :

- Filtre de validation système : ce premier filtre assure la sécurité au niveau du système (capteurs/actionneurs). Les spécifications qu'il contient, sont définies par les entrées/sorties du système,
- Filtre de validation fonctionnel : ce second filtre permet d'assurer le respect du cahier des charges en validant le bon séquencement des fonctions, ainsi que les conditions d'exécution des fonctions par rapport aux conditions d'activation et de désactivation,
- Exécuteur de fonctions : lorsqu'une fonction est envoyée vers la PO et validée par le filtre de validation fonctionnel, le bloc « exécuteur de fonctions », exécute le morceau de programme implanté par l'expert,
- Adaptateur d'informations : à partir des informations capteurs/actionneurs, ce bloc adapte les informations au niveau de l'utilisateur, c'est-à-dire qu'il reconstruit des informations compréhensibles par l'utilisateur à partir des informations capteurs/actionneurs.

Le filtre de validation système est constitué de la manière suivante (figure 3.4.c) :

- Un premier bloc qui reconstruit les informations nécessaires si les informations fournies par les capteurs ne sont pas suffisantes,
- Un deuxième bloc applique l'ensemble des contraintes,
- Le troisième bloc met à jour les sorties si les spécifications sont respectées.

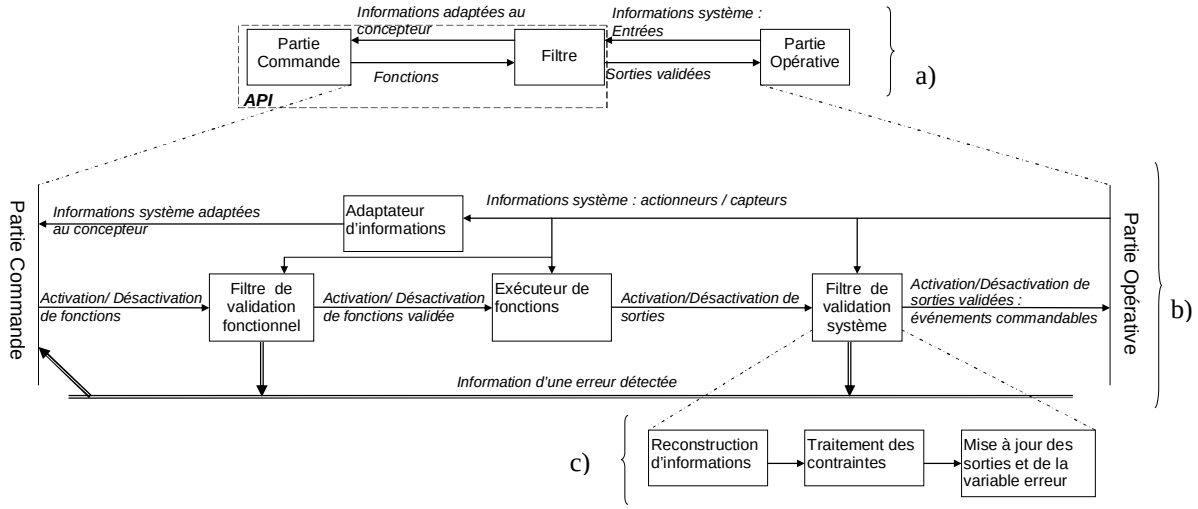


Figure 3.4. Approche par filtre a) idée générale, b) Différents blocs constituant le filtre, c) détails du filtre de validation système

Pour mettre en place cette approche de validation en ligne par filtre, l'expert doit faire un travail préalable d'analyse fonctionnelle et dysfonctionnelle du système pour définir chacun des blocs. Le paragraphe suivant, présente le travail fait par l'expert pour définir le champ d'actions du concepteur, sécuriser le système et remonter l'information sur l'état du système.

3.2 Le travail de l'expert

À partir de ses connaissances sur le système, et sur les compétences des concepteurs, l'expert doit mettre en place l'approche de validation de la commande. La figure 3.5. indique les différentes tâches qu'il doit effectuer (Marangé et al., 2008 a, 2007 a, 2007 f).

3.2.1 Définition du système adaptée au concepteur

Pour que le système soit compréhensible par le concepteur, l'expert doit définir le champ d'actions de celui-ci, c'est-à-dire l'ensemble de fonctions ($F_{concepteur}$) qu'il aura à sa disposition pour commander le système. Pour cela, il va se baser sur sa connaissance du système, ainsi que sur les compétences et les capacités du concepteur et ainsi définir les fonctions et les informations à remonter ($I_{concepteur}$).

Une fois, les fonctions et les informations du concepteur définies, l'expert doit implémenter les morceaux de programme représentant chaque fonction, dans le bloc Exécuteur. En parallèle, il doit aussi définir la reconstruction des informations à remonter au

concepteur ($I_{concepteur}$) dans le bloc Adaptateur, à partir des entrées/sorties du système. La partie 4.2 présente la manière de reconstruire les informations lorsque que l'information capteur n'est pas suffisante, lorsqu'une information temporelle ou produit est nécessaire. Par l'intermédiaire de ces nouvelles informations et fonctions, l'expert définit un cahier des charges compréhensible par le concepteur.

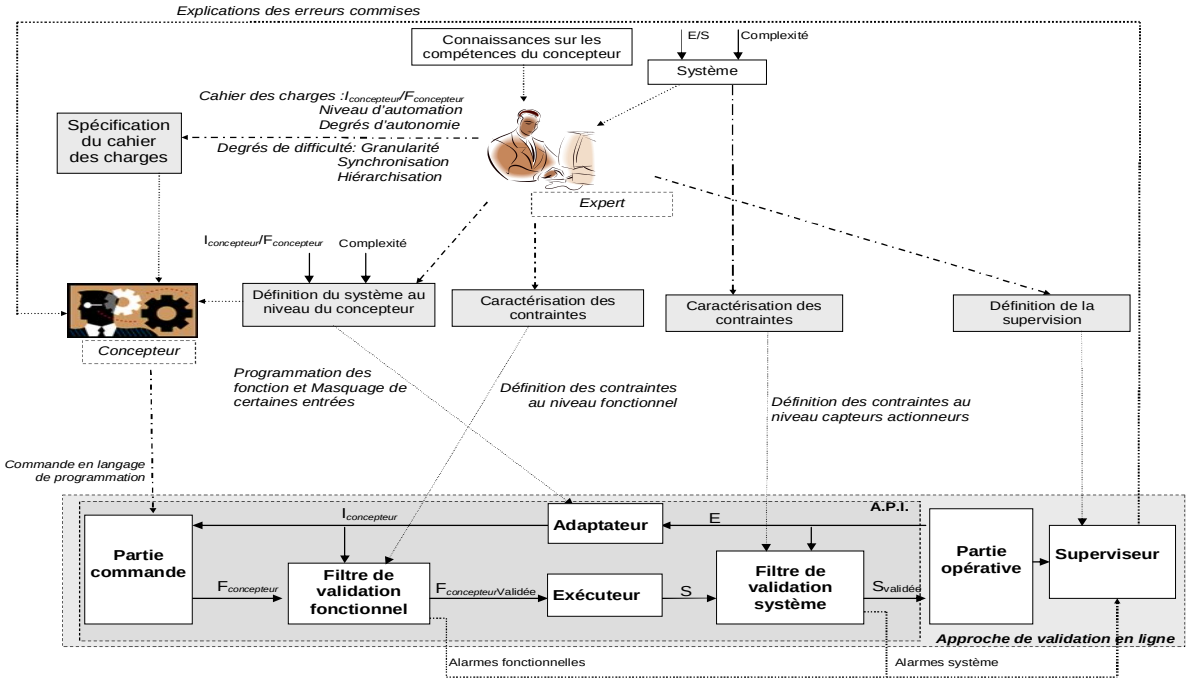


Figure 3.5. Définition du travail de l'expert

3.2.2 Définition des spécifications du filtre

La définition des spécifications par l'expert va permettre d'une part d'assurer la sécurité du système et d'autre part de valider le respect du cahier des charges. La sécurisation du système se fait dans le filtre de validation système et le respect du cahier des charges au niveau du filtre de validation fonctionnel. Seul le premier filtre a fait l'objet de recherche pendant cette thèse.

L'expert assure la sécurité du système en définissant des contraintes à partir des informations système. La définition au niveau du système va permettre de définir le maximum de sécurité et ne sera réalisé qu'une seule fois pour le système. En effet, les spécifications placées dans le filtre de validation système sont indépendantes du champ d'actions du concepteur, du cahier des charges et donc elles doivent être vraies tout le temps.

L'expert assure le respect du cahier des charges au niveau du filtre de validation fonctionnel. La définition des spécifications au niveau du filtre de validation fonctionnelle porte sur l'activation ou la désactivation des fonctions, lorsque les conditions d'activations sont vraies, sur l'interdiction de réactiver des fonctions si elles sont en cours d'exécution et sur le respect des spécifications du cahier des charges. Le travail présenté dans ce manuscrit a porté sur la sécurisation du système et non sur le respect du cahier des charges. Cela fera l'objet de perspectives à ces travaux de recherches.

3.2.3 Définition d'une interface de dialogue

Au niveau des échanges d'informations avec le concepteur, une interface de supervision peut être définie par l'expert. L'interface de supervision peut indiquer les fonctions en cours d'exécution, les conditions qui ne sont pas respectées lors d'une activation ou d'une désactivation de fonction… Ce point n'est pas plus détaillé car il ne fait pas l'objet des travaux effectués dans cette thèse.

Le travail présenté dans cette thèse porte sur la sécurisation du système face aux erreurs de conception de commande. La sécurité du système est assurée par le filtre de validation système qui contient un ensemble de contraintes pour ne pas atteindre de situations dangereuses. Le paragraphe suivant présente la mise en place du filtre de validation système.

4 Conception du filtre de validation système

La sécurité du système est assurée en plaçant des contraintes définies au niveau capteurs/actionneurs, dans le filtre de validation système. La définition des contraintes peut nécessiter de reconstruire de l'information lorsque les états des capteurs ne sont pas suffisants pour connaître précisément la position du système. La reconstruction d'information fait l'objet de la première partie. La deuxième partie propose une démarche pour définir les contraintes de sécurité à mettre en place. Avant de présenter la reconstruction d'information et l'approche, certaines définitions et notations vont être rappelées.

4.1 Préliminaire : définitions et notations

Pour reconstruire les informations et définir les contraintes, les termes états, événements, commandables, non commandables, mesurées, non mesurées vont être utilisés.

4.1.1 États / événements

Pour exprimer les contraintes ou reconstruire des informations, nous avons besoin de connaître la valeur d'une variable et si elle a subi un changement. La valeur statique d'une variable de la PO définit l'état qui est caractérisé par une variable booléenne (0 ou 1), noté *X*. L'état peut être caractéristique d'un actionneur, d'un capteur, d'un produit ou d'une information reconstruite. Cette notion est donc plus générale que celle définie dans les automates booléens du chapitre 2, où les états correspondaient uniquement aux différentes valeurs du vecteur d'entrées/sorties. L'état va donc ainsi pouvoir correspondre aussi bien à des positions physiques des actionneurs que des transformations subies par les produits (déplacés, assemblés, stockés, …).

Les changements d'une variable d'état sont appelés événements. Cela représente le passage de 0 à 1 (activation) ou de 1 à 0 (désactivation) d'une variable d'état, noté $\uparrow E$ pour l'activation $\downarrow E$ pour la désactivation. Comme pour les états, l'événement peut représenter un changement d'un ordre sur un actionneur, un capteur, un état produit ou une information reconstruite.

En fonction de ce que caractérise l'état ou l'événement, il sera dit commandable ou non commandable.

4.1.2 Commandables / non commandables

Pour la définition des contraintes, il est nécessaire de considérer la commandabilité de l'événement et de l'état. Pour cela, comme dans le chapitre 2, l'interprétation de Balémi (Balémi, 1993) est utilisée. Un événement est dit commandable lorsqu'il représente une activation $\uparrow Ec$ ou une désactivation $\downarrow Ec$ d'une sortie (actionneur) de la commande. En revanche, il est dit non commandable, lorsqu'il représente une activation $\uparrow Enc$ ou une désactivation $\downarrow Enc$ d'une entrée (capteur, information reconstruite, produit).

La même interprétation est utilisée pour l'état lorsqu'il fait référence à un actionneur, l'état sera dit commandable *Xc*. Par contre, lorsqu'il fait référence à un capteur, un produit ou une information reconstruite, l'état sera dit non commandable *Xuc*. Bien qu'un état en toute

rigueur ne puisse pas être dit commandé ou non, nous faisons un abus de langage pour spécifier qu'il est le résultat d'un événement commandable ou non commandable. Cette interprétation est utilisée pour permettre de différencier les expressions de contraintes. Dans l'expression des contraintes ou des informations, le terme état est utilisé d'une manière générale pour représenter un ensemble d'états d'actionneurs ou de capteurs, noté f(*Xc*), f(*Xuc*).

4.1.3 Information mesurée / reconstruite

Les variables utilisées dans l'expression des contraintes ne font pas toujours référence à des variables directement lues, c'est-à-dire mesurées par un capteur. Dans le cas où l'événement ou l'état est identifié directement par la valeur d'un capteur, la variable sera dite mesurée. Par contre dans le cas contraire, lorsqu'il faudra la déterminer par la combinaison d'événements ou/et d'états d'actionneurs, de capteurs, de produits, ou d'autres informations reconstruites, la variable sera dite reconstruite.

Ces termes vont être utilisés au travers des méthodes pour reconstruire l'information ou pour établir le canevas de contraintes. Il peut être nécessaire de reconstruire l'information lorsque l'information apportée par les capteurs présents n'est pas suffisante.

4.2 Besoin de reconstruire l'information

En fonction de l'instrumentation utilisée dans le système, la connaissance sur celui-ci est plus ou moins importante et nécessite de reconstruire ou non des informations. Trois cas sont distingués :

- Plusieurs positions d'un ElPO ne sont pas distinguables par l'information capteur,
- Une information temporelle est nécessaire pour valider certaines contraintes,
- Une information sur l'état du produit peut influer sur l'évolution du système.

4.2.1 Outil utilisé

Pour reconstruire l'information, l'outil utilisé est l'automate communicant. Les automates communicants sont définis par $G = (X, \sum, \delta, x_0, X_m)$ où :

- X : un ensemble fini d'états de l'automate G
- $\sum$: un ensemble fini d'événements

- δ : un ensemble de fonctions de transition définies par une garde (G) qui permet d'exprimer une condition, et par une synchronisation (S) qui permet de synchroniser plusieurs automates par échange de messages. Les échanges sont notés de la manière suivante : pour l'émission *mess !* et pour la réception *mess ?*, et pour la mise à jour (M) : permet de mettre à zéro certaines horloges, ou variables entières.
- x_0 : un état initial de l'automate
- X_m : un ensemble d'états marqués

Au niveau de la reconstruction d'information, la notion de synchronisation n'est pas utilisée.

4.2.2 Positions d'un élément de PO

En fonction des capteurs constituant le système, les informations sont directement mesurées ou nécessitent d'être reconstruites. Deux cas sont à distinguer. Soit la même information indique des positions différentes, soit le nombre de capteurs est insuffisant.

Dans le chapitre 2, le système est décrit par des états booléens définis par un vecteur d'entrées/sorties uniquement. Dans le cas de la figure 3.6., l'information provenant de la valeur des entrées est insuffisante pour distinguer la position du chariot lorsqu'il quitte le capteur intermédiaire. Sous l'hypothèse de non inertie, l'information peut être reconstruite en combinant l'information capteur avec l'information actionneur. Pour l'exemple d'un chariot à 3 positions (*a*, *b*, *c*) (figure 3.6.*)*, le front descendant sur la variable *b* ne permet pas de savoir si le chariot se trouve entre *a* et *b* ou entre *b* et *c*. Si une modélisation booléenne classique est utilisée, il y a un problème d'indéterminisme au niveau du $\downarrow b$. Par contre, si une combinaison est faite entre l'état de l'ordre *Avancer* et l'événement $\downarrow b$, cela indique que le chariot est entre *b* et *c*. Donc, l'information reconstruite sera i_1= Avancer $\wedge \downarrow$b, de même pour i_2 = Reculer $\wedge \downarrow$b. Ces informations reconstruites nous permettent de savoir que le chariot est entre *a* et *b* ou entre b et *c*.

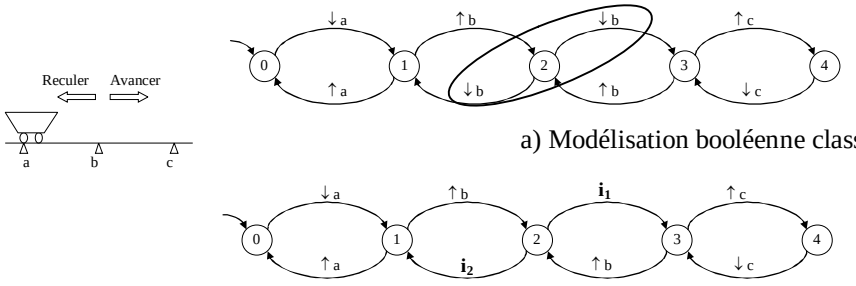


Figure 3.6. Reconstruction de l'information dans le cas de deux situations décrites par un même événement

La reconstruction se fait au moyen d'automates communicants qui permettent par l'intermédiaire d'une garde de définir des conditions de franchissement lorsqu'un événement apparaît.

Dans le second cas, il n'y a pas de capteurs pour connaître l'état du système, il faut donc ajouter une information temporelle combinée à l'ordre pour savoir où se trouve l'ElPO. Dans l'exemple d'un vérin sans capteur de fin de course (figure 3.7.), après un laps de temps t_1, suite à l'envoi de l'ordre *Avancer*, le vérin est supposé avoir quitté sa position rentrée et après un laps de temps t_2, le vérin est supposé être arrivé en fin de course.

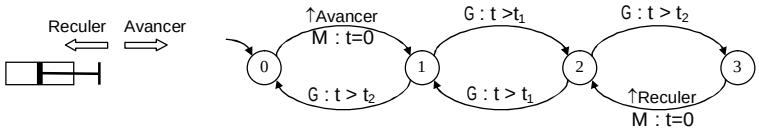


Figure 3.7. Reconstruction de l'information dans le cas où il n'y a pas de capteurs

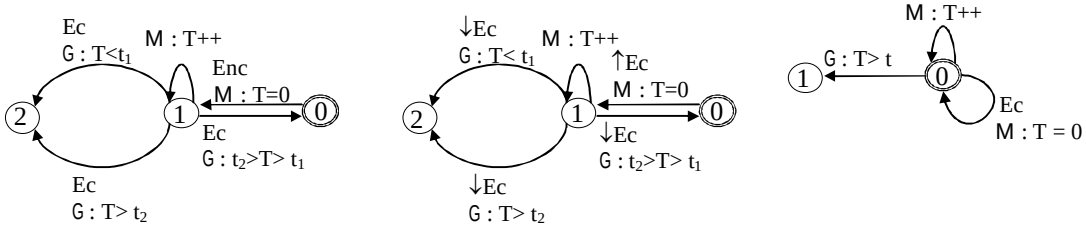
Ces deux méthodes permettent d'avoir des informations supplémentaires sur le système. L'autre information qui peut être utile, est l'information temporelle qui peut être utilisée simplement car l'approche de validation est en ligne.

4.2.3 Information temporelle

L'approche proposée de validation de la commande s'effectue en ligne ce qui permet d'ajouter des contraintes temporelles sans complexifier le modèle de PO. Les contraintes temporelles représentent soit une spécification du cahier des charges, c'est-à-dire que le système doit se trouver dans un certain état pendant un temps défini, soit un blocage c'est-à-dire qu'il n'y a pas d'évolution ni au niveau de la commande, ni au niveau de la PO, soit une

évolution trop tôt ou trop tard de la PO ou de la commande. Ce dernier type d'information est utile pour détecter une défaillance au niveau du système. Pour prendre en compte ces différentes contraintes, les modèles de la figure 3.8. sont proposés ;

- L'envoi d'un ordre trop tôt (t_1) ou trop tard (t_2) par rapport à la réception d'une information capteur ou reconstruite. Pour cela, l'automate de la figure 3.8.a est défini. A partir de l'état initial, lors de la réception de l'événement non commandable Enc, une temporisation *T* est mise à 0 (M : *T*=0) et est incrémentée dans l'état 1, si l'évolution de l'ordre Ec a lieu avant t_1 (G : T<t_1) ou après t_2 (G : T>t_2), l'automate passe dans l'état 2. Par contre si la commande évolue pendant l'intervalle de temps (G : t_1<T<t_2), l'automate revient dans l'état 0.
- La désactivation d'un ordre trop tôt (t_1) ou trop tard (t_2) permettant ainsi de représenter des spécifications du cahier des charges. Par l'intermédiaire de l'automate de la figure 3.8.b, cette spécification est représentée. A partir de l'état initial, suite à l'occurrence de ↑Ec, la temporisation *T* est mise à 0, l'automate passe dans l'état 1 où la temporisation est incrémentée. Si l'ordre est désactivé (↓Ec) trop tôt (G : T<t_1) ou trop tard (G : T>t_2), l'automate passe dans l'état 2 ; sinon, il revient dans l'état 0, lorsque que la désactivation (↓Ec) a lieu pendant l'intervalle (G : t_1<T<t_2).



a) Envoi d'une commande b) Désactivation d'une commande c) blocage

Figure 3.8. Modélisation des contraintes temporelles

- Le blocage d'un système peut être déterminé si pendant un laps de temps (t_1) aucun événement n'a eu lieu. Pour cela, dans l'automate de la figure 3.8.c, à chaque fois qu'il y a une évolution de commande du système (Ec) la temporisation T est mise à 0. Si la temporisation devient supérieure à une valeur limite t, l'automate passe dans l'état 1 représentant que le système n'a subi aucune évolution, c'est-à-dire qu'il est bloqué.

4.2.4 Information sur l'état produit

L'information sur l'état produit doit représenter les opérations subies par le produit ou la position occupée par celui-ci. Les opérations considérées sont de quatre types: transformation, assemblage/désassemblage, déplacement et stockage. Pour définir le nombre d'état du produit, l'expert doit considérer les effets de chacun des ElPO sur le produit et la précision dont il a besoin. En effet, si un produit se déplace sur un tapis entre deux capteurs, il est peut être suffisant de savoir qu'il est au départ, entre les deux, à la fin, ou au contraire, il faut connaître plusieurs positions intermédiaires entre les deux extrémités.

Après avoir étudié les évolutions du produit qui étaient nécessaires pour définir les contraintes, un automate à deux états modélise le produit qui a subi ou pas l'opération. Pour modéliser l'état du produit, une modélisation modulaire a été choisie pour éviter l'explosion combinatoire et permettre de gérer plusieurs produits. Nous avons fait l'hypothèse que chaque état ne peut représenter qu'un seul produit.

Dans le cas d'une transformation, d'un assemblage ou d'un désassemblage, chaque état représente : « Le produit a été transformé ou non ? », « Le produit a été assemblé ou non ? », « Le produit a été désassemblé ou non ? », et le passage d'un état à l'autre se fait sous certaines conditions mesurées ou pouvant être reconstruites.

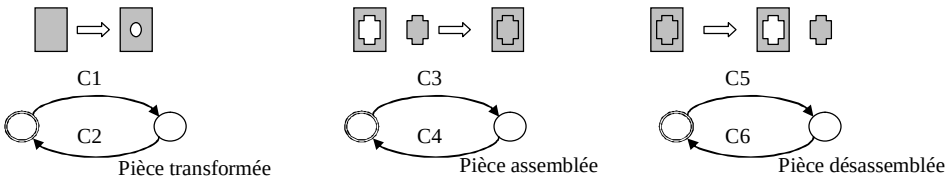


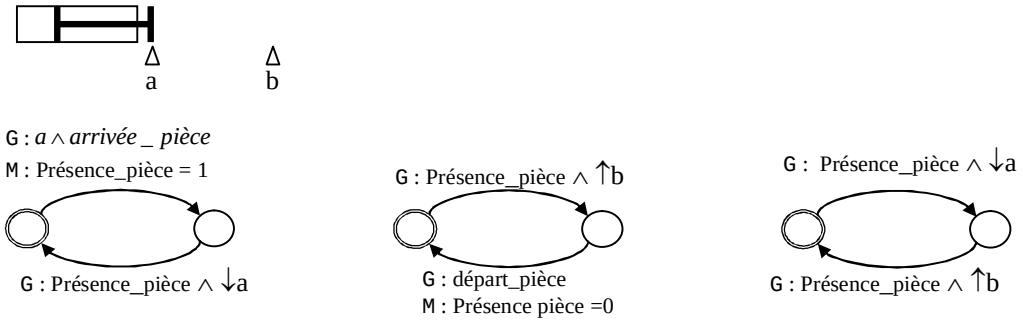
Figure 3.9. État produit pour une transformation, un assemblage et un désassemblage

Dans le cas d'un déplacement, il faut distinguer deux cas. Soit le déplacement du produit est lié directement au mouvement de l'ElPO, soit il doit être maintenu pour être déplacé par l'ElPO. Dans le premier cas (figure 3.10.), il y a 3 états du produit : en a, en b, et entre a et b :

- Une pièce sera présente en *a*, si une pièce arrive et que l'ElPO est en *a* et la pièce quittera *a*, lors de la désactivation de a. (figure 3.10.a).
- Une pièce sera présente en b, quand l'ElPO activera b et si une pièce est présente et que la pièce quittera b, quand la pièce sera retirée (figure 3.10.b).

- La pièce est entre *a* et *b*, dés que *a* est désactivé et ne l'est plus quand *b* est activé (figure 3.10.c).

S'il n'y a pas de pièce en *a* lors du déplacement, il est logique que la pièce n'évolue pas. Pour cela au niveau de l'automate de pièce en a, la garde vérifie que l'ElPO est en *a* et qu'une pièce est arrivée d'un autre ElPO (*arrivée_pièce*). L'information *Présence_pièce* est mise à 1 et sera mise à 0 lorsqu'elle sera retirée de b par une intervention extérieure ou par un autre ElPO.

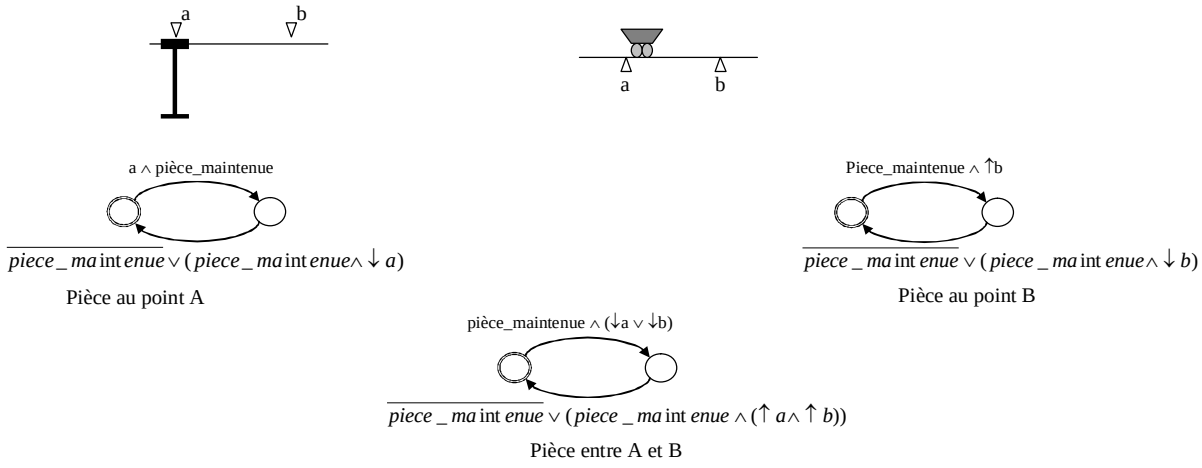


a) Pièce en a b) Pièce en b c) Pièce entre a et b

Figure 3.10. État produit pour un déplacement sans maintien

Dans le second cas (figure 3.11.), il y a 3 états de produit en a, en b, entre a et b, et il faut reconstruire l'information que la pièce est maintenue ou non dans le cas d'un préhenseur, ou la pièce est chargée / déchargée dans le cas du chariot :

- Une pièce sera présente en *a*, si une pièce est maintenue et que l'ElPO est en *a* et la pièce quittera *a*, lors de la désactivation de *a* ou si la pièce n'est plus maintenue (figure 3.11.a).
- Une pièce sera présente en *b*, quand l'ElPO activera *b* et si la pièce est maintenue et la pièce quittera *b*, si l'ElPO quitte *b* en maintenant la pièce ou si la pièce n'est plus maintenue (figure 3.11.c).
- La pièce est entre a et b, dès que *a* ou *b* est désactivée en maintenant la pièce et ne sera plus entre *a* et *b* si le produit arrive en *a* ou *b*, ou si la pièce n'est plus maintenue (figure 3.11.b).



a) Pièce en a b) Pièce en b c) Pièce entre a et b

Figure 3.11. État produit pour un déplacement avec maintien

Cette partie a présenté des modèles simples pour reconstruire l'information lorsque l'instrumentation du système n'est pas suffisante. Cependant, la difficulté réside dans la définition des conditions pour passer d'un état à l'autre et dans la précision que l'expert souhaite donner au modèle.

4.3 Démarche pour la définition des contraintes

L'expert doit suivre la démarche proposée figure 3.12. pour définir les contraintes qui assurent la sécurité du système sans faire d'hypothèses sur la commande proposée. La démarche est constituée des six étapes suivantes :

- Dans la première étape, l'expert fait une analyse structurelle du système jusqu'au niveau capteurs/actionneurs pour définir les ElPO constituant le système, les interactions qui existent entre les ElPO et les interactions avec le produit. Cette première étape de décomposition du système permet de traiter de façon modulaire chaque ElPO, et chaque interaction.

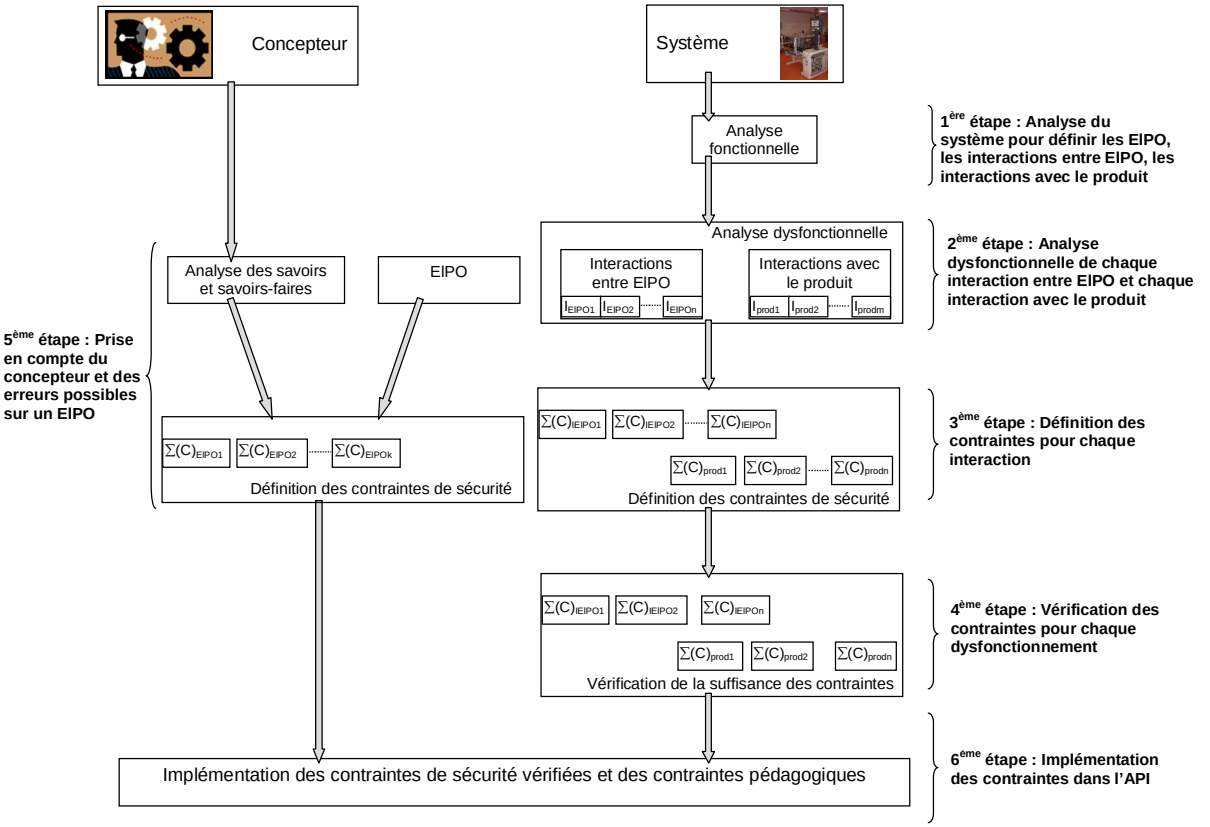


Figure 3.12. Démarche de l'expert pour mettre en place la validation en ligne

- Dans la deuxième étape, l'expert fait une analyse dysfonctionnelle des interactions entre ElPO et avec le produit pour définir les éventuelles détériorations.
- Dans la troisième étape, l'expert considère chaque interaction indépendamment du reste du système et définit les contraintes pour empêcher les détériorations associées.
- Dans la quatrième étape, l'expert vérifie la suffisance des ensembles de contraintes en considérant chaque interaction indépendamment (développé dans le chapitre 4).
- Dans la cinquième étape, l'expert prend en compte les savoirs et les savoir-faire du concepteur pour définir des contraintes supplémentaires sur les ElPO pris indépendamment les uns des autres. Ces contraintes, notées contraintes pédagogiques, ne nécessitent pas d'être vérifiées car elles n'influencent pas la sécurité du système.
- Dans la sixième et dernière étape, l'expert implémente les contraintes dans l'API en langage Ladder diagramme.

4.3.1 Analyse du système

La définition des contraintes est plus difficile si la vision du système reste globale. La première tâche de l'expert est de découper le système en ElPO (figure 3.13.a), puis de définir les interactions entre eux (figure 3.13.b), et les interactions avec le produit (figure 3.13.c).

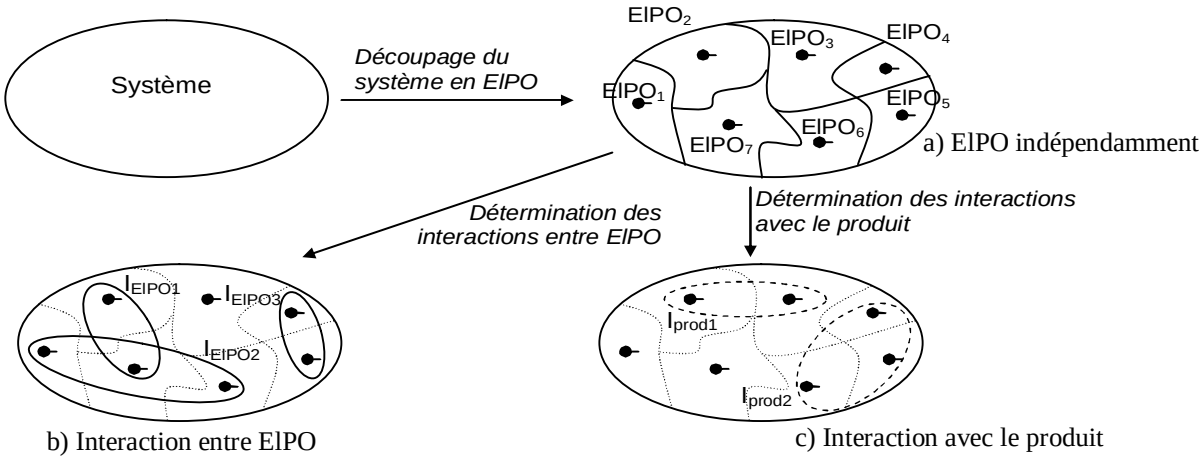


Figure 3.13. Étapes d'analyse du système

Une fois le système découpé en ElPO par l'analyse structurelle, l'expert définit les interactions. Les interactions entre les ElPO peuvent être dues à un lien physique entre les ElPO ou à un partage de zones communes entre les ElPO. Les ElPO ayant un lien physique (par exemple un préhenseur formé de deux vérins) peuvent être considérés comme un seul en interaction avec les autres ElPO.

Les interactions peuvent aussi venir de la présence du produit dans le système, l'expert doit donc considérer les positions prises par le produit. En effet, un état produit peut engendrer une évolution particulière ou occuper une position interdite, ou empêcher certaines évolutions du système.

Les interactions entre les ElPO se font à l'intérieur d'une zone commune d'évolution qu'ils partagent, et où il peut y avoir collision entre eux.

Pour chaque interaction, l'expert va définir l'ensemble des combinaisons de positions prises par les ElPO mis en jeu dans une interaction et définir les combinaisons qu'il ne veut pas atteindre. Dans le cas où des ElPO ont un lien physique et qu'ils sont en interaction avec d'autres ElPO, l'expert utilise les positions prises par la combinaison des ElPO, pour définir les états interdits pour la nouvelle interaction.

Pour ne pas atteindre les positions interdites, l'expert doit définir un ensemble de contraintes. Pour cela, il considère chaque interaction indépendamment et définit les contraintes nécessaires.

4.3.2 Canevas de contraintes pour assurer la sécurité

Pour un système donné, des positions sont dites interdites, et ne doivent pas être atteintes. Pour cela, au niveau du filtre de validation système, un ensemble de contraintes est défini pour empêcher d'atteindre l'ensemble des positions interdites. En effet, la définition d'une contrainte ne permet plus d'atteindre certaines positions. Il faut donc définir assez de contraintes pour que toutes les positions interdites ne soient plus atteignables (figure 3.14.).

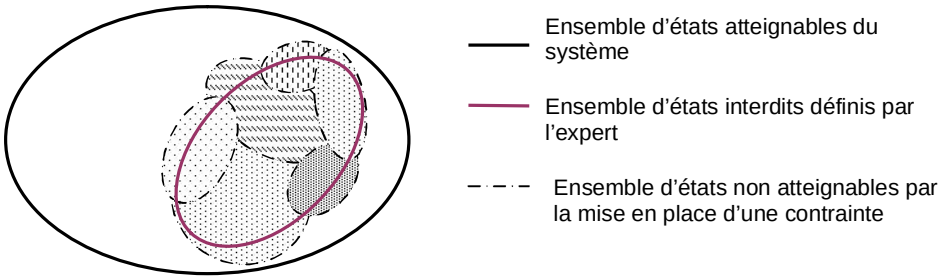


Figure 3.14. Ensembles de contraintes

L'ensemble de contrainte peut conduire à un fonctionnement du système plus restrictif que celui définit par les états interdits, mais ceci est obligatoire pour tenir compte de la technologie utilisée dans le système et ne pas faire d'hypothèse sur la commande à valider.

Les contraintes à mettre en place pour ne pas atteindre les situations interdites, sont dites contraintes de sécurité représentant ce que ne doit pas faire le système. Elles peuvent être, selon notre point de vue de deux types : contraintes de sécurité statique ou de sécurité dynamique (Marangé et al., 2007 a). Nous allons, dans la suite de ce chapitre, proposer un canevas d'écriture des contraintes. Ce canevas est utilisé aussi pour définir les contraintes pédagogiques qui ne mettent pas obligatoirement en danger le système. Elles sont définies sur un ElPO pris indépendamment et expriment les envois de commande qui n'ont pas d'effets sur le système.

L'utilisation de ce canevas ne permet pas de garantir que l'ensemble de contraintes relatif à un SAP quel qu'il soit, est suffisant. En vue de remédier à ce problème et de garantir la suffisance des contraintes, une approche de vérification formelle sera proposée dans le chapitre 4.

Chaque définition de contraintes est repérée par des initiales pour permettre d'y faire référence lors de l'utilisation dans les exemples.

4.3.2.1 Contraintes de sécurité statiques

Les contraintes de sécurité statiques sont définies pour représenter une impossibilité physique entre des ElPO ou technique d'un ElPO, c'est-à-dire que la commande ne peut pas avoir deux états commandables à 1 au même instant. Cette contrainte est représentée par l'équation logique :

$$Xc_i \wedge Xc_j = 0 \ (\mathrm{CSS_i}) \tag{3.2}$$

Dans le cas d'un ElPO, la contrainte de sécurité statique représente une impossibilité technique, par exemple : l'activation de deux commandes inverses sur un même ElPO.

Dans le cas de deux ElPO, la contrainte de sécurité statique représente l'envoi en même temps de deux ordres conduisant vers la zone commune. Dans le cas de l'interaction entre ElPO, la contrainte de sécurité statique peut être vérifiée que pour certaines positions de la PO ($f(Xuc_k)$), l'équation logique se généralise donc par :

$$Xc_i \wedge Xc_j \wedge f(Xuc_k) = 0 \ (\mathrm{CSS_{ii}}) \tag{3.3}$$

4.3.2.2 Contraintes de sécurité dynamiques :

Les contraintes de sécurité dynamiques sont définies pour représenter l'interdiction d'un événement en fonction d'un ensemble d'états du système. L'occurrence d'un événement peut dépendre de conditions d'activation ($Conda_i$) ou de désactivation ($Condd_i$). Lorsqu'un ElPO est considéré indépendamment du reste du système, les conditions sont définies en fonction de son propre état. Lorsqu'il y a interaction entre des ElPO ou le produit, les conditions sont liées à l'état des différents ElPO mis en jeu. Lors de l'occurrence d'un événement, trois contraintes peuvent prises en considération :

- L'envoi d'un événement commandable peut dépendre de conditions sur le système, c'est-à-dire l'activation d'un ordre dépend de conditions d'activation et il en est de même pour la désactivation d'un ordre qui n'a lieu seulement lorsque les conditions de désactivation sont présentes.
- La demande d'activation d'un ordre est sans effet, ou peut être dangereuse si les conditions de désactivation sont déjà présentes
- l'occurrence des conditions de désactivation par rapport à un état commandable

Les premières permettent d'activer ou de désactiver un ordre seulement si les conditions requises sont présentes. Par rapport aux conditions d'activation et de désactivation définies par un ensemble d'états non commandables, un événement commandable pourra avoir ou non lieu, ce qui est traduit d'une manière générale par l'équation suivante : $Ec_i \wedge \neg f(Xuc_j) = 0$. Le respect des conditions d'activation, lors de l'activation de l'événement commandable, s'écrit :

$$\uparrow Ec_i \wedge \neg Conda_j = 0 \ (\text{CSD}_\text{i}) \tag{3.4}$$

Il en est de même pour le respect des conditions de désactivation :

$$\downarrow Ec_i \wedge \neg Condd_j = 0 \ (\text{CSD}_\text{ii}) \tag{3.5}$$

Les secondes, dans le cas d'un ElPO pris indépendamment, ne permettent pas d'activer un ordre si les conditions de désactivation sont déjà présentes. En d'autres termes, pour un ensemble d'états non commandables représentant la condition de désactivation, un événement commandable est interdit : $\uparrow Ec_i \wedge f(Xuc_j) = 0$, ce qui peut aussi s'écrire pour les conditions de désactivations :

$$\uparrow Ec_i \wedge Condd_j = 0 (\text{CSD}_\text{iii}) \tag{3.6}$$

Les troisièmes obligent l'arrêt d'un ordre en fonction de l'occurrence des conditions de désactivation, c'est-à-dire que pour l'activation d'une combinaison d'états non commandables, un état commandable est interdit : $Xc_i \wedge \uparrow f(Xuc_j) = 0$, ce qui peut aussi s'écrire :

$$Xc_i \wedge \uparrow Condd_j = 0 (\text{CSD}_\text{iv}) \tag{3.7}$$

Les contraintes sont vérifiées après l'exécution du programme du concepteur avant l'écriture des sorties, ainsi la contrainte CSD_iv peut être utilisée. En effet, en début de cycle d'API, lors de la lecture, cette contrainte n'est pas respectée, elle devra l'être une fois le programme du concepteur exécuté.

Ce canevas de contraintes a deux intérêts. Le premier intérêt est d'accompagner l'expert dans sa définition de contraintes pour empêcher les dysfonctionnements. Le second intérêt est d'apporter une explication en cas de non respect. En effet, les explications suivantes peuvent être associées à chaque contrainte :

- Pour la contrainte CSS_i : Impossibilité technique pour un même ElPO de recevoir

deux ordres inverses

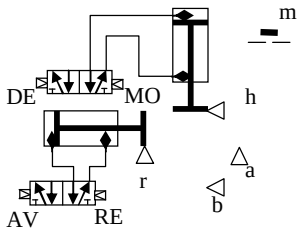
- Pour la contrainte CSS_{ii} : Interdiction d'envoyer en même temps les deux ordres
- Pour la contrainte CSD_i : Interdiction d'envoyer l'activation de l'ordre si les conditions requises ne sont pas présentes
- Pour la contrainte CSD_{ii} : Interdiction d'envoyer la désactivation de l'ordre si les conditions requises ne sont pas présentes
- Pour la contrainte CSD_{iii} : Ordres sans effet sur le système car les conditions de désactivation sont déjà présentes
- Pour la contrainte CSD_{iv} : Obligation de désactiver l'ordre car les conditions de désactivation sont devenues vraies.

La définition de ces contraintes pour une interaction ou pour un ElPO n'a pas le même rôle. Dans le premier cas, elles ont en premier lieu un rôle de sécurisation du système puis d'explication. Par contre dans le second cas, le non respect des contraintes sur un ElPO n'entraîne pas de détérioration, mais ces contraintes sont nécessaires pour prendre en compte d'éventuelles erreurs du concepteur et ainsi y apporter une explication.

Il peut être remarqué que les contraintes CSD_{ii} et CSD_{iii} peuvent être regroupées en une seule : $Xc_i \wedge Condd_j = 0$, qui détecterait les mêmes positions interdites. Cependant, l'explication associée aux contraintes CSD_{ii} et $CSD_{iii,}$ serait perdue.

4.3.2.3 Exemples de contraintes

A titre d'exemple, nous allons traiter le partage d'une zone commune entre deux vérins pilotés par des distributeurs 5/2 et définir les contraintes de sécurité.



a) Système

	Variables système	Variables API
Vérin vertical	DE, MO, h, b	DE_auto, MO_auto, h_auto, b_auto
Vérin horizontal	AV, RE, r, a	AV_auto, RE_auto, r_auto, a_auto

b) Variables

Figure 3.15. Exemple vérin zone commune et leur commande

Il y a une interaction entre les deux ElPO provenant du partage de la zone commune. L'analyse dysfonctionnelle met en évidence que la collision entre les 2 vérins doit être évitée. Il faut aussi prendre en compte les erreurs que peut faire le concepteur sur un ElPO. Ici, seul

un exemple de chaque type de contrainte est donné et l'exemple sera traité complètement dans le chapitre 4 :

- CSS_i : $AV\_auto \wedge RE\_auto = 0$ représente l'impossibilité technique pour le vérin horizontal de se déplacer dans les deux sens en même temps.
- CSS_{ii} : $AV\_auto \wedge DE\_auto = 0$ représente l'interdiction de demander au deux vérins de se déplacer l'un vers l'autre
- CSD_i : $\uparrow AV\_auto \wedge \neg h\_auto = 0$ représente le fait que le vérin horizontal peut avancer à la condition que *h* soit à 1. Dans le cas contraire, l'ordre *AV* est interdit si le vérin vertical n'est pas en *h*.
- CSD_{ii} : $\downarrow MO\_auto \wedge \neg h\_auto = 0$ montre que l'on ne peut désactiver l'ordre *MO* seulement si le vérin est en *h*.
- CSD_{iii} : $\uparrow AV\_auto \wedge a\_auto = 0$ représente l'interdiction d'activer l'ordre *AV*, si le vérin est en *a*. En effet, l'activation n'aura aucun effet sur le système.
- CSD_{iv} : $AV\_auto \wedge \uparrow a\_auto = 0$ représente l'obligation d'arrêter l'ordre *AV*, si le vérin vient activer le capteur *a*. En effet, l'activation n'aura aucun effet sur le système.

La définition de ce canevas de contraintes permet d'accompagner l'expert dans sa démarche de sécurisation du système au niveau du filtre de validation système et dans l'exécution correcte des fonctions au niveau du filtre de validation fonctionnelle. Le travail de définition des contraintes est simplifié car le système n'est plus vu d'une manière globale, mais par ElPO pris indépendamment ou en interaction. En effet, l'expert traite dans un premier temps les contraintes pour chaque ElPO pris indépendamment, puis pour chaque interaction entre ElPO, puis pour chaque interaction avec le produit. La partie suivante reprend chaque étape pour mettre en place la validation de la commande : analyse fonctionnelle, dysfonctionnelle et définition des contraintes sur l'exemple du tri de caisses.

5 Exemple

Dans le chapitre 2 § 4.2, l'exemple du tri de caisses a été étudié pour expliquer l'approche de validation hors ligne par synthèse. Ce système est repris pour appliquer

l'approche de validation en ligne en effectuant le découpage fonctionnel, l'analyse dysfonctionnelle et la définition des contraintes.

5.1 Découpage fonctionnel et analyse dysfonctionnelle

Pour étudier le système, la fonction principale est définie puis décomposée pour arriver au niveau le plus bas, celui des capteurs/actionneurs. Le tableau de la figure 3.16. présente l'analyse fonctionnelle et dysfonctionnelle du système.

Analyse fonctionnelle					Dysfonctionnements possibles
Trier des caisses	Alimenter en caisses	Tapis1			
	Placer la caisse	Mettre caisse devant V_3	Faire un aller retour du vérin V_1	Sortir V_1	Vérin V_3 pas en position rentrée (Dp1)
				Rentrer V_1	
			Faire un aller retour du vérin V_2	Sortir V_2	(Dp1) Présence d'une grande caisse (Dp2)
				Rentrer V_2	
		Mettre caisse devant V_4	Faire un aller retour du vérin V_1 et V_2	Sortir V_1	Vérin V_4 pas en position rentrée (Dp3)
				Rentrer V_1	
				Sortir V_2	(Dp3)
				Rentrer V_2	
	Évacuer	Mettre la caisse sur le tapis d'évacuation	Faire un aller retour du vérin V_3	Sortir V_3	(Dp2)
				Rentrer V_3	
			Faire un aller retour du vérin V_4	Sortir V_4	Présence d'une petite caisse (Dp4)
				Rentrer V_4	
		Activer le tapis	Tapis 2		
			Tapis 3		

Figure 3.16. Analyse fonctionnelle et dysfonctionnelle

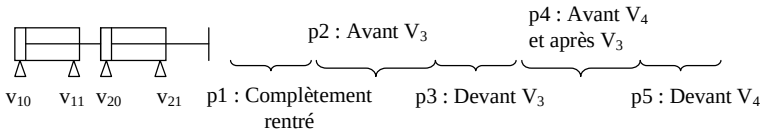
Ainsi, les détériorations possibles sont dues à un déplacement d'un vérin alors qu'un autre vérin n'est pas en position rentrée. Le non-respect du cahier des charges revient quant à lui à un mauvais positionnement des caisses (c'est-à-dire une grande caisse dans le stock petites caisses, et inversement).

Comme nous l'avons vu, pour fournir des explications à l'apprenant, certaines erreurs ne conduisant pas forcément à une dégradation du système peuvent être prises en compte. Il peut s'agir de :

- L'envoi de deux ordres inverses sur un même ElPO
- L'envoi d'un ordre alors que les conditions de désactivation sont déjà présentes
- La non désactivation d'un ordre lorsque les conditions de désactivation deviennent vraies

Pour détecter ce type d'erreur, l'expert définit des contraintes en utilisant le canevas précédemment présenté pour chaque ElPO pris indépendamment.

À partir de cette analyse, les positions interdites peuvent être définies par rapport aux interactions : entre V_1 et V_2 (lien physique), entre V_1, V_2, et V_3 (zone commune), entre V_1, V_2 et V_4 (zone commune). Pour la première interaction, aucun état n'est défini comme interdit. Pour la deuxième interaction, les ElPO V_1V_2 peuvent prendre cinq positions : $\{p_1, p_2, p_3, p_4, p_5\}$ (figure 3.17.) et l'ElPO V_3 peut prendre 3 positions : $\{p_{10}, p_{11}, p_{12}\}$. Le vérin V_3 ne pourra se déplacer que si le vérin composé V_1V_2 se trouve en position p_1 et le vérin V_1V_2 ne pourra se déplacer que si V_3 est en position p_{10}. Pour la troisième interaction, le vérin V_4 ne pourra pas se déplacer si le vérin V_1V_2 est complètement sorti (p_5)



v10 v11 / v20 v21	10	00	01
10	p1	p2	p3
00	p2	p2 ou p3 ou p4	p4
01	p3	p4	p5

Figure 3.17. Positions prises par le vérin V_1V_2

Une fois les interactions définies, la liste de contraintes est établie pour les ElPO pris indépendamment, les interactions entre ElPO et avec le produit.

5.2 Définition des contraintes

En fonction des différents dysfonctionnements possibles, l'expert va établir la liste de contraintes pour chaque ElPO, pour les interactions entre les ElPO et pour l'interaction avec le produit.

5.2.1 Contraintes pour chaque ElPO indépendamment

Chaque ElPO du système de tri de caisses est pris indépendamment en considérant la technologie. Cela permet de prendre en compte les erreurs pouvant être commises par le concepteur sur un ElPO. Le vérin V_1 et le vérin V_2 sont des bistables, il faut donc définir cinq contraintes pour chaque vérin.

- La première contrainte est une contrainte de sécurité statique (CSS_i) représentant l'impossibilité technique du vérin de se déplacer dans les deux sens en même temps :

$$AVANCER1 \wedge RECULER1 = 0 \quad (3.8)$$

- Les seconde et troisième contraintes sont des contraintes de sécurité dynamiques (CSD_{iii}) permettant de valider que la commande n'est pas envoyée si les conditions de désactivation sont présentes :
  - Interdire l'ordre AVANCER1, si le vérin V_1 est déjà en v_{11} :

$$\uparrow AVANCER1 \wedge v_{11} = 0 \quad (3.9)$$

  - Interdire l'ordre RECULER1, si le vérin V1 est déjà en v_{10} :

$$\uparrow RECULER1 \wedge v_{10} = 0 \quad (3.10)$$

- Les quatrième et cinquième contraintes sont des contraintes de sécurité dynamiques (CSD_{iv}) qui vérifient que l'ordre est arrêté lorsque les conditions de désactivation occurrent
  - Désactiver l'ordre RECULER1, si le vérin V_1 est arrivé en v_{10} :

$$\uparrow v_{10} \wedge RECULER1 = 0 \quad (3.11)$$

  - Désactiver l'ordre AVANCER1, si le vérin V_1 est arrivé en v_{11} :

$$\uparrow v_{11} \wedge AVANCER1 = 0 \quad (3.12)$$

La même logique est utilisée pour définir les contraintes sur le vérin V_2 :

- Interdire l'envoi des deux ordres en même temps :

$$AVANCER2 \wedge RECULER2 = 0 \quad (3.13)$$

- Interdire l'ordre *AVANCER2*, si le vérin V_2 est déjà en v_{21} :

$$\uparrow AVANCER2 \wedge v_{21} = 0 \quad (3.14)$$

- Interdire l'ordre *RECULER2*, si le vérin V_2 est déjà en v_{20} :

$$\uparrow RECULER2 \wedge v_{20} = 0 \quad (3.15)$$

- Désactiver l'ordre *RECULER2*, si le vérin V_2 est arrivé en v_{20} :

$$\uparrow v_{20} \wedge RECULER2 = 0 \quad (3.16)$$

- Désactiver l'ordre *AVANCER2*, si le vérin V_2 est arrivé en v_{21}

$$\uparrow v_{21} \wedge AVANCER2 = 0 \quad (3.17)$$

Pour les deux autres vérins V_3 et V_4 et pour les tapis, il n'y a pas de contraintes car la technologie utilisée n'en nécessite pas.

5.2.2 Contraintes pour les interactions entre ElPO

Les contraintes définies pour les interactions doivent permettre de détecter les dysfonctionnements Dp1, et Dp3 (cf. figure 3.16.). Pour le premier dysfonctionnement (Dp1), les interactions prises en compte sont celles entre V_1 et V_3 et entre V_2 et V_3 ; il faut définir les 6 contraintes suivantes :

- Les contraintes 3.18 et 3.19 représentent l'impossibilité aux deux vérins de se déplacer l'un vers l'autre (CSS_{ii}), c'est-à-dire interdire l'envoi des deux ordres d'avancer en même temps :

$$AVANCER1 \wedge AVANCER3 = 0 \tag{3.18}$$

$$AVANCER2 \wedge AVANCER3 = 0 \tag{3.19}$$

- Les contraintes 3.20 et 3.21 vérifient les conditions d'activation (CSD_i) pour envoyer la commande du vérin V_3. Le vérin V_1 doit être en position rentrée :

$$AVANCER3 \wedge \neg v_{10} = 0 \tag{3.20}$$

Il en est de même pour V_2 :

$$AVANCER3 \wedge \neg v_{20} = 0 \tag{3.21}$$

- Les contraintes 3.22 et 3.23 permettent de rendre équivalent le sens de déplacement (CSD_{ii}) du vérin V_1 et l'ordre AVANCER1 :

$$\downarrow AVANCER1 \wedge \neg v_{11} = 0 \tag{3.22}$$

De même pour le vérin V_2 :

$$\downarrow AVANCER2 \wedge \neg v_{21} = 0 \tag{3.23}$$

Pour le deuxième dysfonctionnement (Dp3), l'interaction considérée est celle entre les vérins V_1, V_2 et V_4 ; il y a 6 contraintes à définir :

- Interdire l'envoi des trois ordres d'avancer en même temps (CSS_{ii}) :

$$AVANCER1 \wedge AVANCER2 \wedge AVANCER4 = 0 \tag{3.24}$$

- Interdire les ordres *AVANCER4 et AVANCER1,* si le vérin V_2 est en v_{21} (CSS_{ii}) :

$$AVANCER4 \wedge AVANCER1 \wedge v_{21} = 0 \tag{3.25}$$

- Interdire les ordres *AVANCER4 et AVANCER2,* si le vérin V_1 est en v_{11} (CSS_{ii}) :

$$AVANCER4 \wedge AVANCER2 \wedge v_{11} = 0 \tag{3.26}$$

- Interdire l'ordre *AVANCER4*, si le vérin V_2 n'est pas en v_{20} et le vérin V_1 est en v_{11} (CSD_i) :

$$AVANCER4 \wedge \neg v_{20} \wedge v_{11} = 0 \quad (3.27)$$

- Interdire l'ordre *AVANCER4*, si le vérin V_1 n'est pas en v_{10} et le vérin V_2 est en v_{21} (CSD_i) :

$$AVANCER4 \wedge \neg v_{10} \wedge v_{21} = 0 \quad (3.28)$$

- Interdire l'ordre *AVANCER4*, si le vérin V_1 n'est pas en v_{10}, ni en v_{11} et le vérin V_2 n'est pas en v_{20}, ni en v_{21} (CSD_i) :

$$AVANCER4 \wedge \neg v_{10} \wedge \neg v_{11} \wedge \neg v_{20} \wedge \neg v_{21} = 0 \quad (3.29)$$

5.2.3 Contraintes pour les interactions avec le produit

Pour éviter les dysfonctionnements Dp2 et Dp4, il faut considérer les interactions entre le vérin V_3 et la petite caisse, entre le vérin V_4 et la grande caisse et entre le vérin V_2 et la grande caisse. Pour cela trois contraintes de sécurité dynamiques pour vérifier que la condition d'activation du bon type de caisse sont définies :

- Interdire l'ordre AVANCER3, s'il n'y a pas de petites caisses (CSD_i) :

$$AVANCER3 \wedge \neg cp_{11} = 0 \quad (3.30)$$

- Interdire l'ordre AVANCER4, s'il n'y a pas de grandes caisses (CSD_i) :

$$AVANCER4 \wedge \neg cp_{12} = 0 \quad (3.31)$$

- Interdire l'ordre AVANCER2, s'il n'y a pas de grandes caisses (CSD_i) :

$$AVANCER2 \wedge \neg cp_{12} = 0 \quad (3.32)$$

Pour assurer la sécurité du système au niveau capteurs/actionneurs, 25 contraintes ont du être définies.

6 Conclusion

Ce troisième chapitre apporte une réponse pour assurer que la commande proposée par le concepteur, valide un ensemble de contraintes de sécurité et de spécifications du cahier des charges. Pour cela, une approche par filtre entre la PO et la PC a été proposée, où l'expert doit

mettre en place des contraintes pour assurer le maximum de sécurité et pour adapter le système au niveau du concepteur.

Le travail proposé dans ce chapitre a porté sur deux points : d'une part *Comment mettre à disposition un système global en prenant en compte le niveau de compétences du concepteur ?* et d'autre part *Comment assurer le maximum de sécurité pour le système ?* :

- Pour adapter la vision du système au niveau de connaissances du concepteur, l'expert étudie le système et en fonction des savoirs du concepteur, il définit le champ d'actions de celui-ci. En effet, l'approche propose d'encapsuler des parties de programme dans des fonctions, permettant ainsi de faire abstraction de structures ou de modes de marche particuliers. Les fonctions sont définies par des conditions d'activation et de désactivation et par un mode d'exécution donnant plus ou moins d'autonomie au concepteur. La définition des fonctions se fait simplement par une analyse descendante du système.
- Pour assurer la sécurité du système, un filtre a été placé entre la PO et la PC. Ce filtre est composé d'un filtre de validation fonctionnel, d'un filtre de validation système, et de deux blocs pour exécuter les fonctions au niveau système et pour adapter l'information système aux savoirs du concepteur. Les filtres contiennent un ensemble de contraintes définissant ce que le système ne doit pas faire et ce qui est attendu par le cahier des charges. La définition des contraintes est une tâche difficile pour l'expert et importante pour que le système soit correctement sécurisé. En effet, la sécurité est complètement basée sur la bonne définition des contraintes. Pour accompagner l'expert dans la définition de cet ensemble, il lui est proposé d'identifier les interactions qui existent dans le système : entre les ElPO et avec le produit, ensuite, pour chaque interaction, l'expert utilise un canevas de contraintes définies au niveau de la sécurité. Il est possible d'ajouter des contraintes pédagogiques qui apportent une explication en cas de non-respect mais ne représentent pas des situations dangereuses pour le système.

La validation de la commande proposée par le concepteur s'effectue simplement en appliquant des contraintes et en s'assurant qu'elles sont toutes respectées avant l'envoi vers la PO. Cette approche de validation repose sur la définition correcte des contraintes que le canevas proposé ne peut hélas pas garantir. Pour répondre à ce manque, le chapitre suivant propose une méthode pour vérifier formellement que l'ensemble des contraintes est suffisant pour assurer la sécurité du système.

Chapitre 4 :
Vérification formelle des contraintes de sécurité

1 Introduction

Dans le chapitre précédent, nous avons proposé une approche de validation de la commande au moyen d'un filtrage en ligne des ordres envoyés de la Partie Commande (PC) vers la Partie Opérative (PO). Le filtrage est réalisé grâce à un ensemble de contraintes devant être respectées continuellement pour garantir la sécurité des installations.

La démarche proposée pour la définition des contraintes n'est pas formelle. Même si leur écriture est générique, la manière de les déterminer, comme nous l'avons vu, dépend de la PO, du produit, de son instrumentation, et de l'expert. De plus, le filtre de commande est implémenté dans un Automate Programmable Industriel (API), dont le fonctionnement séquentiel et cyclique a nécessairement des conséquences sur la validité des contraintes (évolutions simultanées, retard de causalité). Il est supposé que toutes les évolutions de la PO sont détectables par l'API.

Il est donc important de s'assurer, pour une PO donnée, que l'ensemble des contraintes proposées est suffisant pour garantir la sécurité du système. Celle-ci est assurée, comme nous l'avons vu dans le chapitre précédent, en interdisant d'une part, pour chaque élément de PO (ElPO) des vecteurs d'entrées/sorties particuliers, et d'autre part, des positions particulières

lorsque des ElPO sont en interaction avec la présence ou non d'un produit. C'est bien entendu ce dernier cas qui est le plus complexe et qui nécessite de s'assurer que les contraintes ont correctement été écrites.

Nous proposons dans ce chapitre, une approche de vérification formelle de l'écriture des contraintes pour des ElPO en interaction (Marangé et al., 2008 b). La vérification s'effectue sur un modèle du sous-système de PO étudié, en considérant le modèle de commandes le plus permissif possible, et l'ensemble des contraintes définies par l'expert pour le sous-système de PO considéré. Tous ces modèles sont définis au travers d'un modèle de l'environnement d'évolution de la commande qui caractérise le fonctionnement de l'API. En utilisant l'approche par Model-Checking (Schnoebelen et al. 1999), tous les chemins pouvant être parcourus par le système sont calculés en s'assurant qu'ils ne contiennent pas d'états interdits. Cela permet ainsi de vérifier que les contraintes sont suffisantes.

Pour montrer la nécessité de vérifier les contraintes, 2 exemples vont être présentés. Le premier est tiré de la thèse de Deschamps (Deschamps, 2007). L'exemple traite d'une zone commune partagée par deux vérins sans prendre en compte la technologie des vérins. Ici, il est supposé que les vérins sont pilotés par des distributeurs 5/2 bistable (figure 4.1.b). On retrouve un système analogue (figure 4.1.a) sur la machine PRODUCTIS (Schneider Electric, 2001) dont dispose l'IUT de Reims-Châlons-Charleville et qui sera détaillé dans le chapitre 5. Il s'agit en réalité d'un préhenseur composé de 2 vérins couplés. En d'autres termes, des mouvements en diagonal sur notre préhenseur correspondent à une violation de la zone commune dans l'exemple de Deschamps, et surtout, sont sans conséquence, du point de vue de la sécurité, pour le système.

Les deux contraintes de sécurité proposées par Deschamps pour l'envoi des commandes *AV* et *DE*, sont sur la position que doit occuper l'autre vérin. Un vérin ne peut pas être déplacé si l'autre n'est pas rentré. La notation « auto » a été choisie pour montrer qu'il s'agit de variables de l'API.

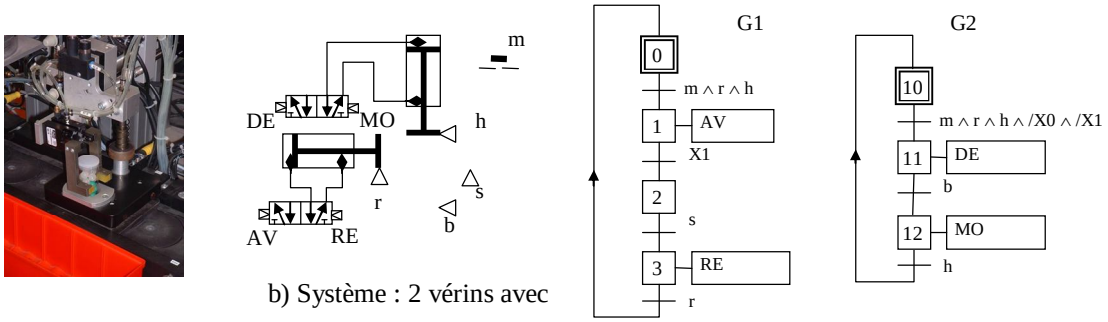
$$AV\_auto \wedge \neg h\_auto = 0 \qquad (4.1)$$

$$DE\_auto \wedge \neg r\_auto = 0 \qquad (4.2)$$

Nous rajoutons une contrainte statique supplémentaire triviale qui consiste à ne pas envoyer simultanément les commandes de sortir des 2 vérins.

$$AV\_auto \wedge DE\_auto = 0 \qquad (4.3)$$

La commande présentée figure 4.1.c a été programmée dans un Automate PREMIUM de la marque SCHNEIDER. L'objectif n'est pas d'émettre un avis sur la qualité du programme où encore de la valider vis-à-vis d'un cahier des charges. On constate la présence d'une étape fugitive (étape 1). Ce programme correspond à une commande possible tout à fait réalisable par un apprenant. Nous désirons déterminer si cette commande est sûre de fonctionnement. En d'autres termes, si les 2 vérins peuvent rentrer en collision.



a) Système réel

b) Système : 2 vérins avec distributeurs 5/2

c) Commande

Figure 4.1. Exemple vérin zone commune et leur commande

Plusieurs essais ont été réalisés sur le préhenseur de la PRODUCTIS en faisant varier le temps de cycle de l'API. Trois cas de figure ont été constatés.

- Cas 1 : si le temps de cycle est inférieur à 10 ms, celui-ci est trop court pour permettre au distributeur du vérin horizontal de se positionner. Par conséquent, uniquement le vérin vertical effectue des allers-retours.

- Cas 2 : si le temps de cycle est supérieur à 20 ms, il n'y a pas de collision et les 2 vérins effectuent des allers-retours à tour de rôle si m est maintenu appuyé. Cela s'explique par le fait que pendant un temps de cycle de l'API, le vérin horizontal a le temps de quitter le capteur *r*. Par conséquent, la réceptivité associée à la transition entre les étapes 10 et 11 n'est pas vraie, et l'étape 11 n'est pas activée.

- Cas 3 : si le temps de cycle est compris entre 10 ms et 20 ms, les 2 vérins effectuent des allers-retours simultanément, ce qui correspondrait à des collisions. Ce comportement s'explique car le temps de cycle est suffisant pour positionner le distributeur du vérin horizontal, mais insuffisant pour permettre un déplacement permettant au vérin de quitter le capteur *r*. Comme il s'agit de distributeurs 5/2 bistable, le vérin continue de se déplacer même si la commande n'est plus maintenue. Par conséquent, lorsque l'étape 2 est activée, la réceptivité associée à la transition entre les étapes 10 et 11 est vraie. La transition est donc franchie. La descente du vérin vertical est

activée et le vérin horizontal continue son déplacement. On constate que les contraintes définies par Deschamps ne sont pas suffisantes dans ce cas pour garantir la sécurité du système.

Ce premier exemple permet de mettre en évidence l'importance de considérer le programme de commande le plus permissif possible et la nécessité de prendre en compte d'une part le fonctionnement de l'API et d'autre part la technologie des actionneurs et pré-actionneurs dans l'écriture des contraintes. Étant donné que le danger vient du déplacement des actionneurs, le cas le plus contraignant revient à considérer qu'un cycle de l'API est suffisant pour déclencher le mouvement de l'effecteur.

Un deuxième exemple proposé met en évidence l'importance du retard de causalité qui est lié à l'API et à l'instrumentation dans la définition des contraintes. On considère un système composé d'un chariot (pouvant avoir de l'inertie) et d'un capteur de position *c*. Ce capteur est caractérisé par un contact d'une certaine longueur. On désire que le chariot arrête de se déplacer lorsqu'il arrive sur le contact *c*. Soit les deux cas de la figure 4.2., où au cycle *i*, il est demandé d'arrêter le chariot. Dans le 1er cas, le chariot s'arrête correctement sur le capteur alors que dans le 2ème cas, du fait du retard de causalité (i.e. temps de cycle de l'API) et/ou de l'inertie, le chariot s'arrête après le capteur *c*. On comprend facilement que la définition des contraintes et l'approche de vérification de l'ensemble doivent tenir compte de ce type de fonctionnement. Toutefois, même dans le cas 1, si l'ordre d'arrêt n'est pas donné lors de l'activation du contact *c*, le chariot peut s'arrêter après avoir désactivé *c*. Dans le cadre de ce travail, nous avons considéré uniquement des capteurs ayant des contacts longs (cas 1). Toutefois, pour s'arrêter sur le contact, l'ordre d'arrêt doit être donné sur un front montant du capteur *c*. Nous reviendrons sur le cas 2 dans les perspectives de recherche.

Ce deuxième exemple met aussi en évidence le fait que si l'on désire faire le minimum d'hypothèses sur le fonctionnement de la PO, il faut nécessairement contraindre plus fortement la commande.

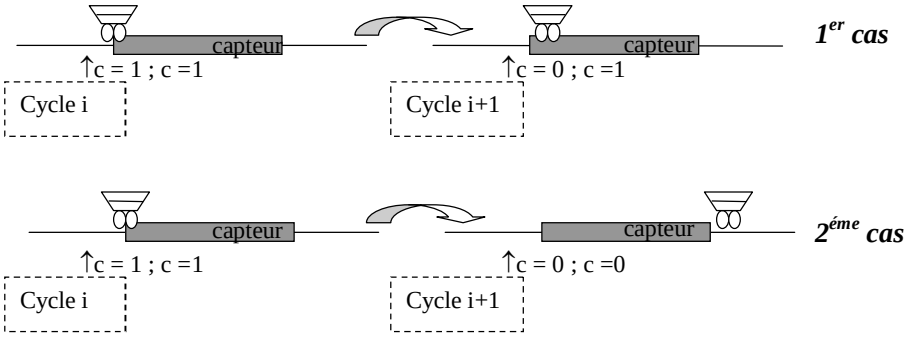


Figure 4.2. Problématique du contact long

Dans la première partie de ce chapitre, la méthodologie de vérification des contraintes est détaillée. La vérification passe par une étape de modélisation modulaire du système complet et une vérification modulaire des contraintes. La seconde partie présente les différents modèles proposés pour représenter de façon fine le comportement du système global (PO +PC). Pour cela, l'approche proposée repose sur une vision causale du système, où la PC génère des sorties (i.e. modèle de commandes) qui vont éventuellement engendrer un déplacement (i.e. modèle de sens) des effecteurs et/ou des produits qui sont repérés au moyen de leurs positions (modèle de positions). Tous ces modèles sont construits à partir d'une vision du système vu de l'API (i.e. modèle de l'environnement de calcul). La troisième partie du chapitre présente l'étape de vérification des contraintes. On considère uniquement les interactions entre ElPO avec ou sans produit. L'implémentation de cette approche de vérification des contraintes repose sur l'utilisation d'automates communicants et du model checker UPPAAL (Behrmann et al., 2002). L'ensemble est illustré au moyen d'exemples pédagogiques mais représentatifs des systèmes rencontrés dans les procédés manufacturiers.

2 Méthodologie de vérification formelle des contraintes

Il est indispensable quelle que soit l'approche de validation de la commande proposée, de s'assurer que celle-ci est sûre de fonctionnement pour garantir la sécurité des biens et des équipements. Il est donc fondamental, suite à la définition des contraintes de se poser les questions suivantes : *Les contraintes ont-elles été correctement définies ? Sont-elles suffisantes pour garantir la sécurité?* Ces questions sont dans notre cas d'autant plus légitimes que la définition des contraintes ne repose pas sur une méthode formelle.

La définition des contraintes, comme cela a été indiqué dans le chapitre 3, est basée sur

une analyse fonctionnelle et dysfonctionnelle du système qui permet de mettre en évidence les situations que la PO ne doit en aucun cas atteindre. Dans l'approche qui a été proposée, les situations interdites sont, soit des vecteurs d'entrées-sorties particuliers, soit des positions physiquement dangereuses pour les ElPO, soit une position du produit « inacceptable ».

Les contraintes permettent d'éviter que le système se retrouve dans une situation non autorisée. Pour cela, deux formes d'écriture ou type de contraintes ont été proposées : contraintes de sécurité statiques et contraintes de sécurité dynamiques. Ces contraintes sont caractéristiques d'un ElPO, de l'interaction entre des ElPO, ou encore du produit manipulé par les ElPO et dépendent de l'instrumentation du système. Les contraintes mises en place sur un ElPO correspondent par exemple à des fins de course de vérin qui ont été atteintes et qui ne peuvent pas être dépassées. Dans ce cas, cela ne représente pas un réel danger et par conséquent elles ne nécessitent pas d'être vérifiées formellement. Toutefois, ces contraintes correspondent à des erreurs de commande pédagogiques (cf. chapitre 3) et permettent aussi de simplifier les modèles caractéristiques des ElPO. En revanche, comme nous l'avons montré au travers des 2 exemples introductifs, le problème n'est pas trivial pour la vérification de la sécurité des ElPO en interaction et la prise en compte du produit. C'est dans ce cadre que nous proposons une approche originale de vérification dont le principe consiste à montrer la disjonction entre les situations de PO interdites et les situations atteignables après la mise en place des contraintes et en considérant la commande la plus permissive possible (figure 4.3.).

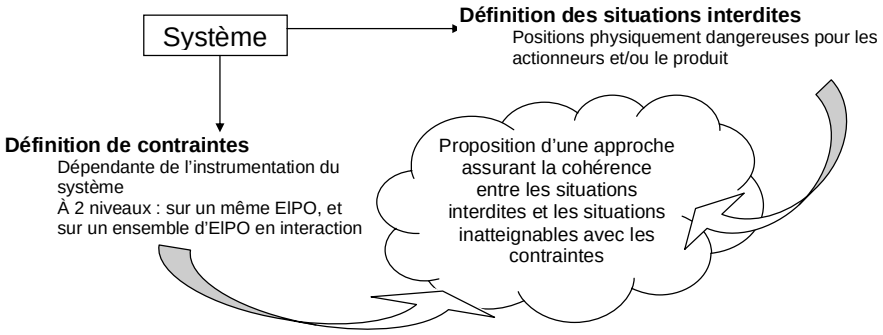


Figure 4.3. Principe de vérification des contraintes

La figure 4.4. présente la démarche de vérification formelle des contraintes permettant d'assurer la robustesse du filtre de validation de commande :

1. L'ensemble des positions et situations interdites est déterminé à partir d'une analyse fonctionnelle et dysfonctionnelle du système.

2. L'ensemble des propriétés formelles est déduit des positions interdites, ainsi que l'ensemble de contraintes à vérifier.

3. L'ensemble de contraintes est correcte si les propriétés formelles sont vérifiées, sous le logiciel UPPAAL, sur le modèle de PO en interaction avec le modèle d'API. Le modèle d'API contient la commande la plus permissive ainsi que les contraintes à vérifier.

4. Si les propriétés sont satisfaites, elles sont implémentées dans l'API et apportent une explication en cas de non respect d'une contrainte. Sinon, il faut modifier l'ensemble de contraintes et refaire la vérification des propriétés jusqu'à ce qu'elles soient satisfaites (étape 3).

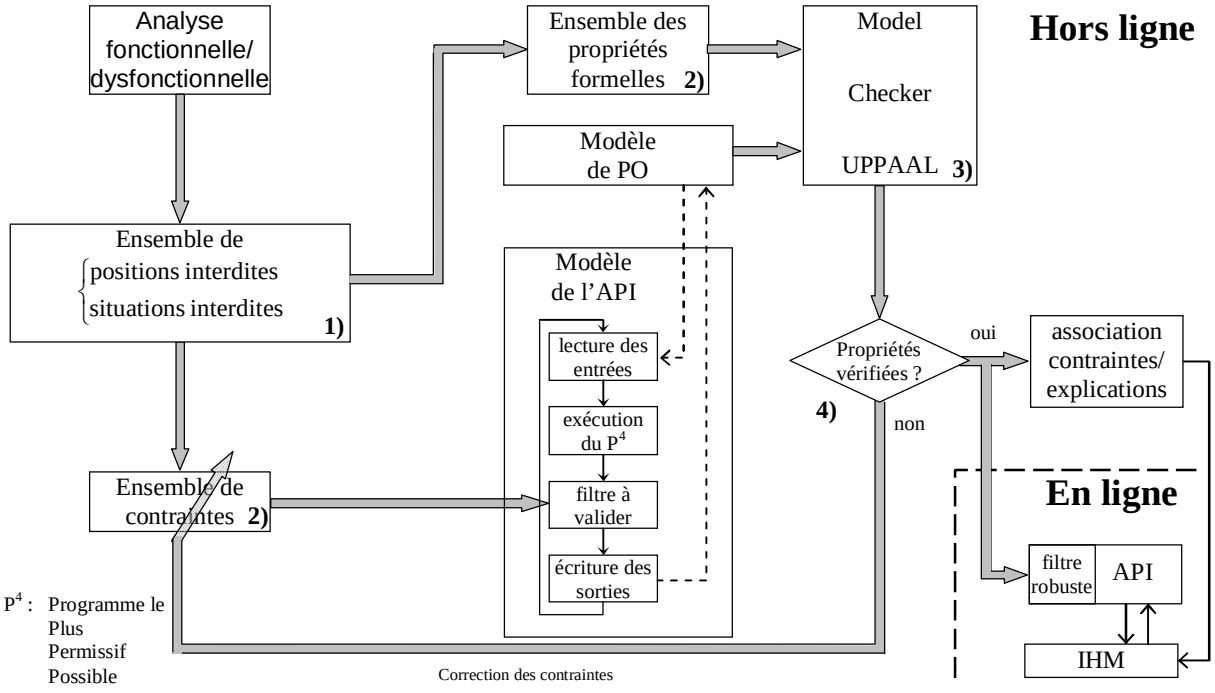


Figure 4.4. Démarche de vérification des contraintes

La vérification des contraintes consiste à vérifier formellement au moyen d'un model-checker (UPPAAL dans notre cas) que l'ensemble des contraintes proposées par l'expert est suffisant et nécessaire pour garantir que la PO (actionneurs et produit) n'atteint jamais, quelle que soit la commande envoyée, des états interdits. Les états interdits sont définis par l'expert lors de l'analyse dysfonctionnelle du système (cf. chapitre 3). Comme le montre la figure 4.3, l'originalité de la démarche est de s'assurer que, grâce aux contraintes, et en considérant la commande la plus permissive possible, il n'existe pas de chemins menant la PO dans une situation de danger. En d'autres termes, on ne cherche pas à vérifier qu'une commande

particulière répond à un cahier des charges et est sûr de fonctionnement, mais on cherche à vérifier qu'un ensemble de contraintes est suffisant pour garantir la sécurité indépendamment du cahier des charges et de la commande. Si la propriété « aucun état dangereux n'est atteint » n'est pas vérifiée, cela signifie que l'ensemble des contraintes n'est pas suffisant, il convient donc de revoir et de modifier la liste.

L'ensemble des contraintes validé formellement peut ensuite être implémenté dans l'API et permet d'obtenir un filtre robuste de commande suffisant pour garantir la sécurité du système. De plus, chaque contrainte est une source d'informations. Si elle n'est pas respectée par le programme de commande, cela signifie qu'il existe au moins un chemin, que le model-checking peut fournir, conduisant la PO dans une situation dangereuse. La connaissance du ou des chemins permet de fournir une explication à l'opérateur. Étant donné que l'ensemble de contraintes est uniquement caractéristique et dépendant de la PO, le filtre reste valable même s'il y a des changements dans le programme de commande. Il s'agit là d'une propriété intéressante car il y a une séparation entre les aspects fonctionnels et sécuritaires.

Cette approche nécessite de modéliser la PO, le fonctionnement de l'API, les contraintes et les propriétés à vérifier. Ce dernier point est souvent problématique car une des difficultés du model-checking est que les propriétés ne sont pas toujours évidentes à formaliser. Cela n'est pas le cas dans l'approche proposée. La propriété correspond à une ou plusieurs positions physiques des actionneurs et/ou des pièces et est donc simple à écrire.

La méthodologie nécessite de vérifier que pour une PO donnée, aucune situation dangereuse n'est atteinte. On peut donc craindre une explosion combinatoire. Nous montrerons qu'une approche modulaire de modélisation de la PO et de vérification permet d'éviter ce problème. En revanche, chaque ensemble de contraintes est spécifique à une interaction entre ElPO et/ou de produits. Toutefois, une fois vérifiée, les contraintes sont toujours valables si l'interaction se retrouve dans une autre PO. Il est donc possible de créer une bibliothèque d'ElPO en interaction avec leurs contraintes vérifiées formellement. Dans le cadre de cette thèse, plusieurs PO seront étudiées. Bien entendu, ces exemples ne sont pas exhaustifs. Dans cette thèse, nous nous sommes intéressés exclusivement à des vérins et des moteurs et aux interactions entre ces actionneurs et avec un produit.

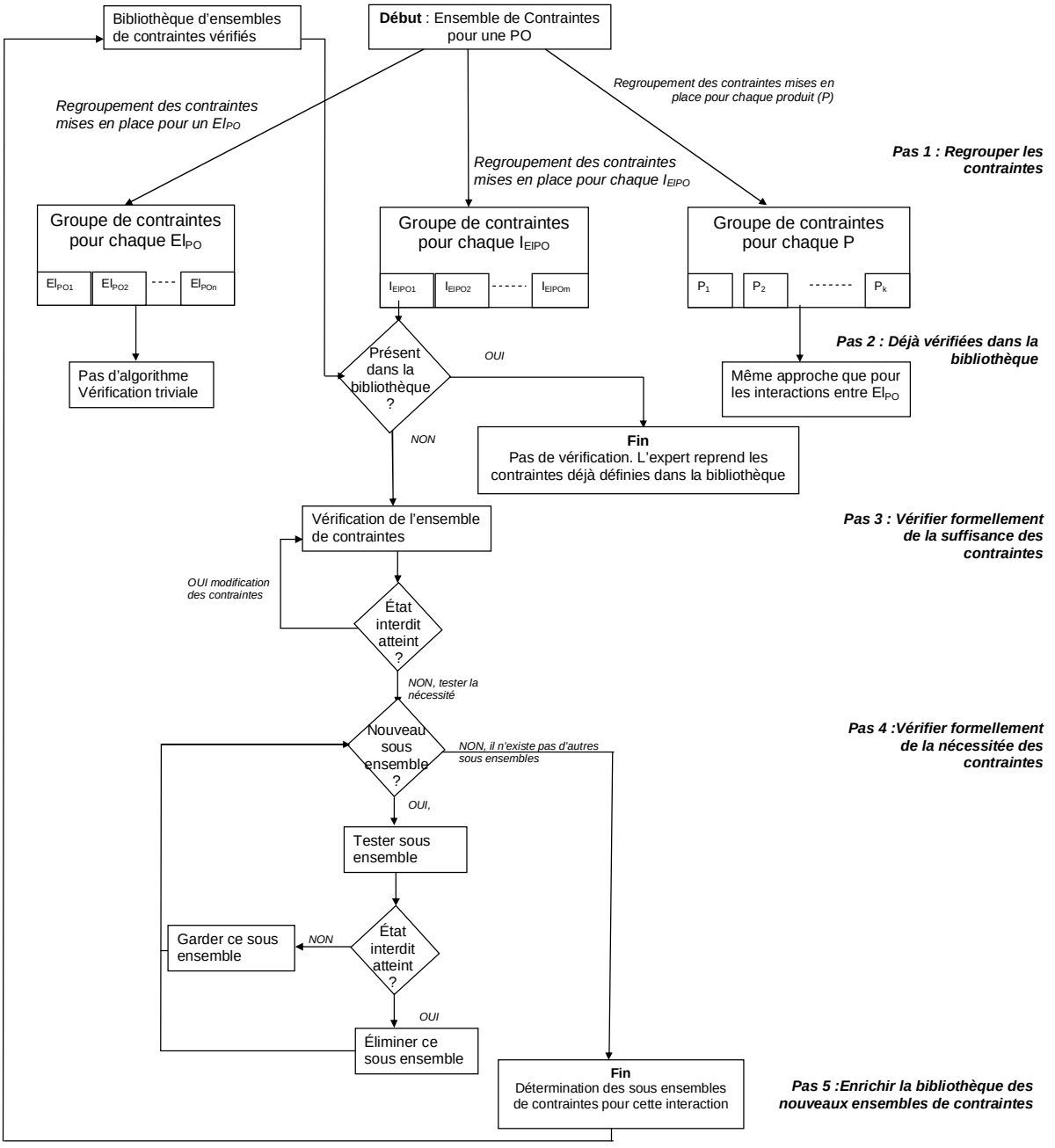


Figure 4.5. Méthodologie de vérification des contraintes

Les points, qui viennent d'être évoqués, nous ont conduits à proposer la méthodologie globale de vérification de l'ensemble des contraintes, présentée dans la figure 4.5. La démarche s'applique sur une PO qui est caractérisée par un ensemble d'ElPO et de produits, en interaction entre eux et d'un point de vue Entrées/Sorties de la PC. La méthode s'applique en 5 pas :

- Le premier pas regroupe les contraintes pour chaque ElPO, chaque interaction entre ElPO (I_{ElPO}), et chaque interaction avec le produit (P) (figure 4.6.). En effet, pour ne

pas être confronté à un problème d'explosion combinatoire, les contraintes vont être vérifiées de façon modulaire pour chacun des trois cas. Il est à noter qu'une même contrainte peut être mise en jeu dans plusieurs interactions.

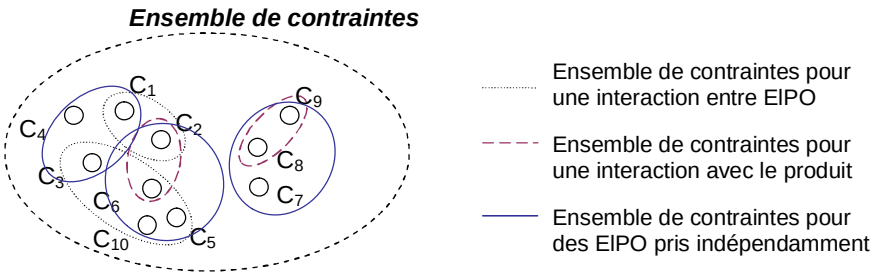


Figure 4.6. Ensembles d'interaction sur la PO

- le second pas, à partir de la bibliothèque de contraintes, regarde si l'interaction a déjà été traitée, car si elle a été vérifiée, il est inutile de revérifier les contraintes. En effet, il est possible d'utiliser directement celles se trouvant dans la bibliothèque en fonction de l'explication souhaitée.

- Le troisième pas vérifie la suffisance des contraintes seulement pour chaque interaction. En effet, les contraintes spécifiques à chaque ElPO pris indépendamment, ne font pas l'objet d'une vérification particulière. La démarche de vérification s'applique pour les ElPO et le produit qui présentent des interactions notées I_{ElPOj} et P_i. Le principe consiste pour chaque I_{ElPOj} et P_i de vérifier si l'ensemble de contraintes défini ne permet pas d'atteindre un état interdit. Si ce n'est pas le cas, l'expert doit redéfinir son ensemble de contraintes et le revérifier.

- Le quatrième pas, une fois que la suffisance de l'ensemble de contraintes a été assurée, vérifie que les contraintes sont toutes nécessaires. Pour cela, à partir de l'ensemble de contraintes, toutes les combinaisons de contraintes sont définies comme des sous-ensembles et chaque sous-ensemble est vérifié. Si le sous-ensemble de contraintes ne permet pas d'atteindre un état interdit, il est gardé pour la bibliothèque, sinon, il est éliminé.

- Le cinquième et dernier pas permet d'enrichir la bibliothèque de plusieurs ensembles de contraintes pour une nouvelle interaction. Le choix du sous-ensemble à implémenter est lié aux capacités explicatives que l'expert souhaite donner aux contraintes. Comme cela a été montré dans le chapitre 3, les contraintes non respectées sont des sources d'informations pour expliquer les erreurs de commande.

Cette approche modulaire permet d'éviter le risque d'explosion combinatoire. De plus, il est facile de se construire des bibliothèques d'ElPO simples et en interaction. En d'autres termes un I_{ElPOj} et/ou un P_i doivent être vérifiés seulement une fois. Les résultats de la vérification peuvent ensuite être exploités par l'expert pour une nouvelle PO étudiée.

La solution proposée pour la vérification des interactions entre ElPO et avec le produit repose sur une méthode en 2 pas. Dans un premier temps, les contraintes mises en place lors d'une interaction entre ElPO sont vérifiées (figure 4.7). Puis, dans un second temps, les contraintes pour limiter les zones d'accès du produit sont vérifiées (figure 4.8). L'approche de vérification est modulaire, c'est-à-dire, que les interactions entre ElPO ou avec le produit sont vérifiées indépendamment les unes des autres.

Pour vérifier l'ensemble de contraintes défini pour chaque interaction (figure 4.7), un modèle de PO est construit uniquement à partir des ElPO mis en jeu dans l'interaction et en considérant le fonctionnement d'un API. La vérification des contraintes se fait simplement en assurant la disjonction entre les états définis comme interdits et les états atteignables dans le modèle de PO lorsque les contraintes sont appliquées.

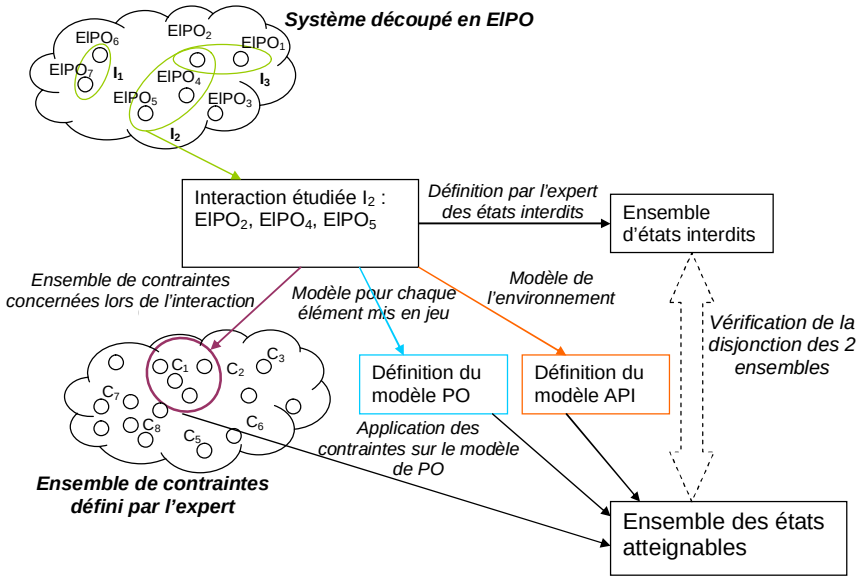


Figure 4.7. Idée pour valider les contraintes sur les interactions

Pour vérifier les contraintes mises en place pour un état de produit donné, la même idée est appliquée en utilisant en plus un modèle de produit (figure 4.8). Le principe consiste à associer aux ElPO concernés, la possibilité de recevoir ou/et de transférer à un autre ElPO, un produit. Si les ElPO ne sont pas correctement positionnés lors du transfert, le produit résultant est défaillant. La vérification des contraintes se fait en assurant que les conditions sont

respectées lors du passage d'un ElPO vers un autre. Si ce n'est pas le cas, le modèle produit sera conduit dans un état interdit.

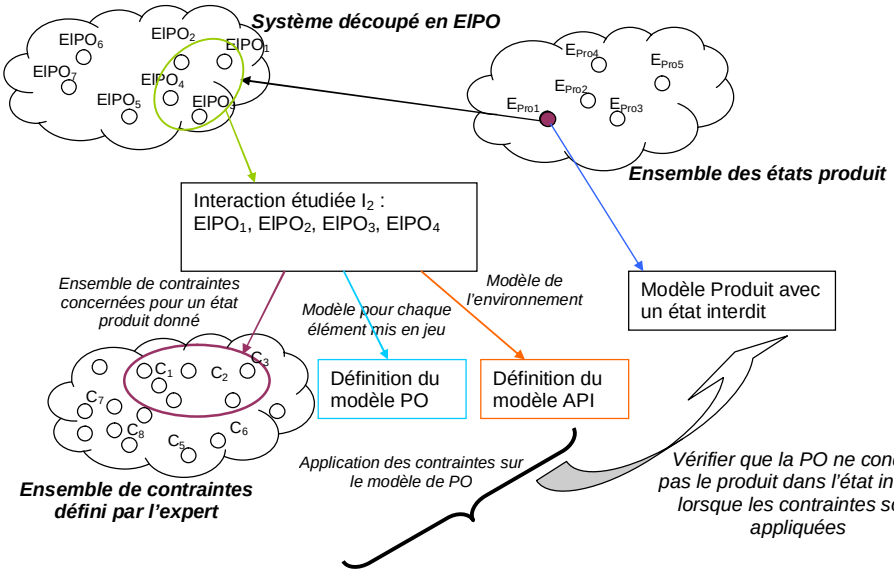


Figure 4.8. Idée pour valider les contraintes pour le produit

La vérification des contraintes s'effectue donc soit sur un modèle de PO, soit sur un modèle du produit. Le chapitre 3 a montré que l'écriture des contraintes est fortement dépendante de l'instrumentation du système et qu'il est parfois nécessaire de reconstruire des informations et des états de la PO. Les modèles de PO doivent être en adéquation avec l'instrumentation existante sur la PO. Enfin, en complément, la commande et les contraintes vont être programmées dans un API dont le fonctionnement séquentiel et cyclique doit être aussi pris en compte.

Le paragraphe suivant aborde la modélisation de la PO, des contraintes, des commandes et de l'environnement de calcul induit par la présence de l'API. Une approche originale est proposée et repose sur une vision causale de la chaîne fonctionnelle du point de vue de l'API.

3 Modélisation du système

La vérification des contraintes s'effectue sur un modèle du système (PO, PC, produit), un modèle de l'environnement de calcul (i.e. l'API), ainsi que sur le modèle de contraintes à respecter. Dans cette partie, nous allons présenter l'approche et les modèles utilisés. Quel que

soit le domaine, à tous les niveaux de description, il existe une distance entre le modèle et le système réel, et le point de vue de l'observateur est fondamental. Il est nécessaire de connaître cette distance, de la maîtriser en vue de la réduire au maximum. Dans le cas des SED, le modèle du système (PO + PC) peut être vu soit de la PO, soit de l'API. Dans le premier cas, on considère souvent l'hypothèse d'asynchronisme (durée nulle et non simultanéité des événements). Cette approche n'est pas envisageable dans notre cas, étant donné que les contraintes doivent être implémentées dans un API. Une approche est proposée pour modéliser le système avec un point de vue API en se basant sur la chaîne fonctionnelle (figure 4.9.).

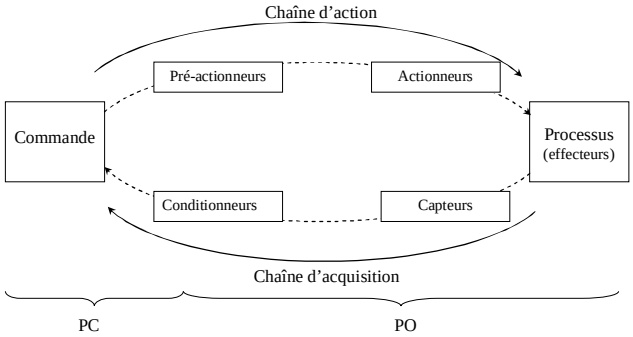


Figure 4.9. Chaîne fonctionnelle d'un processus

La chaîne fonctionnelle (Bodart et al., 2000) est l'ensemble des constituants, regroupant les ElPO et la PC, qui contribuent à la réalisation d'une fonction opérative. Elle a pour rôle d'envoyer des ordres à la PO, à partir de la PC (chaîne d'action) et de retranscrire la réaction de la PO, à la PC (chaîne d'acquisition).

Dans l'approche hors ligne présentée dans le chapitre 2, nous avons utilisé une modélisation structurée à base de règles d'occurrence et de relations de précédence. Cette modélisation a l'avantage d'être méthodique dans sa construction, ce qui diminue le côté intuitif de certaines constructions de PO. En revanche, comme nous l'avons vu, elle ne tient pas compte de la technologie des ElPO, de l'existence du produit au sein du système, et du fonctionnement de l'API. Dans le cadre de ce travail sur la vérification des contraintes, nous désirons être au plus près du système réel.

3.1 Préliminaires, hypothèses et outil de modélisation

Un modèle est toujours construit en fonction de l'utilisation qui en est faite. Dans le cadre de cette thèse, le système est piloté par un API. Le modèle doit donc tenir compte du

fonctionnement séquentiel et cyclique de l'API. Dans ces travaux, il a été fait l'hypothèse que le temps de cycle de l'API permet de détecter tous les changements d'état de capteurs et tous les mouvements des actionneurs. En revanche, aucune hypothèse n'a été faite sur la commande. L'approche proposée se restreint aux ElPO manufacturiers : vérins, tapis, moteurs.

Compte-tenu du domaine d'application (validation de commande implantée dans un API), en vue de minimiser la distance entre le modèle et le système réel (API + PO), il nous semble nécessaire que le modèle du système prenne en compte le retard de causalité qui comprend :

- L'effet induit par l'évolution d'une entrée (capteur) sur une sortie (actionneur) ne se fait pas avec un retard nul,
- La PO doit continuer d'évoluer pendant le cycle de l'API,
- L'actionneur et l'effecteur ont des éventuels retards liés à des inerties.

L'approche proposée, appelée « Commande, Sens, Position », notée C.S.P. (figure 4.10), suit la chaîne fonctionnelle (figure 4.9.), et considère que la PC génère des commandes (modèle de commandes : C) et que celles-ci vont éventuellement être à l'origine du déplacement (modèle de sens : S) des effecteurs et/ou des produits. Les effecteurs et le produit, dans cette approche, sont caractérisés par leur position (modèle de positions : P et modèle de produit). À cela, il faut ajouter un modèle représentant le fonctionnement cyclique de l'API (modèle d'environnement).

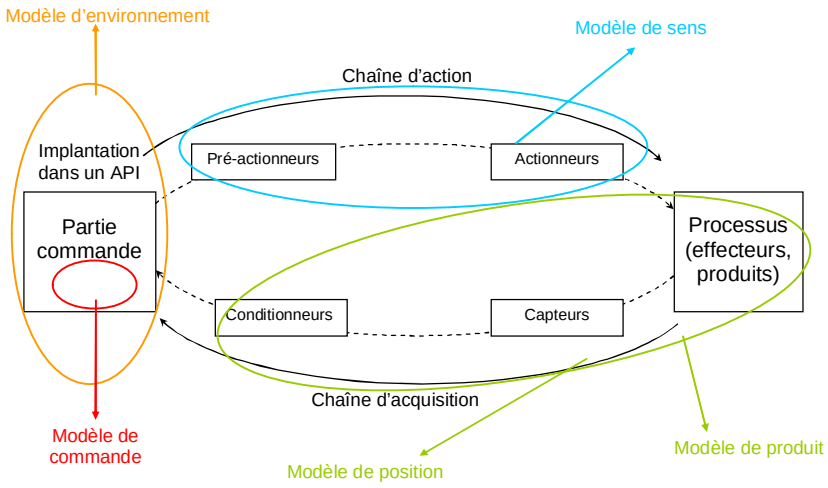


Figure 4.10. Modèle (C.S.P.) du système à partir de la chaîne fonctionnelle

La modélisation modulaire du système proposée, comme nous l'avons vu, permet de

diminuer les risques d'explosion combinatoire. C'est aussi dans cet objectif, comme dans le chapitre 3, que nous avons choisi d'utiliser des automates communicants comme outil de modélisation. En effet, le concept d'automate communicant facilite l'expressivité d'un produit synchronisé d'automates. Comme le souligne Julliand (2003), la notion d'automates communicants est liée à la décomposition du modèle global en plusieurs automates pour simplifier la conception. De plus, le fait de décomposer nécessite de pouvoir recomposer par produit synchronisé. Les automates communicants sont un moyen d'expression implicite d'un produit synchronisé.

Les différents modèles, représentés au moyen d'automates communicants interagissent entre eux, par échange d'informations. Cet échange d'informations se fait par l'intermédiaire de synchronisations ou de mises à jour des variables, dans un automate communicant.

Dans nos modèles, la synchronisation se fait de deux manières : soit par un échange de messages, soit par la mise à jour de variables. Des exemples théoriques d'automates à états, et d'automates communicants sont détaillés en Annexe 2 pour montrer la différence entre ces deux modèles.

Dans l'approche proposée pour le modèle C.S.P., la communication se fait en suivant la chaîne causale d'événements. Le modèle de commandes informe le modèle de sens sur l'évolution de la commande. Le modèle de positions évolue en fonction des informations envoyées par le modèle de sens. Enfin, l'évolution du produit va dépendre des informations des modèles de positions. Dans les paragraphes suivants, les différents modèles vont être présentés, ainsi que les échanges d'informations.

3.2 Modèle de l'environnement de calcul

Il est nécessaire de prendre en compte les contraintes liées au fonctionnement séquentiel et cyclique de l'API. Cela impose entre autres de gérer la possibilité de simultanéité d'événements pendant un cycle d'automate.

Le cycle de fonctionnement de l'API, compte tenu de la présence du filtre de commande contenant les contraintes (cf. chapitre 3), est le suivant :

1. Lecture des entrées,
2. Exécution du programme décrit par le concepteur,
3. Filtre de commande (liste de contraintes),

4. Écriture des sorties.

Le filtre de commande contient les contraintes statiques et dynamiques et nécessite éventuellement la mise en place d'estimateurs/reconstructeurs d'état si l'instrumentation n'est pas suffisante (cf. chapitre 3).

L'approche de vérification proposée repose sur la vérification des contraintes spécifiques à chaque sous-ensemble d'ElPO et de produit en interaction. L'idée consiste à envisager la commande la plus permissive possible et à vérifier qu'aucun état interdit n'est atteint dans les modèles de PO et de produit, compte-tenu de la présence du filtre de commande et de ses contraintes.

Afin de prendre en compte correctement l'environnement de calcul dans la phase de vérification, la figure 4.11 donne le chronogramme de deux entrées et de deux sorties au niveau de l'API (E_{API1}, E_{API2}, S_{API2}, S_{API2}) et au niveau de la PO (E_{PO1}, E_{PO2}, S_{PO2}, S_{PO2}). Chaque cycle (Cyc_i) se déroule comme expliqué précédement (Lecture, évolution de la commande, Contraintes, Écriture). Lors de la phase de lecture, les valeurs des entrées sont recopiées dans l'API et restent constantes pendant le reste du cycle (❶). Bien qu'il n'y ait pas d'évolutions simultanées au niveau de la PO, l'API les voit comme simultanées. Il est à constater que la PO continue d'évoluer en parallèle du cycle normal. Au niveau des sorties de l'API, elles évoluent seulement pendant le laps de temps « évolution de la commande ». Pendant cette période, les valeurs des commandes peuvent être changées plusieurs fois. Cependant, seule la dernière valeur sera utilisée (❷) pour le calcul des contraintes. Les sorties sont mises à jour lors de l'étape d'écriture. Si les contraintes ne sont pas toutes respectées, le système doit être arrêté (❸).

Pour prendre en compte ces observations, la modélisation du cycle automate se fait en 6 étapes séquentielles et cycliques :

1. Lecture des entrées,
2. Simulation de la PO,
3. Commande la plus permissive,
4. Filtre de commande (liste de contraintes),
5. Écriture des sorties,
6. Vérification des états interdits.

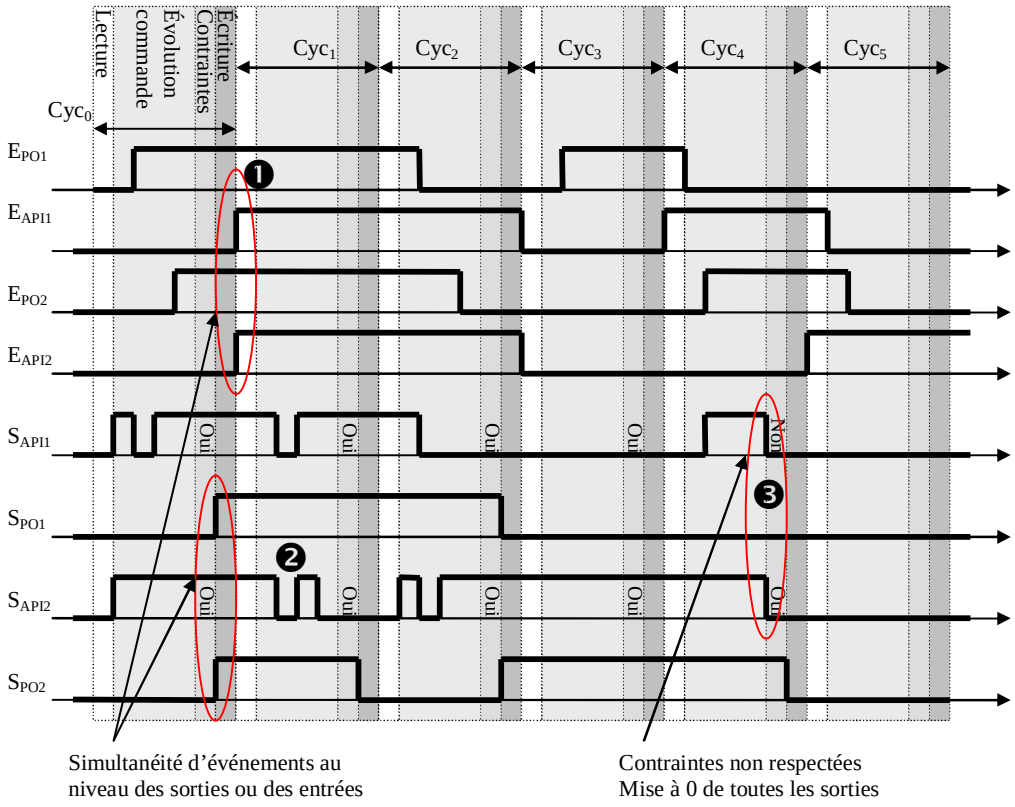


Figure 4.11. Chronogramme d'évolution API PO

Le calcul de l'évolution de la PO se fait juste après la lecture des entrées et avant la commande (figure 4.12.). Cela permet de différencier clairement les valeurs utilisées dans l'API de celles (réelles) de la PO. Ainsi, le calcul de la nouvelle situation de la PO se fait bien à partir de l'état de la commande au cycle précédent, et la PO peut évoluer pendant le cycle de l'API. Les conséquences de cette évolution de la PO seront prises en compte au cycle suivant lors de l'exécution de la nouvelle étape 1. En d'autres termes, pendant un cycle de l'API, la PO peut en théorie atteindre un état interdit. La figure 4.12. présente le modèle d'exécution qui est en fait une séquence permettant de simuler conjointement l'exécution de l'API, l'évolution de la PO et de vérifier si des états interdits sont atteints.

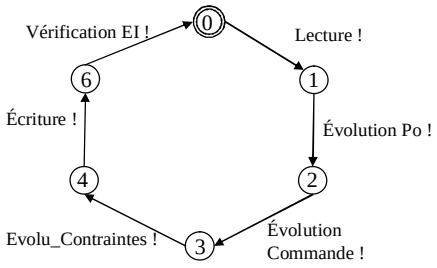


Figure 4.12. Modèle de l'environnement de calcul

Le modèle d'environnement va permettre d'orchestrer l'évolution des autres modèles. En effet, les 6 étapes vont déclencher séquentiellement le modèle de lecture, le modèle de PO, le modèle de commandes, le modèle de contraintes, le modèle d'écriture et enfin le modèle de vérification. Le modèle d'environnement a un rôle de chef d'orchestre entre les différents modèles, c'est-à-dire que c'est lui, qui indiquent aux autres modèles quand ils doivent évoluer (figure 4.13.).

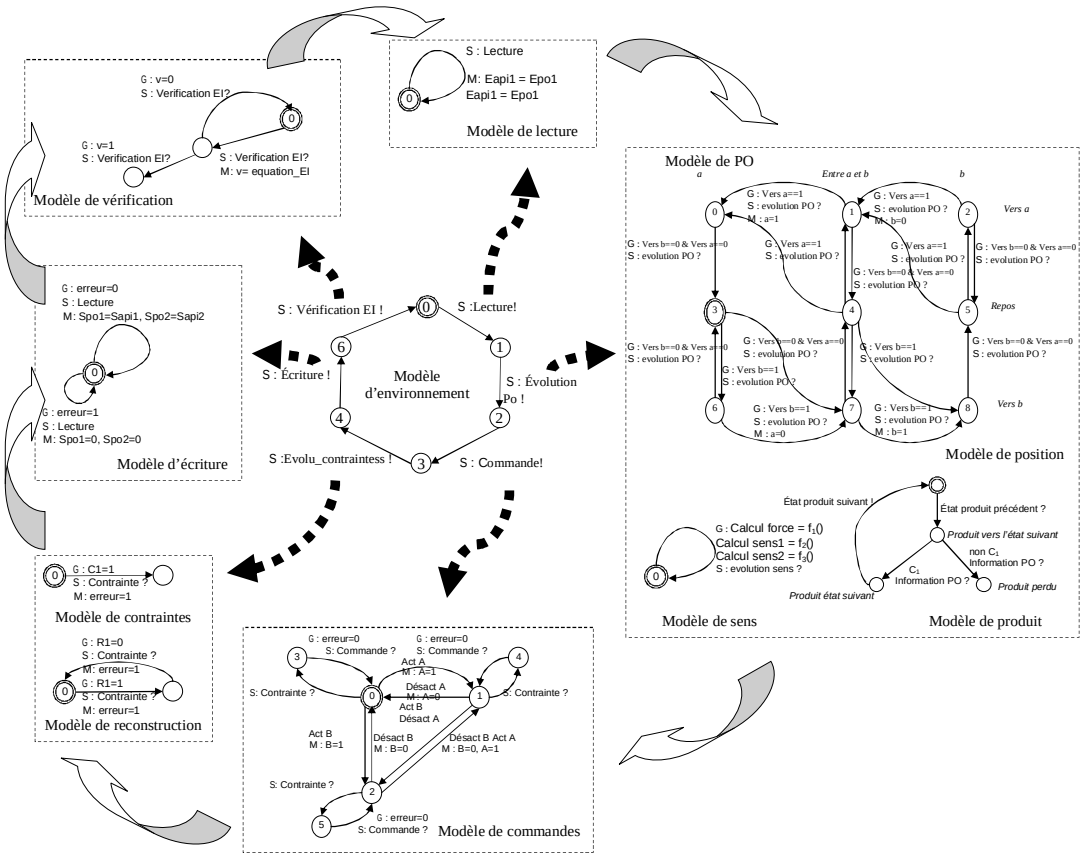


Figure 4.13. Appel des différents modèles par le modèle d'environnement

Le modèle de PO est défini par un sens de déplacement, une position actionneur et une position produit. L'évolution de ces trois modèles doit apparaître dans le modèle d'environnement (figure 4.14.). Il faut d'abord calculer le sens de déplacement qui influe sur la position de l'actionneur, qui lui-même influe sur la position du produit.

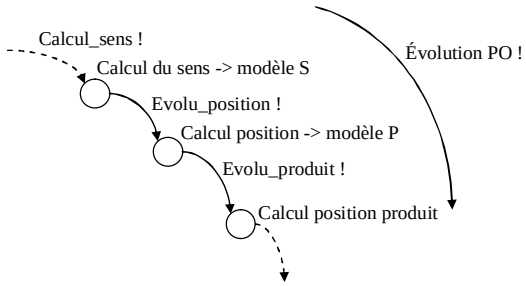


Figure 4.14. Détails de l'évolution de la PO

Le calcul des contraintes peut demander des informations supplémentaires à celles fournies par la lecture de l'état de la PO. Il faut donc ajouter une étape pour éventuellement reconstruire les informations avant l'exécution des contraintes.

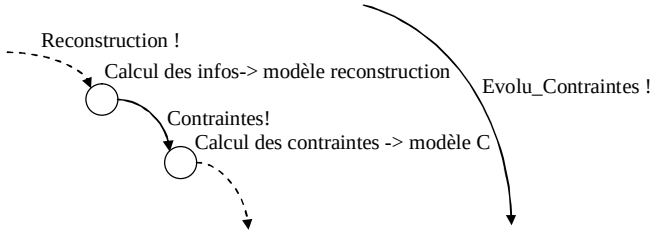


Figure 4.15. Détails de l'évolution des contraintes

Le modèle complet de l'environnement de calcul, est donc la succession de 9 étapes (figure 4.16.).

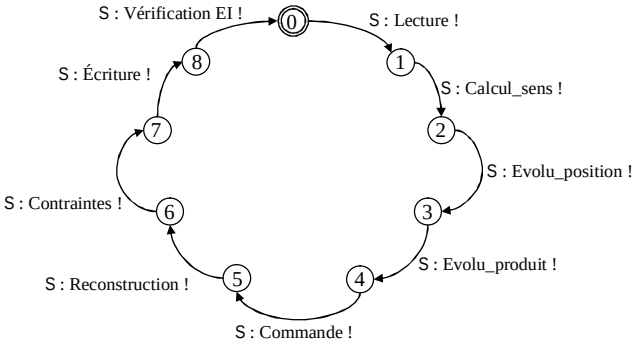


Figure 4.16. Modèle complet de l'environnement de calcul

La vérification de l'atteignabilité des états interdits se fait au niveau des positions prises par les ElPO et le produit. Lorsque les contraintes sont implémentées dans l'API, elles sont calculées à partir des informations contenues dans l'API. La figure 4.17. indique les variables utilisées (cases grisés), soit celles contenues dans l'API, soit celles du système, ainsi que les échanges d'informations. Par exemple, les valeurs utilisées dans les modèles de contraintes

sont des variables API provenant d'une part du modèle de lecture, des modèles de reconstruction d'information et des modèles de commande.

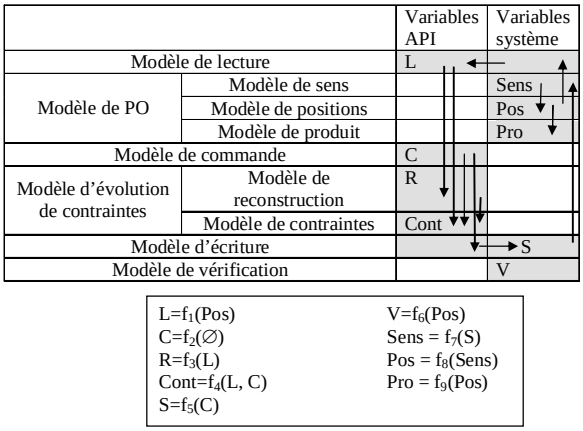


Figure 4.17. Informations utilisées dans chaque modèle

3.3 Modèle de commandes

Le modèle de commandes représente les évolutions qui peuvent être envoyées vers l'ElPO. Le modèle global de la commande est défini par un automate communicant, noté G_{gcom}, où les états sont définis par la valeur booléenne que peut prendre chaque ordre, les transitions représentent les évolutions envoyées par la commande du concepteur, c'est-à-dire : l'activation ou la désactivation d'un ordre. Pour un ElPO donné, le nombre d'état est égal à $2*2^{no}$, no étant le nombre de commandes, avec 2^{no} états pour empêcher l'évolution de la commande hors de l'intervalle de temps autorisé et 2^{no} états où la commande peut évoluer, et le nombre de transitions est égal à $(2^{no}-1)*2^{no}+ 2*2^{no}$, car pour chaque état, les commandes peuvent être soit activées, soit désactivées et il peut y avoir des combinaisons d'évolutions simultanées ($(2^{no}-1)*2^{no}$). A cela, il faut ajouter les 2 transitions par état ($2*2^{no}$) pour passer dans un état d'attente et mémorisant la commande précédemment envoyée. Les évolutions au niveau de la commande ne peuvent se faire que pendant l'intervalle de temps d'évolution de la commande et si le filtre de contraintes n'indique pas d'erreur. Les messages « *Commande !* » et « *Contraintes !* » provenant du modèle d'environnement (figure 4.12.) définissent l'intervalle de temps où la commande peut être modifiée. Durant le reste du cycle API, la commande n'est pas autorisée à évoluer. Le modèle de sens est informé de la valeur de la commande appliquée à l'ElPO à la fin de l'intervalle d'évolution de la commande. Il faut aussi interdire à la commande d'évoluer si une contrainte a été violée. Pour cela, une variable *erreur* globale a été définie. Lorsqu'une contrainte n'est pas respectée, *erreur* est

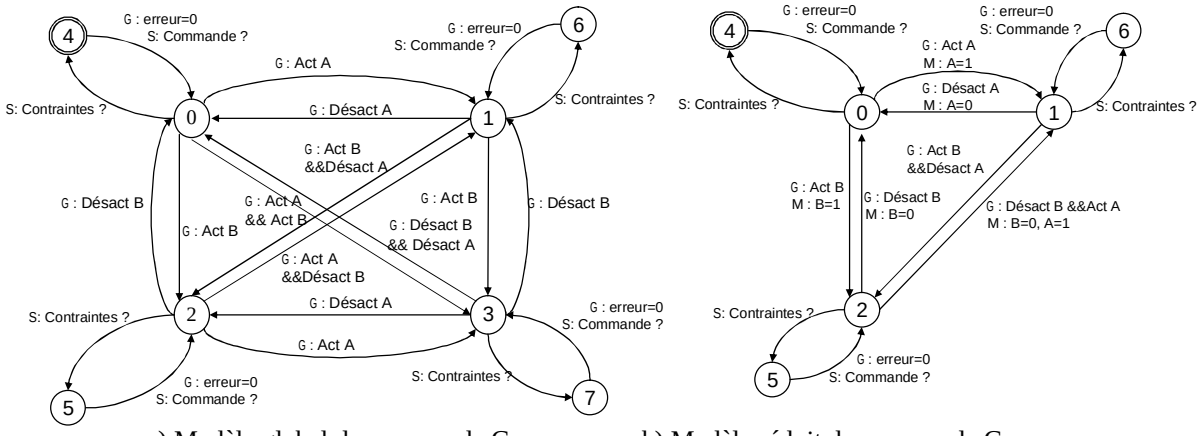
mise à 1, permettant ainsi d'empêcher définitivement toute évolution de la commande.

Si nous considérons un élément bistable décrit par les commandes *A* et *B*. Le modèle de commandes G_{gcom} (figure 4.18.a) est constitué de $2*2^2$ états qui représentent les situations où aucune commande n'est envoyée (état 0, 4), la commande *A* est envoyée uniquement (état 1, 6), la commande *B* est envoyée uniquement (état 2, 5), et les deux commandes sont envoyées en même temps (état 3, 7). Il y a $(2^2-1)*2^2 = 12$ transitions représentant les évolutions possibles en fonction de l'état dans lequel l'automate se trouve, plus 2^2 permettant l'activation et la désactivation de la commande.

Lorsque la commande peut évoluer, l'évolution d'un état à un autre se fait :

- Soit sous l'occurrence d'un seul événement commandable, c'est le cas pour le passage de l'état 0 à l'état 1. Suite à l'activation de la commande A, noté *Act A*, l'automate passe dans l'état 1. Il en est de même, pour le passage de l'état 1 à l'état 0, suite à la désactivation de *A*, noté *Désact A*

- Soit sous l'occurrence de deux événements simultanés, c'est le cas pour le passage de l'état 0, où sur l'occurrence de *Act A* && *Act B*[4], l'automate passe directement dans l'état 3. En effet, les évolutions de la commande se font à partir de l'API, il est donc possible d'avoir des évolutions simultanées de la commande dans le modèle.



a) Modèle global de commande G_{gcom} b) Modèle réduit de commande G_{com}

Figure 4.18. Modèles de commande G_{gcom} et G_{com}

[4] Les symboles booléens utilisés sous UPPAAL sont ceux du langage de programmation (!, && ; ||)

Comme nous l'avons présenté dans le chapitre 3, des contraintes de sécurité sont appliquées sur les ElPO, indépendamment du système dans lequel ils se trouvent. Elles sont de deux types : statiques et dynamiques. Les premières permettent d'interdire l'envoi simultané de commandes contradictoires sur un même ElPO. Par l'intermédiaire de cette contrainte statique, le modèle de commandes est simplifié en retirant les états, où les 2 commandes sont actives.

Sur l'exemple du vérin, la contrainte de sécurité statique qui interdit l'envoi des deux commandes au même instant, permet de ne plus atteindre l'état 3. Cette première contrainte s'écrit dans l'algèbre de Boole : $A \ \&\& \ B = 0$. Elle interdit d'une part l'envoi de *Act A* si la commande *B* est déjà active (réciproquement pour l'envoi *Act B* si la commande *A* est active) et d'autre part l'envoi simultané de *Act A* et *Act B*. Lorsque la contrainte est appliquée, l'état 3 n'est plus atteignable. Sous la condition d'appliquer la contrainte $A \ \&\& \ B = 0$, le modèle de commandes peut être simplifié et réduit à 3 états (figure 4.18.b).

Dans l'approche proposée C.S.P., le modèle de commandes permet éventuellement le déplacement d'effecteurs (et/ou de produits) qui sont caractérisés par leur position. Le paragraphe suivant aborde ces aspects au travers la modélisation de la PO.

3.4 Modèle de PO

Dans l'approche C.S.P., l'idée consiste à séparer les ordres de commande de la modélisation des ElPO. En d'autres termes, les commandes en provenance de l'API vont éventuellement induire un déplacement des effecteurs et des produits que l'on caractérise par leur position. Pour représenter l'évolution de la PO, trois modèles sont définis :

- Un modèle de sens : permettant de calculer la force appliquée sur l'actionneur et ainsi en déduire le sens de déplacement,
- Un modèle de positions : permettant de générer les états des capteurs du système,
- Un modèle de produit : permettant de connaître l'état du produit au cours de son évolution dans le système.

Cette méthode permet de tenir compte de la technologie des actionneurs sans avoir à modifier le modèle de positions de chaque ElPO.

La figure 4.14. indique les messages envoyés aux modèles de sens et de positions pour les ElPO et au modèle produit, qui sont détaillés dans les paragraphes suivants.

3.4.1 Modèle de sens

A partir du modèle de commandes et de la technologie de l'ElPO, le modèle de sens est défini. Ce modèle permet de représenter ce qui se passe physiquement sur la PO. En effet, lorsque l'actionneur reçoit une commande du pré-actionneur, il transforme cette information en énergie physique : force. Cette force physique va provoquer ou non un déplacement de l'ElPO. Cette information ne peut prendre que 3 états différents : soit la force est suffisante pour provoquer un mouvement dans le sens positif, soit elle est suffisante pour provoquer un mouvement négatif, soit elle est insuffisante pour provoquer un mouvement (situation de repos). Par la mise à jour des variables du modèle de commandes, l'intensité de la force est calculée et comparée avec des valeurs seuils pour passer d'un état à un autre, lorsque le modèle de sens reçoit le message « *Calcul_sens* » du modèle d'environnement. Cette modélisation suppose, comme cela a été énoncé dans les hypothèses de départ, que l'API peut être capable de détecter les changements de sens des effecteurs.

Si nous considérons un élément bistable recevant deux commandes *A* et *B*, le calcul de la force appliquée va dépendre de la technologie de l'ElPO. Dans le cas, d'un vérin piloté par un distributeur 5/3 centre ouvert, par un distributeur 5/3 centre fermé, ou d'un chariot ou d'un tapis, lorsqu'il n'y a pas de commande envoyée, l'élément retourne vers un état de repos (figure 4.19.a).

Par contre dans le cas, d'un vérin piloté par un distributeur 3/2 monostable, par un distributeur 5/2 monostable, par un distributeur 5/2 bistable, la position de repos n'est atteinte que lorsque les commandes sont inversées (figure 4.19.b). Le modèle de sens proposé tient compte de ces comportements et permet aussi de mettre en évidence que le changement de sens impose de passer par un état de repos.

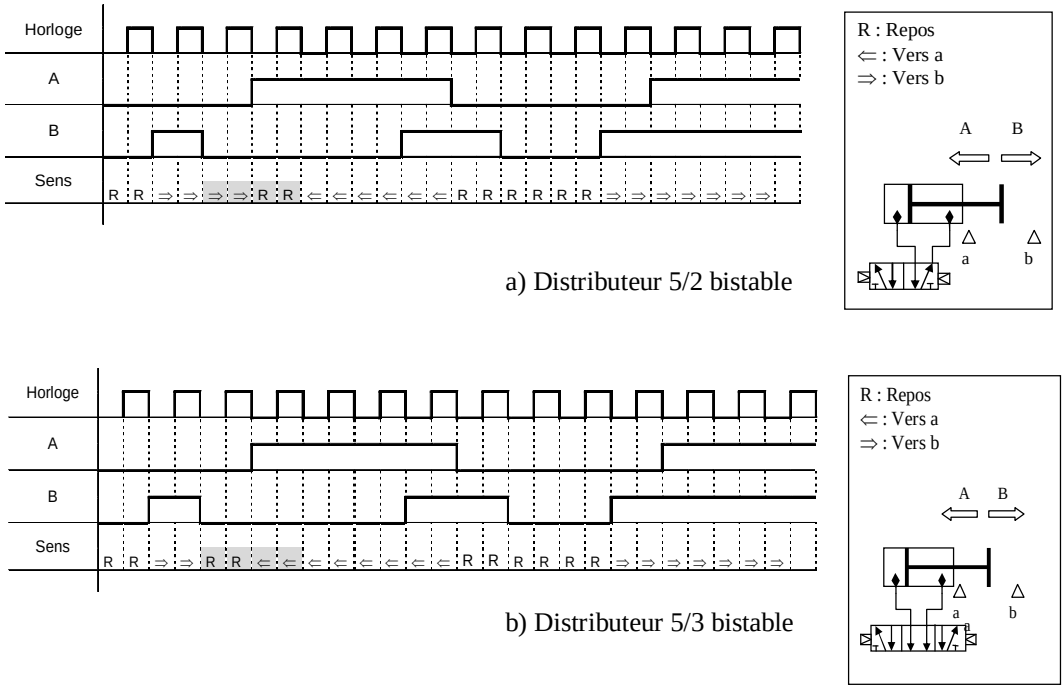


Figure 4.19. Chronogramme d'évolutions du sens

Le modèle sens permet aussi de modéliser le fonctionnement physique du système et ainsi de mettre en avant les problèmes d'inertie. Le choix des paramètres du modèle de sens dépend de ce que l'on désire modéliser et vérifier (PO sans inertie, avec peu ou beaucoup d'inertie).

Ce modèle de sens utilise un automate communicant à un seul état et les possibilités de programmation offertes par le logiciel UPPAAL. En effet, il est possible de coder des lignes de programme (figure 4.20). Pour cela, trois fonctions ont été mises en place :

- Calculer la force appliquée à l'ElPO
- Déterminer la valeur du sens positif
- Déterminer la valeur du sens négatif.

Dans le cas d'un vérin commandé par un distributeur 5/2, le programme de la figure 4.20 est proposé. Lorsque la commande *A* est envoyée, la force *F* est incrémentée d'une constante *f* dont la valeur permet de moduler la vitesse d'incrémentation de la force. Par contre si la commande *B* est activée, la force est décrémentée de *f*. Si aucune commande n'est appliquée, la force *F* n'est pas modifiée. La force *F* est bornée par une valeur maximum *fmax*. En fonction de la valeur de la force *F*, les fonctions sensp et sensm vont déterminer le sens de déplacement. Si *F* est supérieur à la valeur seuil *frp*, l'ElPO se déplacent dans le sens positif (*Sp*=1). Si *F* est inférieur à la valeur seuil *frm*, l'ElPO se déplacent dans le sens négatif

(*Sm*=1). Sinon, l'ElPO n'est pas en mouvement.

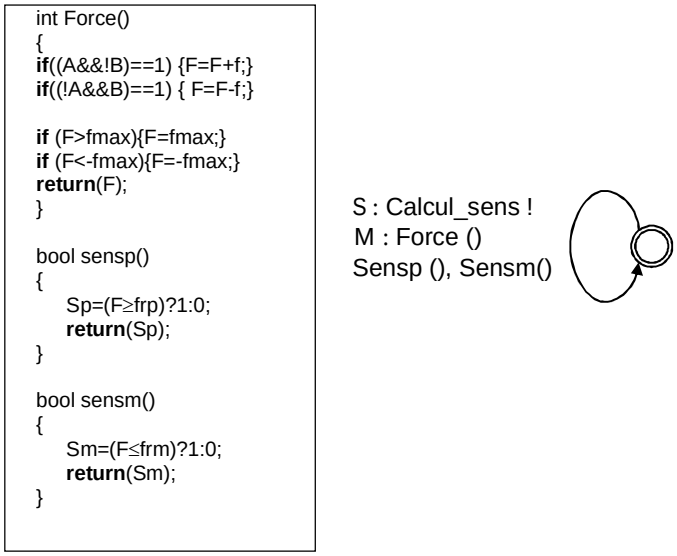


Figure 4.20. Implémentation du modèle sens

Si le système est muni d'un capteur de sens, l'API peut connaître directement à chaque cycle de l'API, la valeur du sens de déplacement. L'information du sens de déplacement va permettre de calculer les positions prises par l'ElPO.

3.4.2 Modèle de positions

Le modèle de positions G_{pos} représente les différentes positions des effecteurs et des produits et génère les différents états de capteur lorsque la PO subit un sens de déplacement. Le modèle de PO doit au minimum pouvoir générer les états de capteurs. Il est donc dépendant de l'instrumentation. Par exemple, si l'on considère un vérin dont la position est repérée au moyen de n_c capteurs distincts (cf. figure 4.21), le modèle global de position est défini par l'automate G_{gpos} à $3*(n_c+n_i)$ états où n_c représente le nombre de capteurs, n_i le nombre de zones intermédiaires entre les capteurs et 3 car il y a 3 états possibles du sens.

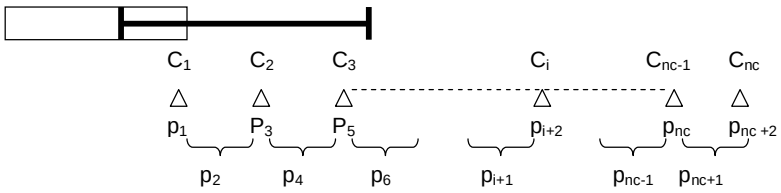


Figure 4.21. Exemple du vérin avec n_c capteurs de position

Dans le cas de l'exemple de la figure 4.21, n_i est égale à n_c-1. On en déduit que le

nombre d'états de l'automate global G_{gpos} est égal à $3*(2n_c-1)$.

Le modèle de positions G_{gpos} de l'ElPO est obtenu en utilisant seulement, les états du modèle de sens G_{sens}, les $(n_c + n_i)$ positions possibles de l'ElPO et les transitions physiquement possibles. En effet, il est possible de retirer les évolutions physiquement impossibles en faisant l'hypothèse que le système n'est pas défaillant. Par exemple, lorsqu'il y a un déplacement vers la droite, le capteur de fin de course gauche ne peut pas passer à 1. De même, lorsqu'il n'y a pas de déplacement, il ne peut y avoir un changement d'état des capteurs. Si les contraintes dynamiques sur un ElPO pris indépendamment sont appliquées, la commande d'aller vers un capteur sera interdite, si l'ElPO est déjà sur le capteur et il sera obligatoire de désactiver cette commande lorsque l'ElPO arrivera sur un capteur de fin de course. En prenant en compte ces impossibilités physiques, le modèle obtenu est noté G_{pos}.

Lorsque que l'ElPO est stimulé par une force de déplacement, il va quitter ou arriver sur un capteur. Ceci est représenté dans le modèle de PO par la mise à jour de variables capteur. L'outil pour modéliser G_{gpos} et G_{pos} est un automate à états communicants, permettant un échange d'informations avec le modèle d'environnement.

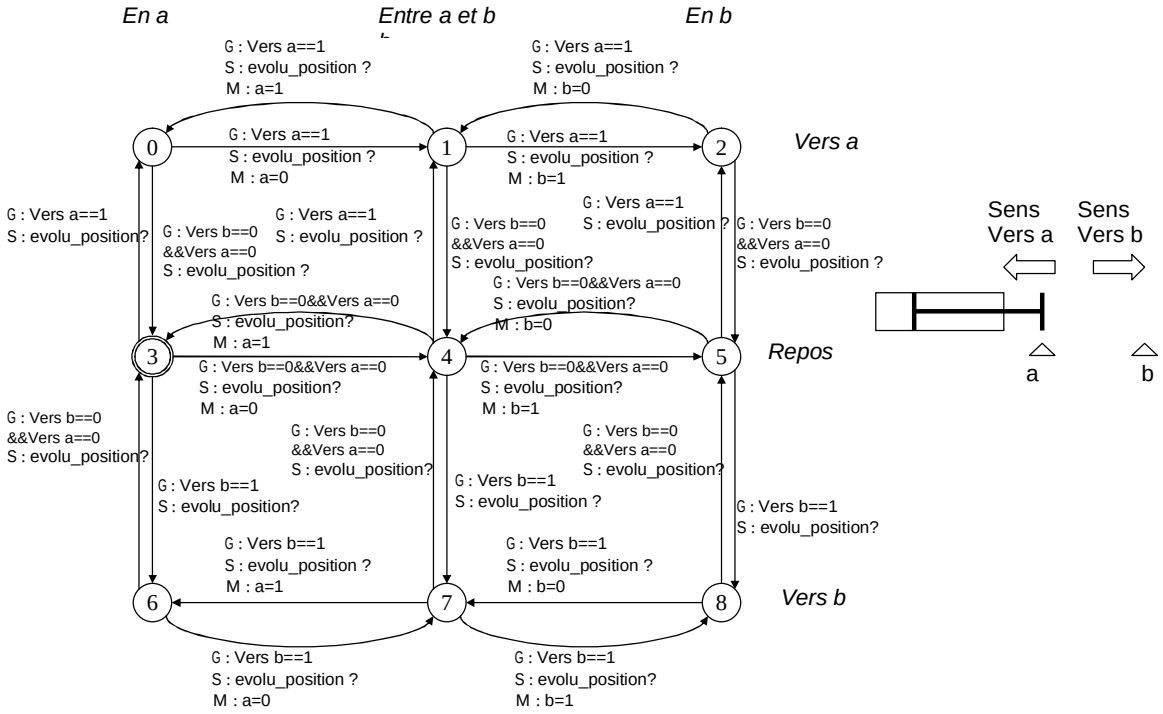


Figure 4.22. Modèle G_{gpos} pour un ElPO à 3 positions

Le modèle complet G_{gpos} d'un vérin à 2 capteurs (*a*, *b*) piloté par un distributeur 5/2 bistable est représenté figure 4.22. Il a 9 états. Le modèle G_{pos} est obtenu en retirant les

transitions physiquement impossibles. En effet, si le vérin se déplace dans le sens *vers b*, le capteur *a* ne peut pas passer à 1, de même le capteur *b* ne peut pas passer à 0. Les transitions 8 vers 7 et 7 vers 6 sont donc supprimées. Il en est de même lorsque le vérin se déplace dans le sens *vers a*. Les contraintes dynamiques interdisant d'atteindre les états où la commande associée à l'activation de la position de fin de course sont actives si l'ElPO est déjà en fin de course (cf. figure 4.23.). Cela permet de simplifier le modèle en retirant les transitions de l'état 3 à l'état 0 et de l'état 5 à l'état 8.

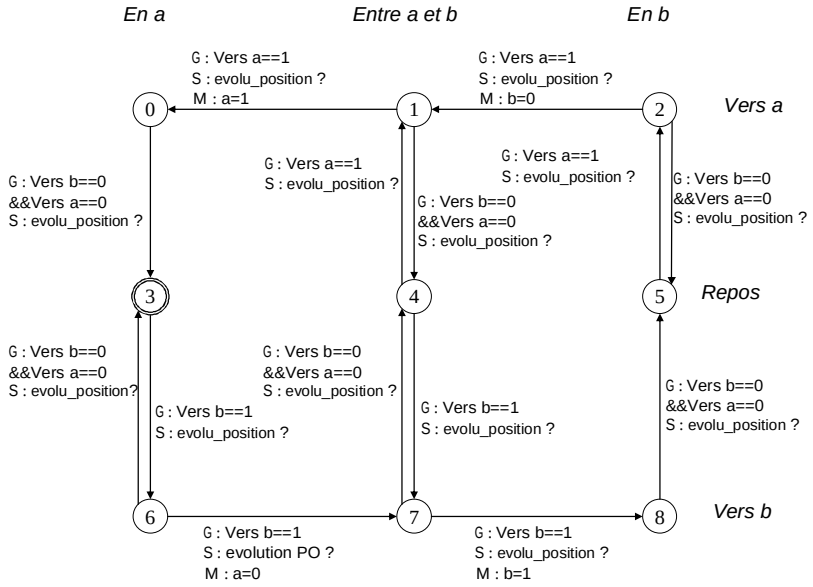


Figure 4.23. Modèle G_{pos} pour un ElPO à 3 positions sans les évolutions impossibles

Le modèle final de PO, permet de générer les informations relatives aux états des capteurs à partir de l'information sens. Dans cette structure, le passage d'un état à un autre dépend uniquement du sens. Cependant, les états durent, quand le système est en déplacement, seulement un cycle de l'API. Comme cela a été montré dans les exemples introductifs, cette solution n'est pas très réaliste. Il est nécessaire d'ajouter une information temporelle. En effet, physiquement, l'ElPO met plusieurs temps de cycle pour quitter un capteur, pour passer d'un capteur à un autre ou pour s'arrêter. Pour prendre en compte ce phénomène physique, une horloge *t* a été ajoutée au modèle et est mise à jour à chaque cycle API. En fonction du sens de déplacement de l'ElPO, l'horloge sera incrémentée ou décrémentée d'un pas δ. Nous avons adopté cette solution plutôt qu'une horloge continue car elle est correspond vraiment à la perception qu'a l'API de la PO.

Ainsi pour permettre le passage d'un état à l'autre, l'horloge *t* est comparée avec une valeur seuil, qui est égale à une valeur maximum dans un sens de déplacement et qui est égale

à 0 dans l'autre sens. Le temps passé sur un capteur est noté $t_{cap.}$ t_{inter} représente le temps nécessaire pour passer d'un capteur à un autre. De plus, il faut ajouter des transitions rebouclantes sur chaque état pour la mise à jour de l'horloge et ajouter un test supplémentaire au niveau de la garde pour passer d'un état à l'autre.

Dans l'exemple du vérin à 3 positions, la figure 4.24. représente l'automate à états complet, noté G_{gpos}. Au niveau des transitions verticales représentant uniquement un changement au niveau du sens (par exemple la transition de l'état 3 vers l'état 6), le temps passé dans cette position est incrémenté ou décrémenté d'un pas δ à condition que le temps soit inférieur strictement au seuil moins un pas. Il faut considérer un pas de moins car le franchissement de la transition représente le déplacement d'un pas δ. Par exemple de l'état 3, lorsque le sens *Vers b* devient vrai, il y a deux cas possibles soit t est inférieur $t_{cap} - \delta$ alors l'automate passe dans l'état 6, soit t est supérieur ou égal $t_{cap} - \delta$, alors il passe directement dans l'état 7. Lors du passage de la transition de l'état 3 vers 6, t est incrémenté d'un pas δ (M : t=t+δ), alors que lors du passage de l'état 3 vers l'état 7, t est mise à 0 et l'information sur le capteur a est générée , ici le capteur a est quitté (a =0).

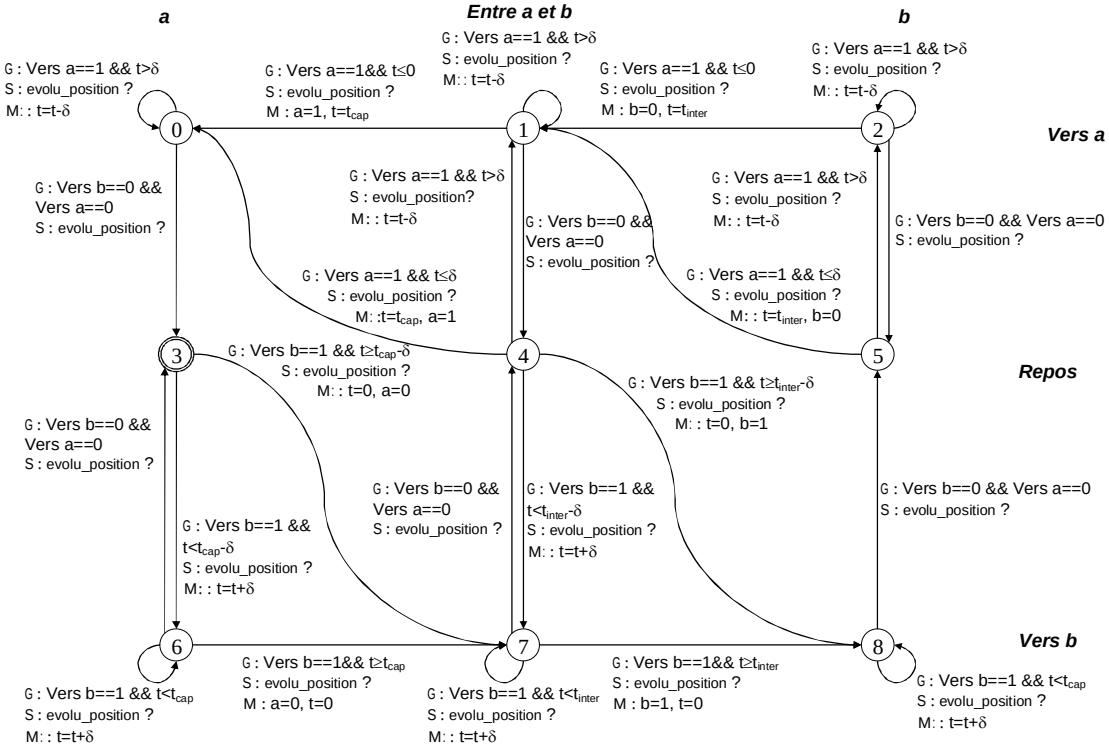


Figure 4.24. Modèle G_{gpos} du vérin à 3 positions

Sur le même exemple, si le vérin se déplace dans le sens *Vers a*, à partir de l'état 4, l'automate peut soit évoluer vers l'état 0 ou l'état 1, en fonction que *t* soit inférieur ou égal, ou strictement supérieur à δ. Dans le cas du passage de l'état 4 vers l'état 0, l'information du capteur *a* est généré à 1 et l'horloge *t* est mise à jour et est égale à t_{cap}.

Il est à noter que le modèle de positions reste le même pour l'ElPO indépendamment du distributeur. Le modèle d'évolution de la PO n'est donc pas complexifié par la prise en compte de différentes technologies.

La nécessité d'ajouter l'information du temps passé dans un état pendant le déplacement est indispensable si un capteur intermédiaire est utilisé. Pour l'exemple d'un vérin avec 3 capteurs de position (*a*, *i*, *b*), et son modèle G_{pos} à 15 états (cf. figure 4.25). Dans cette configuration, si l'information temporelle n'est pas prise en compte, le vérin ne pourra jamais s'arrêter sur la position *i*. En effet, l'ordre d'arrêt ne peut être donné que lorsque la position *i* est atteinte. Même sans inertie, le retard de causalité entraînera l'arrêt du vérin après la position repérée par *i*. Cela vient du fait que pendant le temps de cycle, où l'API génère l'ordre d'arrêter la commande après la lecture de *i*, la PO continue d'évoluer et quittera *i*. En vue de modéliser cet aspect, la possibilité de rester dans chaque état un certain temps est rajoutée dans le modèle final G_{pos}, (cf. figure 4.25), au moyen d'une horloge *t*.

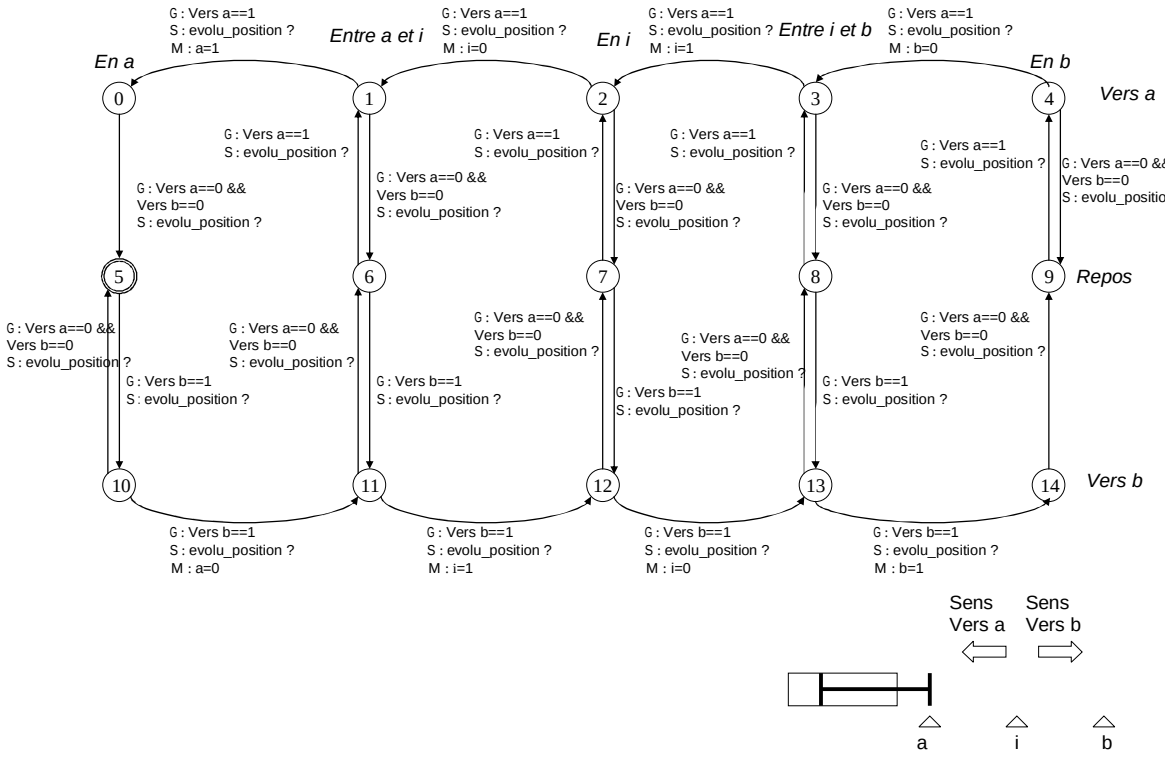


Figure 4.25. Modèle final G_{pos} pour le vérin à 3 capteurs

Par exemple, à partir de l'état initial 5 (figure 4.26.), suite à l'occurrence de l'information *vers b*, une horloge notée *t*, est lancée dans l'état 10 pour générer la désactivation de *a* lorsque *t* sera supérieur à t_{cap}. L'horloge *t* est incrémentée à chaque cycle d'automate et elle est remise à 0 à chaque changement d'état, c'est-à-dire lorsque *t* est supérieur à t_{cap} (passage dans l'état 11). Il en est de même pour représenter le temps de déplacement d'un capteur à l'autre, noté t_{inter}.

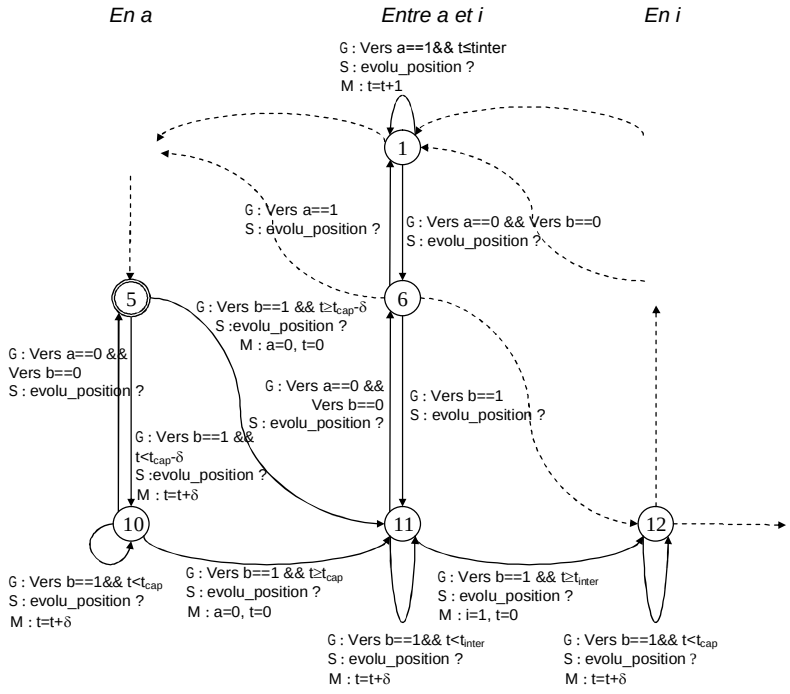


Figure 4.26. Vue détaillée de G_{gpos} avec prise en compte du temps

Le modèle global avec la prise en compte de l'information temporelle pour l'exemple d'un vérin avec 3 capteurs est donné dans la figure 4.27.

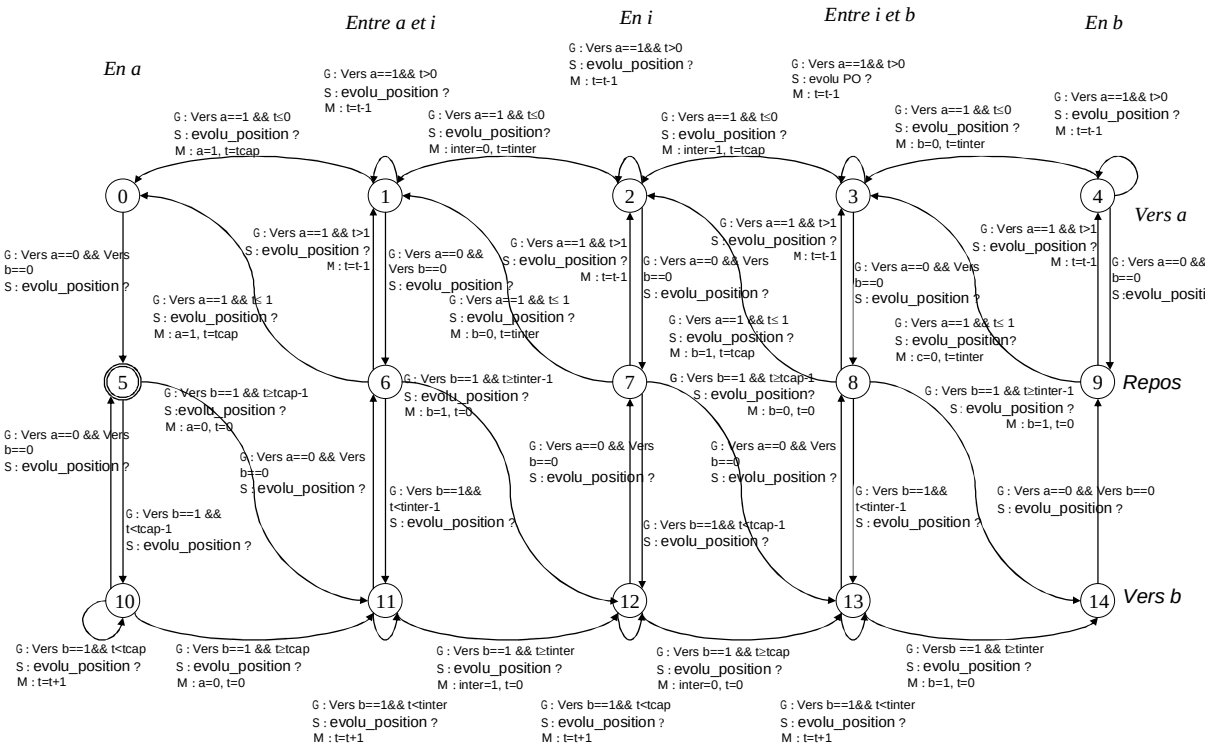


Figure 4.27. Modèle final G_{gpos} avec prise en compte du temps

Le modèle de PO ainsi défini va permettre de générer les différentes positions prises par la PO. L'ensemble des trois modèles : G_{com}, G_{sens} et G_{gpos} constituent le modèle G_{csp} de l'ElPO utilisé pour la vérification des contraintes entre ElPO.

Il est évident que le nombre de cycles de l'API pendant lesquels le capteur est actif et la prise en compte éventuelle de l'inertie modifient les résultats. Nous reviendrons sur cet aspect dans les perspectives de recherche.

Les problèmes de collision peuvent aussi provenir d'un mauvais positionnement du produit par les actionneurs. Dans le cadre de ce travail, nous avons pris en compte certaines interactions entre le produit et les ElPO. Ce travail est présenté dans le paragraphe suivant.

3.4.3 Modèle de produit

Le modèle de produit doit permettre de suivre l'état et les positions des pièces dans le système. Les ElPO ont pour objectif de transformer ou de déplacer des produits. Le produit est souvent le lien entre des ElPO (cf. figure 4.28.).

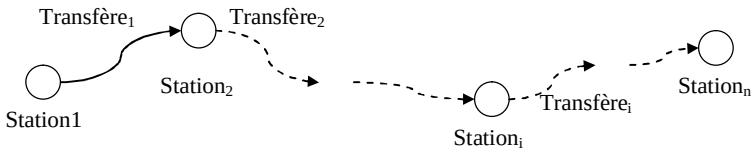


Figure 4.28. Lien procédé / produit

Nous avons uniquement considéré le cas d'un ensemble de stations en série sans stock intermédiaire où la station i transfère le produit vers la station i+1. Chaque station est caractérisée par un ou plusieurs ElPO en interaction. Le transfert peut se faire uniquement si les stations i et i+1 se trouvent dans des états (i.e. positions) particuliers et qu'une pièce est présente à la station i.

L'idée est de définir, pour chaque état transitoire du produit entre les stations i et i+1, un modèle communicant représentant le passage de la station i vers la station i+1. La station i est génératrice de la pièce. En revanche, la station i+1 est réceptrice de la pièce. Les modèles des ElPO doivent donc être enrichis en vue de générer les événements représentant la mise à disposition et la prise en compte de la pièce.

Cette solution évite l'explosion combinatoire et permet de conserver (et donc de ne pas modifier) la structure des modèles des ElPO. On obtient ainsi un automate $G_{produit}$ à 4 états (figure 4.29). La transition de l'état 0 vers l'état 1 pour l'état produit i, se fait par synchronisation avec l'état produit dans la station i-1. Deux évolutions sont alors possibles lorsque l'ElPO se trouve dans une position donnée, notée $Info_{po}$ et que le modèle reçoit le message « *Évolu_produit* » du modèle d'environnement :

- Soit les conditions requises notées $Cond_1$ pour passer dans l'état 2 du modèle de produit suivant (station i+1) sont présentes, dans ce cas le produit est transféré correctement dans la station suivante en actualisant le modèle de produit de la station i+1.
- Soit les conditions $Cond_1$ ne sont pas vraies, dans ce cas, le produit passe dans un état perdu (état 3).

C'est ce dernier état qui va permettre d'effectuer la vérification de l'ensemble de contraintes relatif au transfert du produit. En effet, si celles-ci sont correctement définies, l'état perdu (état 3) ne doit pas être atteint. Le retour vers l'état initial du modèle de produit se fait en envoyant un message qui est utilisé pour activer le modèle de produit de la station suivante. Pour que l'information soit immédiatement traitée, un état « committed » est défini

sous UPPAAL, noté par un C. Pour un état défini comme committed, les transitions de cet état sont traitées en priorité par rapport aux autres états.

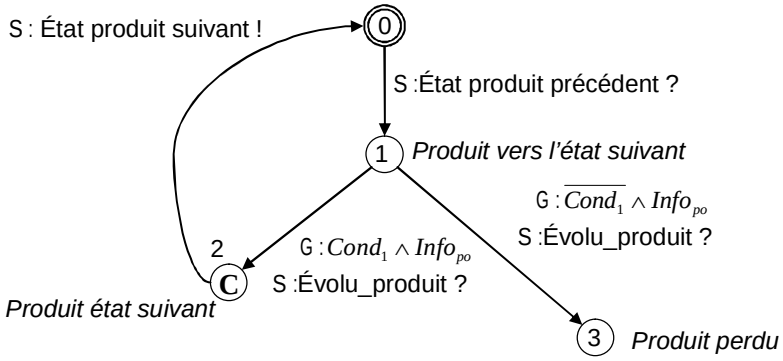
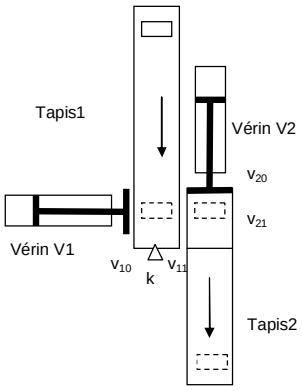


Figure 4.29. Automate pour modéliser le produit

Pour illustrer ce propos, l'exemple d'un système de transfert de pièces entre deux tapis (Tapis 1 et Tapis 2) au moyen de 2 vérins double effet 5/2 (V_1, V_2) (figure 4.30.) est traité.



	Variables système	Variables API
Tapis 1	T1	T1_auto
Vérin V1	$V1^+$, $V1^-$, v_{10}, v_{11}, k	$V1^+$_auto, $V1^-$_auto, v_{10}_auto, v_{11}_auto, k_auto
Vérin V2	$V2^+$, $V2^-$, v_{20}, v_{21}	$V2^+$_auto, $V2^-$_auto, v_{20}_auto, v_{21}_auto
Tapis 2	T2	T2_auto

Figure 4.30. Exemple du transfert de pièces

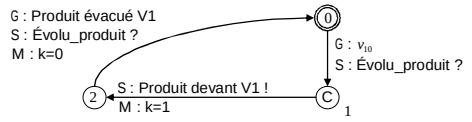
Afin de suivre l'état du produit dans le système, deux modèles de produit sont nécessaires: « *passage V_1 vers V_2* » (figure 4.31. b) et « *passage V_2 vers Tapis 2* » (figure 4.31. c). Le premier informe que le produit est passé de V_1 à V_2 et le second que le produit a été évacué sur le Tapis 2 par V_2.

Le passage de la pièce du vérin V_1 au vérin V_2 se fera lorsque le vérin V_1 arrivera en position sortie (v_{11}=1) et se passera correctement à condition que le vérin V_2 soit en position rentrée (v_{20}=1) et pas en train de se déplacer vers v_{21}. Pour cela, sur le modèle de PO du vérin V_1, lorsque le modèle passe dans l'état 8, il faut générer l'information du transfert de la pièce notée *Produit_evacuéeV1* qui repassera à 0 au cycle suivant (figure 4.32.). Pour être certain

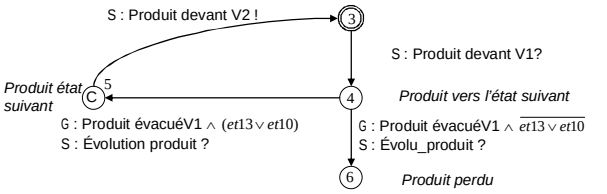
que le vérin V_2 est correctement positionné, le modèle de positions est enrichi pour informer qu'il se trouve dans l'état 10 ou 13 (figure 4.33.). Le modèle de produit représentant le passage du vérin V_1 vers le vérin V_2 est donné par la figure 4.31. b. Lorsqu'une pièce arrive devant V_1, l'automate passe dans l'état 3 à l'état 4 par le message « *Produit devant V_1* ». Ce message est généré par l'automate de la figure 4.31. a, à condition que le vérin V_1 soit rentré. Lorsque le vérin V_1 arrive dans l'état 8, il génère l'information *Produit_evacuéV1* qui fait évoluer le modèle de produit lors de la réception du message « *Evolu_produit* » :

- Soit vers l'état 5 si le modèle de positions de l'ElPO V_2 se trouve dans l'état 10 ou l'état 13, c'est-à-dire que le vérin V_2 doit être en position rentrée
- Soit vers l'état 6 si le modèle de positions de l'ElPO V_2 se trouve dans une autre position.

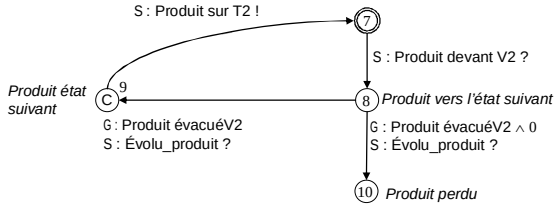
Le passage de la pièce du vérin V_2 au Tapis 2 se fait sous aucune condition. Le modèle est donné figure 4.31. c.



a) Génération de l'arrivé d'une pièce en k



b) Modèle produit passage de V_1 à V_2



c) Modèle produit passage de V_2 à Tapis2

Figure 4.31. Modèles des états produit du système de transfert

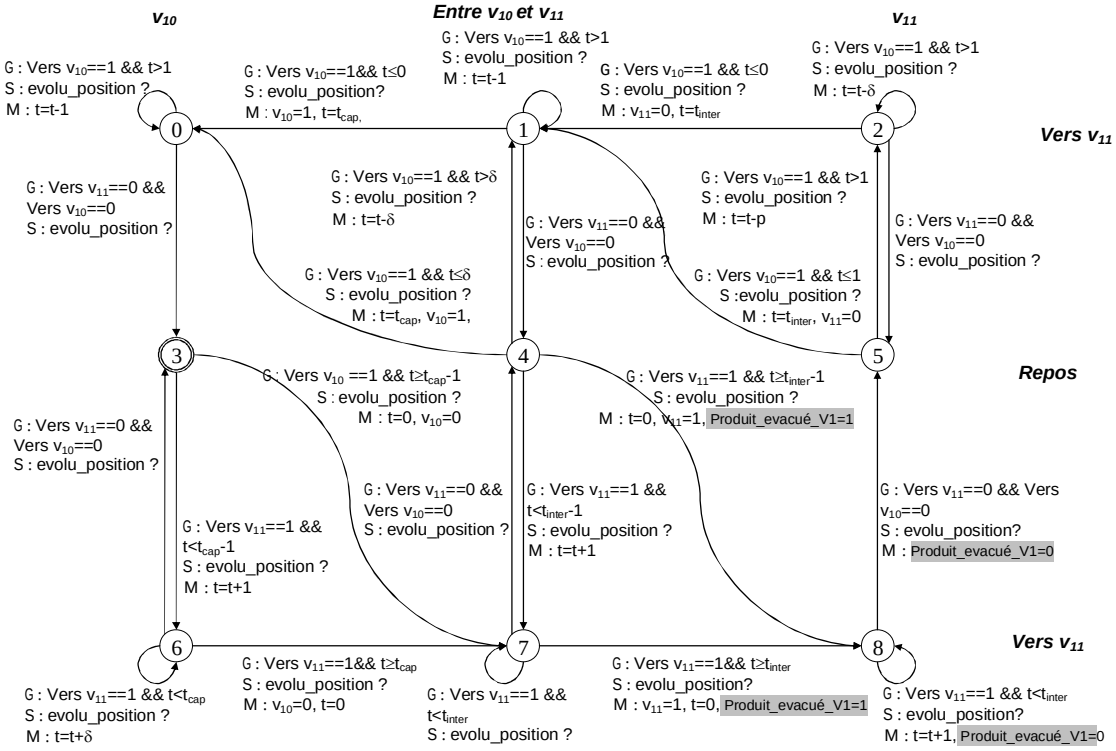


Figure 4.32. Modèles de positions du vérin V_1 du système de transfert

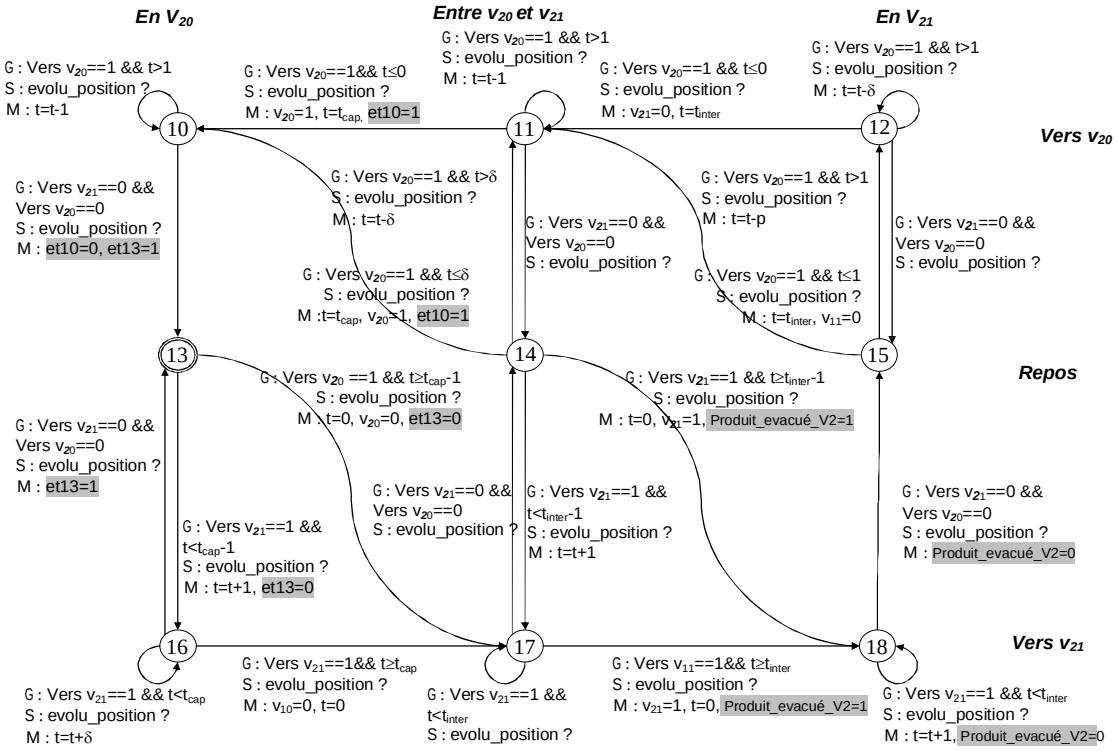


Figure 4.33. Modèles de positions du vérin V_2 du système de transfert

Le modèle de produit ainsi proposé permet de connaître l'état du produit sans pour autant complexifier lourdement le modèle de positions. C'est à l'expert de définir la précision du nombre d'états pris par le produit. Nous avons uniquement traité le cas où une seule pièce est transférée entre 2 stations. La gestion de stock ou de file d'attente n'a pas été traitée et fera l'objet de perspectives.

Cette troisième partie a permis de présenter les modèles et variables utilisés pour effectuer la vérification des contraintes. La partie suivante explique l'approche pour vérifier la suffisance des ensembles de contraintes dans le cas d'une interaction entre ElPO et dans le cas d'une interaction avec le produit.

4 Approche de vérification des contraintes

4.1 Mise à jour de la valeur des contraintes et des informations reconstruites

La vérification de chaque contrainte se fait lors de la réception du message « *Contraintes* ». La contrainte est calculée dans la garde G (figure 4.34). Si celle-ci est vraie, l'automate de contraintes passe dans l'état erreur et la variable globale *erreur* est mise à 1. La variable *erreur* est globale à tout le système et permet d'empêcher l'évolution de tous les modèles de commande lorsqu'elle est à 1.

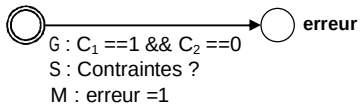


Figure 4.34. Modèle de contraintes

Les contraintes peuvent nécessiter des informations reconstruites (cf. chapitre 3) en complément des valeurs des capteurs et actionneurs. Pour cela, les modèles de reconstruction peuvent être définis.

Une fois que les contraintes ont été appliquées, lors de l'étape d'écriture des sorties, deux cas sont possibles. Soit toutes les contraintes sont respectées, la variable globale *erreur* est égale à 0 et les sorties de l'API peuvent être envoyées vers la PO (*UA=UA_auto*, *UB=UB_auto*). Soit il y a une ou plusieurs contraintes qui ne sont pas respectées, dans ce cas

la variable globale *erreur* est à 1, le système est arrêté en mettant à 0 toutes les sorties (*UA*=0, *UB=0*).

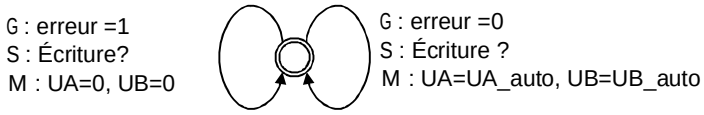


Figure 4.35. Modèle d'écriture des sorties

Si les contraintes sont correctement définies, aucun état interdit (ou dangereux) de la PO (actionneurs et pièces) ne doit pouvoir être atteint. Pour cela deux propriétés génériques sont vérifiées. La première porte sur une interaction entre ElPO et la seconde sur une interaction entre un ElPO et la pièce.

4.2 Vérification des contraintes pour les interactions entre ElPO

Le principe, comme nous l'avons vu, consiste à vérifier que les ElPO ne se retrouvent pas dans des positions interdites. Pour cela, l'étape 6 de vérification, présentée dans le paragraphe 2, est réalisée au moyen d'un automate à 3 états représenté figure 4.36

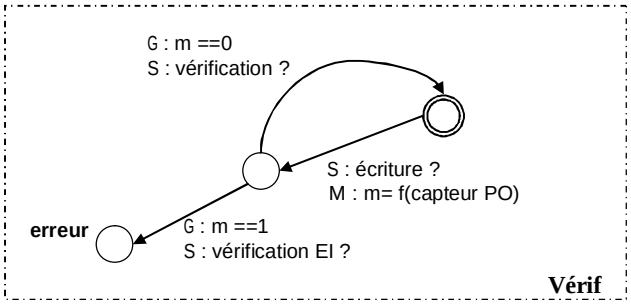


Figure 4.36. Modèle pour la vérification

Pour vérifier qu'aucun état interdit n'est atteignable, la propriété suivante a été testée :

P1 : A[] not(Verif.erreur)

Dans la syntaxe d'UPPAAL, dans les formules, A est le quantificateur universel de chemin, qui se lit *sur tout chemin...*, et [ ] est la modalité partout. La combinaison A[ ] signifie donc pour *tout état dans le futur*. De plus, *S.e* désigne l'état *e* de l'automate *S*. La propriété P1 se lit donc : « *Pour tous les états de tous les chemins, l'état erreur de l'automate Verif n'est pas vrai* ». Si la propriété suivante (P1) est satisfaite, cela signifie qu'aucun état défendu n'est

atteint et que, par conséquent, les contraintes sont suffisantes pour garantir la sécurité du système étudié.

A chaque cycle de l'API, lors de l'écriture des nouvelles sorties, l'équation logique *m* représentant les états interdits est calculée. Si la variable *m* est égale à 1 lors de la réception du message « *vérification EI* », le modèle passe dans l'état erreur, sinon le cycle continue. Il est à remarquer que le calcul de *m* peut se faire n'importe quand après l'évolution de la PO.

4.3 Vérification des contraintes pour les interactions avec le produit

La vérification des positions pouvant être prises par le produit est effectuée directement sur les modèles de produit. Dans ce cas, on teste que l'ensemble des contraintes est suffisant pour ne pas perdre la pièce ou la conduire dans des positions dangereuses, c'est-à-dire atteindre l'état noté « produit_perdu » dans le modèle de produit noté « Etat_produit » (figure 4.29). Ceci est garanti sous UPPAAL si la propriété suivante est vérifiée :

P2 : A[] not(Etat_produit.produit_perdu)

L'approche de vérification de la suffisance des contraintes, ainsi que les modèles à utiliser ont été présentés de manière théorique, la partie suivante propose de les appliquer sur plusieurs exemples différents et complémentaires permettant d'illustrer l'intérêt de la méthodologie.

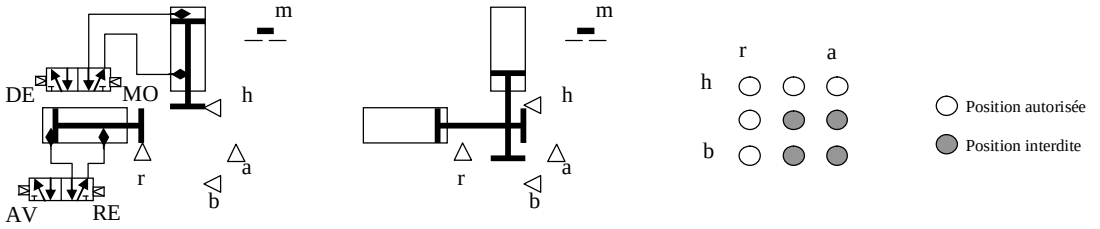
5 Exemples pédagogiques et implémentation sous UPPAAL

L'approche a été testée sur plusieurs exemples pédagogiques, reprenant des cas rencontrés dans les systèmes manufacturiers. Cette partie présente l'ensemble de contraintes couramment défini pour chaque exemple, sa vérification sous UPPAAL et les modifications apportées si l'ensemble de départ n'est pas suffisant. Deux exemples sont traités : le partage d'une zone commune entre deux vérins et un préhenseur avec capteur intermédiaire. Ces exemples vont permettre de montrer l'intérêt de ces travaux, la possibilité de réutiliser les mêmes modèles pour des systèmes utilisant les mêmes ElPO, et la nécessité de contraindre plus fortement la commande pour garantir la sécurité compte tenu de l'existence du retard de causalité.

5.1 Partage d'une zone commune entre deux vérins

Ce premier exemple concerne le partage d'une zone commune entre deux vérins repris de la thèse de Deschamps (Deschamps, 2007), et déjà présenté dans l'introduction en y ajoutant la technologie utilisée pour les distributeurs.

Soit un système composé de deux vérins : un vérin horizontal {*a, r, AV, RE*} et un vérin vertical {*b, h, MO, DE*} pilotés par des distributeurs pneumatiques 5/2. Les vérins sont placés perpendiculairement l'un par rapport à l'autre et partagent une zone commune (figure 4.37.a). L'analyse dysfonctionnelle met en évidence que la situation de la figure 4.37.b ne doit pas être possible pour éviter la détérioration du système. Les quatre positions de la figure 4.37.c sont considérées comme interdites.



a) Position autorisée b) Position interdite c) Différentes positions atteignables

Figure 4.37. Vérin avec distributeur 5/2 avec une zone commune

Pour cet exemple, les variables utilisées sont les suivantes :

- Au niveau du système (PO) :
  - Vérin horizontal {*a, r, AV, RE, sens_vers_a, sens_vers_r*}
  - Vérin vertical {*b, h, MO, DE, sens_vers_b, sens_vers_h*}
- Au niveau de l'API :
  - Vérin horizontal {*a_auto, r_auto, AV_auto, RE_auto*}
  - Vérin vertical {*b_auto, h_auto, MO_auto, DE_auto*}
  - Variable indiquant le non respect d'une contrainte : erreur

Comme nous l'avons vu dans l'exemple introductif, les contraintes définies dans la thèse de E. Deschamps interdisent d'activer l'ordre *AV* si le vérin vertical n'est pas en *h*, ce qui s'écrit :

$$AV\_auto \wedge \neg h\_auto = 0 \qquad (4.4)$$

De même l'ordre *DE* est interdit si le vérin horizontal n'est pas en *r* :

$$DE\_auto \wedge \neg r\_auto = 0 \qquad (4.5)$$

Une troisième contrainte (non spécifiée par Deschamps) est ajoutée pour interdire l'envoi simultané de *AV* et *DE* :

$$AV\_auto \wedge DE\_auto = 0 \qquad (4.6)$$

Pour tester ces contraintes, les modèles de commande (figure A2.6), sens (figure A2.8), positions (figure A2.11), évolution de l'environnement (figure A2.4) ont été implémentés. Le modèle de vérification a été défini au moyen de l'équation logique représentant les états interdits :

$$\neg r \wedge \neg h = 0 \qquad (4.7)$$

La vérification de la propriété sous UPPAAL n'est pas satisfaite. Le chemin retourné par le model-checker permet de comprendre pourquoi la propriété n'est pas vérifiée et que par conséquent les 2 vérins peuvent entrer en collision.

On suppose au départ que les deux vérins se trouvent en position rentrée, sans commande envoyée et sans sens de déplacement. La commande *AV* est ensuite activée. Toutes les contraintes sont respectées, donc les sorties du système sont affectées. Au cycle suivant, la désactivation de *AV* et l'activation de *DE* sont demandées. Les contraintes sont toujours respectées et donc les sorties sont mises à jours. Étant donné que des distributeurs 5/2 sont utilisés, les déplacements des vérins vont se faire même si la commande n'est pas maintenue. Les vérins vont donc entrer en collision sans qu'aucune des 3 contraintes ne soit violée. Le problème vient du fait que la commande n'est pas similaire au sens de déplacement. Il est possible de garantir cette égalité en contraignant plus fortement la commande en ajoutant par exemple les 2 contraintes suivantes qui interdisent de désactiver la commande tant que les 2 vérins ne sont pas arrivés en fin de course :

$$\downarrow AV\_auto \wedge \neg a\_auto = 0 \qquad (4.8)$$

$$\downarrow DE\_auto \wedge \neg b\_auto = 0 \qquad (4.9)$$

Cette solution comprenant 5 contraintes a été vérifiée sous UPPAAL mais n'est pas unique. En effet, les 2 contraintes suivantes peuvent aussi être choisies pour contraindre la commande :

$$\downarrow AV\_auto \wedge r\_auto = 0 \qquad (4.10)$$

$$\downarrow DE\_auto \wedge h\_auto = 0 \qquad (4.11)$$

L'idée est ici d'interdire la désactivation des commandes tant que les vérins ne sont pas sortis.

Il est intéressant de noter que dans les 2 solutions, 5 contraintes sont suffisantes pour garantir que quel que soit le programme de commande implémenté dans l'API, la sécurité de la PO est assurée. En revanche, le fait qu'une commande conduise au non respect d'une contrainte ne signifie pas nécessairement que la commande proposée conduit à une collision des 2 vérins. Le choix entre les deux ensembles de contraintes, sera fait par l'expert en fonction de l'explication qu'il souhaite apporter.

À ces cinq contraintes, il faut rajouter les contraintes sur chaque ElPO. Leur non respect ne conduit pas à des détériorations, mais permettent d'apporter des explications au concepteur sur des commandes inutiles ou sans effet :

- Interdire l'envoi des deux commandes en même temps sur un même ElPO

$$AV\_auto \wedge RE\_auto = 0 \qquad (4.12)$$

- Interdire la commande *AV*, si le vérin horizontal est déjà en *a*

$$\uparrow AV\_auto \wedge a\_auto = 0 \qquad (4.13)$$

- Interdire la commande *RE*, si le vérin horizontal est déjà en *r*

$$\uparrow RE\_auto \wedge r\_auto = 0 \qquad (4.14)$$

- Désactiver la commande *RE*, si le vérin horizontal est arrivé en *r*

$$\uparrow r\_auto \wedge RE\_auto = 0. \qquad (4.15)$$

- Désactiver la commande *AV*, si le vérin horizontal est arrivé en *a*

$$\uparrow a\_auto \wedge AV\_auto = 0 \qquad (4.16)$$

- Interdire l'envoi des deux commandes en même temps

$$MO\_auto \wedge DE\_auto = 0 \qquad (4.17)$$

- Interdire la commande *MO*, si le vérin vertical est déjà en *h*

$$\uparrow MO\_auto \wedge h\_auto = 0 \qquad (4.18)$$

- Interdire la commande *DE*, si le vérin vertical est déjà en *b*

$$\uparrow DE\_auto \wedge b\_auto = 0 \qquad (4.19)$$

- Désactiver la commande *MO*, si le vérin vertical est arrivé en *h*

$$\uparrow h\_auto \wedge MO\_auto = 0 \qquad (4.20)$$

- Désactiver la commande *DE*, si le vérin vertical est arrivé en *b*

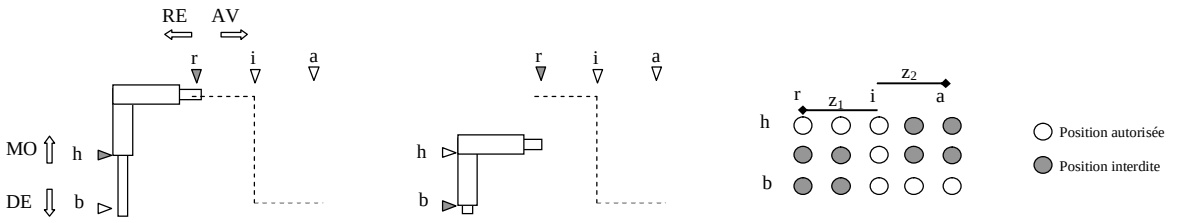
$$\uparrow b\_auto \wedge DE\_auto = 0 \qquad (4.21)$$

Ce premier exemple montre l'importance de vérifier au préalable les contraintes surtout lorsqu'elles sont utilisées sur un système réel. Cet exemple a été testé sur la maquette PRODUCTIS (cf. chapitre 5) et a donc permis d'assurer la sécurité indépendamment du temps de cycle de l'API.

Le deuxième exemple concerne aussi 2 vérins en interaction : il s'agit d'un préhenseur avec un capteur intermédiaire.

5.2 Préhenseur avec capteur intermédiaire

Nous considérons un préhenseur composé de deux vérins : un vérin horizontal {*a, i, r, AV, RE*} piloté par un distributeur 5/3 bistable et un vérin vertical {*b, h, MO, DE*} piloté par un distributeur 5/2 bistable. Le système doit se déplacer suivant un mouvement précis (figure 4.38) en S. L'analyse fonctionnelle et dysfonctionnelle consiste à définir les positions autorisées et interdites. On constate dans ce cas, que les contraintes de sécurité sont très proches du cahier des charges.



a) Position autorisée b) Position interdite c) Différentes positions atteignables

Figure 4.38. Préhenseur avec un capteur intermédiaire

Les variables utilisées au niveau de la PO et de l'API sont présentées figure 4.39.

	Variables système	Variables API
Vérin vertical	*MO, DE, h, b*	*MO_auto, DE_auto, h_auto, b_auto*
Vérin horizontal	*AV, RE, a, i, r*	*AV_auto, RE_auto, a_auto, i_auto, r_auto*

Figure 4.39. Variables pour le préhenseur avec capteur intermédiaire

Ayant la même information capteurs pour les positions intermédiaires entre *r* et *i* et entre *i* et *a*, il est nécessaire de reconstruire l'information sur les deux positions du vérin horizontal (cf. chapitre 3) pour écrire les contraintes, lorsque le vérin horizontal quitte le capteur *i*. Pour cela les zones *z1* et *z2* sont définies de la manière suivante : zone *z1* [*r, i* [et la zone *z2*]*i, a*]. La solution la plus simple pour reconstruire *z1* consiste à utiliser l'information provenant de *i* et de la commande au même instant. Si l'événement $\downarrow i$ occure alors que la commande RE est active, le vérin passe dans la zone *z1* d'où *z1* = 1 si $\downarrow i \wedge RE$ =1. Le vérin horizontal quitte la zone *z1* lorsque l'événement $\uparrow i$ occure, d'où *z1* =0 si $\uparrow i$ =1. Le raisonnement est identique pour *z2*.

Donc, dans la phase de vérification, les deux modèles de reconstruction de la figure 4.40. ont été définis à partir des informations lues par l'API : {*AV_auto, RE_auto, z1_auto, z2_auto, fd_i_auto, fm_i_auto*} où *fd_i_auto* et *fm_i_auto* représentent la désactivation et l'activation du capteur *i*.

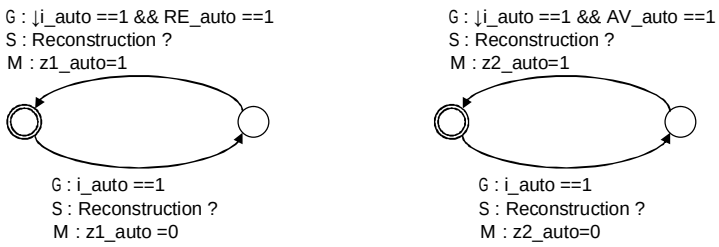


Figure 4.40. Modèle de reconstruction des zones z1 et z2

Ces deux reconstructeurs peuvent s'exprimer au moyen des équations logiques d'auto-maintien suivantes :

$$z1\_auto = \neg i\_auto(\wedge((\downarrow i\_auto \wedge RE\_auto) \vee z1\_auto)) \qquad (4.22)$$

$$z2\_auto = \neg i\_auto(\wedge((\downarrow i\_auto \wedge AV\_auto) \vee z2\_auto)) \qquad (4.23)$$

Toutefois, comme dans l'exemple précédent de la zone commune, ces reconstructeurs ne sont valables que si le sens de déplacement est équivalent à la commande lorsque le vérin quitte le capteur. Il est donc nécessaire de rajouter les contraintes suivantes qui autorisent l'arrêt de la commande uniquement sur un front montant ou un front descendant du capteur intermédiaire et en fin de course.

$$\downarrow AV\_auto \wedge \left(\neg(\uparrow i\_auto) \wedge \neg(\downarrow i\_auto) \wedge \neg a\_auto\right) = 0 \qquad (4.24)$$

$$\downarrow RE\_auto \wedge \left(\neg(\uparrow i\_auto) \wedge \neg(\downarrow i\_auto) \wedge \neg r\_auto\right) = 0 \qquad (4.25)$$

On notera ici l'importance de l'hypothèse concernant les contacts longs. L'arrêt de la commande *AV* sur un front montant du capteur conduit nécessairement le vérin à s'arrêter sur le capteur *i*. Sans cette hypothèse, le préhenseur pourrait se retrouver dans des zones interdites.

Les contraintes pour que le mouvement vertical s'effectue sur le capteur *i* sont :

$$MO\_auto \wedge \neg i\_auto = 0 \qquad (4.26)$$

$$DE\_auto \wedge \neg i\_auto = 0 \qquad (4.27)$$

Les six suivantes permettent d'assurer le mouvement horizontal en haut dans la zone *z1* et en bas dans la zone *z2* :

- Interdire la commande *AV* si le préhenseur n'est pas en position *h* dans la zone *z1*

$$AV\_auto \wedge z1\_auto \wedge \neg h\_auto = 0 \qquad (4.28)$$

- Interdire la commande *RE* si le préhenseur n'est pas en position *h* dans la zone *z1*

$$RE\_auto \wedge z1\_auto \wedge \neg h\_auto = 0 \qquad (4.29)$$

- Interdire la commande *AV* si le préhenseur n'est pas en position *b* dans la zone *z2*

$$AV\_auto \wedge z2\_auto \wedge \neg b\_auto = 0 \qquad (4.30)$$

- Interdire la commande *R*E si le préhenseur n'est pas en position *b* dans la zone *z2*

$$RE\_auto \wedge z2\_auto \wedge \neg b\_auto = 0 \qquad (4.31)$$

- Interdire la commande *RE* lorsqu'il est en position *i* s'il n'est pas en position *h*

$$RE\_auto \wedge i\_auto \wedge \neg h\_auto = 0 \qquad (4.32)$$

- Interdire la commande *AV* lorsqu'il est en position *i* s'il n'est pas en position *b*

$$AV\_auto \wedge i\_auto \wedge \neg b\_auto = 0 \qquad (4.33)$$

À ces huit contraintes, il faut ajouter deux contraintes pour interdire les mouvements diagonaux en i qui restent autorisés malgré les contraintes déjà définies :

$$AV\_auto \wedge MO\_auto = 0 \qquad (4.34)$$

$$RE\_auto \wedge DE\_auto = 0 \qquad (4.35)$$

Comme dans l'exemple précédent, comme il s'agit d'un distributeur 5/2 bistable, il est nécessaire de contraindre plus fortement la commande. Deux contraintes supplémentaires sur le vérin vertical sont ajoutées :

$$\downarrow DE\_auto \wedge h\_auto = 0 \qquad (4.36)$$

$$\downarrow MO\_auto \wedge b\_auto = 0 \qquad (4.37)$$

Comme cela a déjà été souligné, d'autres contraintes spécifiques à des ElPO ou à des interactions entre ElPO peuvent être rajoutées en vue de leur pouvoir explicatif.

La vérification de la propriété sous UPPAAL se fait sur le modèle de vérification en posant l'équation logique suivante reflétant les huit positions que la PO ne doit pas atteindre (figure 4.38.c) :

$$(\neg z1 \wedge \neg i \wedge \neg h) \vee (\neg z2 \wedge \neg i \wedge \neg b) \qquad (4.38)$$

Les variables *z1* et *z2* proviennent du modèle de positions de la PO (cf. Annexe 2). La propriété est alors respectée.

Ce deuxième exemple a permis de présenter l'utilisation des modèles de reconstruction basés sur l'information contenue dans l'API. D'autres exemples ont été traités et sont présentés dans l'annexe 2.

6 Conclusion

Ce chapitre a proposé une approche de vérification d'un ensemble de contraintes, permettant de s'assurer que celui-ci est suffisant et complet pour assurer la sécurité sans faire d'hypothèse sur la commande envoyée. Pour cela, la démarche appelée C.S.P. « Commande, Sens, Positions » repose sur l'utilisation de modèles représentant la chaîne fonctionnelle. L'utilisation de ces modèles permet de minimiser l'écart entre le système réel et le modèle en prenant en compte la technologie des ElPO, l'inertie qui peut être présente, le retard de causalité et le fonctionnement d'un API.

Plusieurs exemples ont été traités pour prendre en compte différents aspects pouvant être rencontrés dans les systèmes manufacturiers. En appliquant cette approche sur différents types d'éléments, on peut développer une bibliothèque de contraintes, vérifiées formellement, pour différents ElPO et produits en interaction (cf. Annexe 3). L'expert dispose ainsi d'un ensemble de contraintes pour des interactions différentes qu'il peut choisir en fonction des explications qu'il veut apporter.

Enfin, étant donné que les contraintes sont indépendantes de la commande proposée, elles restent valables pendant tout le cycle de vie de la PO. En d'autres termes, même si un opérateur modifie la commande, le filtre continue d'assurer la sécurité de l'installation.

Par rapport aux objectifs fixés au début de ce chapitre, l'approche proposée permet de vérifier la suffisance des contraintes et permet ainsi de garantir la sécurité du système. Cependant, la dernière étape qui consiste à vérifier la nécessité de chaque contrainte n'a pas été traitée et fait l'objet de perspectives. Cela permettrait de trouver le plus petit sous-ensemble de contraintes suffisant. Le fait de minimiser les ressources occupées par le filtre dans l'API peut présenter un intérêt pour certaines applications industrielles nécessitant un temps de cycle rapide.

Le chapitre suivant présente les applications qui ont été réalisées en vue de valider l'approche par filtrage robuste de la commande qui a été proposée.

Chapitre 5 :
Applications et expérimentations

1 Introduction

Ce dernier chapitre présente deux applicatifs dans le cadre de l'enseignement, l'un sur un système réel et l'autre sur un système virtuel permettant de mettre en avant :

- L'adaptation de la partie opérative (PO) au niveau des apprenants tout en conservant une vision globale du dispositif,
- L'intérêt et les performances du filtre robuste de commandes pour d'une part sécuriser le système et d'autre part fournir des explications à l'apprenant.

Le premier exemple concerne la machine PRODUCTIS de chez Schneider (Schneider Electric, 2001). Celle-ci peut être considérée comme complexe et nous l'avons sécurisée au moyen d'un filtre robuste de commandes. Il est à noter que cette installation a été adaptée aussi bien dans des activités de sensibilisation aux automatismes pour des automaticiens novices que par des étudiants de niveau DUT et MASTER.

Le second applicatif repose sur l'utilisation du logiciel de simulation de systèmes industriels ITS PLC de chez Real Games (Riera et al., 2008) (Real Games, 2008), avec qui le CReSTIC a signé un partenariat scientifique et technique. Nous avons choisi de traiter l'exemple d'un magasin automatique, car il fait apparaître aussi bien des dysfonctionnements au niveau du système, qu'au niveau du produit. Ces deux applicatifs ont un rôle éducatif (Marangé et al. 2006 a ; 2008 a). Il est clair que le filtre de validation est principalement

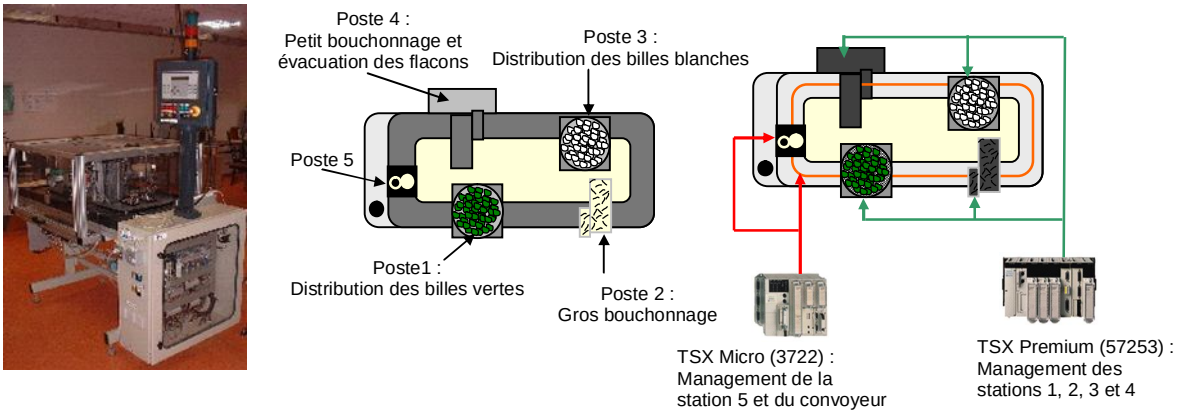
utilisé dans le premier cas pour éviter toutes détériorations de la machine PRODUCTIS et l'adapter au niveau des apprenants. En revanche, ce sont les capacités explicatives du filtre qui sont exploitées dans le second exemple. En effet, le problème de casse matérielle et de danger ne se pose pas lors de l'utilisation de maquettes virtuelles. Cependant, la simulation permet de visualiser et donc de faire prendre conscience à l'apprenant des conséquences d'une mauvaise commande pour le système. Le filtre est dans ce cas uniquement utilisé pour détecter les erreurs et fournir une explication à l'apprenant.

Pour chacune des applications, le système est dans un premier temps décrit. Ensuite, une analyse de celui-ci et des dysfonctionnements est proposée. L'ensemble des contraintes est alors défini et validé formellement par l'approche proposée dans le chapitre 4. Enfin, le filtre est implémenté dans l'Automate Programmable Industriel (API). Les différentes expérimentations mises en place avec les apprenants sont présentées ainsi que les résultats obtenus.

2 Application sur un système réel

Le département GCE (Génie du Conditionnement et de l'Emballage) de l'IUT de Reims-Châlons-Charleville dispose d'un système de conditionnement de médicaments, appelée PRODUCTIS (figure 5.1.) de chez Schneider. Ce système, même s'il s'agit d'une maquette pédagogique, est réalisé au moyen de composants industriels coûteux. Il peut être considéré comme complexe compte tenu du nombre d'Entrées (64) et de Sorties (46) des 2 Automates Programmables Industriels en réseau qui permettent de le piloter. PRODUCTIS présente également des risques importants de casse si la commande est erronée. D'une part, il est donc impératif de le sécuriser pour qu'il soit utilisé par les étudiants automaticiens dans le cadre de leur formation. D'autre part, il est nécessaire d'adapter la PRODUCTIS au niveau des apprenants. Il s'agit donc d'un applicatif idéal pour valider notre travail de recherche

Les paragraphes suivants présentent comment a été conçu et mis en place le filtre de validation. La définition et la vérification de l'ensemble de contraintes sont détaillées. Nous avons menée plusieurs expérimentations lors de manifestations telles que la Fête de la Science ou lors de TP avec des étudiants de GCE et de Master Pro en Automatisation et Supervision, où la PRODUCTIS a été adaptée au niveau des apprenants.



a) Système b) Architecture réseaux

Figure 5.1. Machine PRODUCTIS

2.1 Description du système

La machine PRODUCTIS (figure 5.1.) est une maquette représentant un système de conditionnement de médicaments. Cinq postes agencés autour d'un convoyeur permettent le conditionnement de flacons de deux tailles, avec deux types de médicaments (billes vertes ou blanches). L'alimentation se fait par le poste 5, où une palette est déposée avec un bouchon et un flacon. Le flacon est rempli par des billes vertes et/ou blanches au poste 1 et au poste 3. Une fois le flacon rempli de billes, il est bouchonné soit au poste 2, soit au poste 4, en fonction de sa taille. La dernière opération est l'évacuation du flacon bouchonné au poste 4. La palette est ensuite ramenée au poste 5 pour être rechargée manuellement en flacon et bouchon.

La distribution de billes aux postes 1 et 3 (figure 5.2.) se fait par rotation de deux plateaux {ROT_i+, ROT_i-, $prot_i$+, $prot_i$-, det_i} où ROT_i+ représente la commande pour faire une rotation dans le sens positif sur le poste i (i = 1 ou 3), ROT_i- représente la commande pour faire une rotation dans le sens négatif. Le capteur $prot_i$+ représente le plateau en position positive et $prot_i$- en position négative. Le système détecte la distribution d'une bille par le capteur det_i. L'actionneur bistable permettant la rotation est alimenté par un distributeur 5/2.

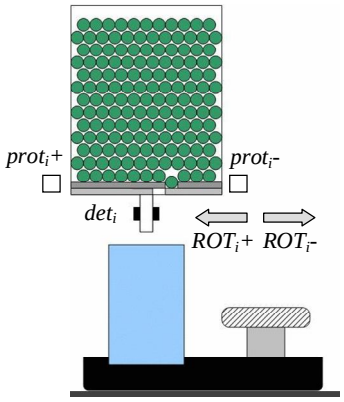


Figure 5.2. Postes de distribution de billes

Le bouchonnage du flacon se fait sur le poste 2 si c'est un gros flacon et sur le poste 4 si c'est un petit flacon (figure 5.3.). Ces postes sont composés d'un préhenseur pneumatique permettant le positionnement du bouchon et d'une tête aspirante permettant le maintien du bouchon. La préhension du bouchon se fait par l'ordre d'aspiration Asp_j, le positionnement sur le flacon est réalisé par l'intermédiaire d'un préhenseur composé de deux vérins : un vérin vertical $\{Av_j, Re_j, a_j, r_j\}$ et un vérin horizontal $\{Mo_j, De_j, h_j, b_j\}$ où Av_j représente la commande pour avancer le préhenseur, Re_j pour le rentrer, Mo_j pour le monter, De_j pour le descendre, et a_j, r_j, h_j, b_j représentent respectivement la position avancée, rentrée, haute et basse du préhenseur sur le poste j (j = 2 ou 4). Pour finir le bouchonnage, l'aspiration est relâchée et le bouchon est éjecté. L'aspiration Asp_j et l'éjection $Eject_j$ sont des actionneurs monostables, l'activation de l'aspiration doit donc être maintenue pendant le déplacement du préhenseur.

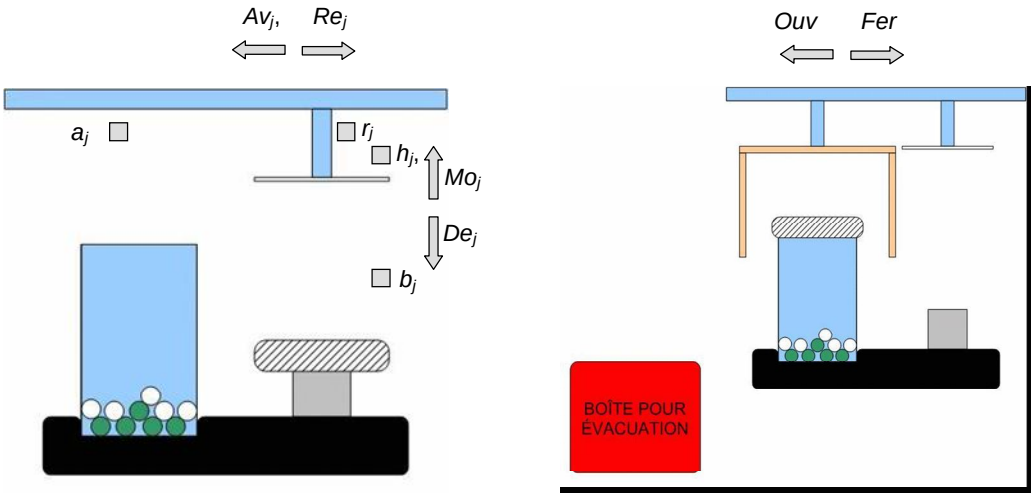


Figure 5.3. Postes de bouchonnage et évacuation

Le passage de la palette contenant le flacon et le bouchon d'un poste à l'autre se fait simplement par relâchement de butées sur chaque poste {B_1, B_2, B_3, B_4}. En effet, le convoyeur est en mouvement perpétuel et chaque poste est muni d'une butée pour retenir la palette pendant l'opération. Lorsqu'une palette est présente sur l'un des postes, un capteur la détecte : {ppp_1, ppp_2, ppp_3, ppp_4}. La butée du poste 5 n'est pas commandée et s'effectue manuellement par l'appui sur un bouton.

Pour finaliser le conditionnement du flacon, celui-ci doit être évacué au poste 4. Pour cela, en plus de la tête aspirante, le préhenseur est muni d'une pince commandée par un actionneur bistable {*Ouv*, *Fer*} permettant de tenir le flacon.

La figure 5.4. rappelle les variables système et les variables dans l'API. Volontairement, cette distinction est faite pour conserver l'écriture des contraintes dans le chapitre 4.

Pour permettre de définir les contraintes nécessaires pour assurer la sécurité du système, une analyse fonctionnelle des interactions entre les éléments de partie opérative (ElPO) et des dysfonctionnements possibles doit être effectuée.

	Variables système	Variables API
Distributeur i=2, 4	ROT_i+, ROT_i-,	ROT_i+_auto, ROT_i-_auto,
	$prot_i$+, $prot_i$-,det_i	$prot_i$+_auto, $prot_i$-_auto, det_i_auto
Vérin vertical j=1, 3	Mo_j, De_j,	Mo_j_auto, De_j_auto,
	h_j, b_j	h_j_auto, b_j_auto
Vérin horizontal j=1, 3	Av_j, Re_j,	Av_j_auto, Re_j_auto,
	a_j, r_j	a_j_auto, r_j_auto
Tête aspirante j=1, 3	Asp_j $Eject_j$	Asp_j_auto, $Eject_j$_auto
Pince	*Ouv, Fer*	*Ouv_auto, Fer_auto*
Butées	B_1, B_2, B_3, B_4	B_1_auto, B_2_auto, B_3_auto, B_4_auto
	ppp_1, ppp_2, ppp_3, ppp_4	ppp_1_auto, ppp_2_auto, ppp_3_auto, ppp_4_auto

Figure 5.4. Variables système / API de PRODUCTIS

2.2 Analyse fonctionnelle et dysfonctionnelle du système

La fonction principale de la machine PRODUCTIS est le conditionnement de médicaments (niveau 0). Pour cela quatre postes permettent les quatre sous fonctions (niveau 1) : distribution du nombre désiré de billes vertes et blanches, le bouchonnage et l'évacuation du flacon. Le tableau de la figure 5.5. décrit chacun des niveaux.

Niveau 0	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5	Dysfonctionnements possibles
Conditionner des médicaments dans des flacons	Mettre N billes vertes	Distribuer N billes	Distribuer 1 bille verte	Rot_1+		Dp_1 : pas de palette ou en mouvement
				Rot_1-		
		Libérer palette	B_1			Dp_2 : palette présente sur le poste suivant Dp_3 : tête du préhenseur du poste suivant pas en position haute
	Mettre M billes blanches	Distribuer M billes	Distribuer 1 bille blanche	Rot_3+		Dp_1
				Rot_3-		
		Libérer palette	B_3			Dp_2 Dp_3
	Mettre un bouchon	Positionner le préhenseur au dessus du bouchon	De_j			Dp_4 : préhenseur pas en position rentrée
		Maintenir le bouchon	Asp_j		↑Asp_j	Dp_5 : préhenseur pas en position rentrée et basse
					↓Asp_j	Dp_6 : préhenseur pas en position avancée et basse
		Positionner le préhenseur au dessus du flacon	Mo_j			Dp_4, Dp_7 : préhenseur pas en position haute pour le mouvement horizontal Dp_8 : Remonter avec le flacon
			Av_j			
			De_j			
			Re_j			
		Ejecter le bouchon	$Eject_j$			Dp_6
		Libérer la palette	B_j			Dp_2, Dp_9 : tête du préhenseur du poste pas en position haute
	Evacuer le flacon	Positionner le préhenseur au dessus du flacon	De_4			Dp_4, Dp_{10} : Pince fermée
		Fermer la pince	Fer			Dp_5
		Positionner le préhenseur au dessus de l'évacuation	Mo_4			Dp_4, Dp_6
			Av_4			
			De_4			
			Re_4			
		Ouvrir la pince	Ouv			Dp_7
		Libérer palette	B_4			Dp_2, Dp_9

Figure 5.5. Analyse du système PRODUCTIS

Comme nous le verrons dans la suite, c'est l'analyse du système qui permet d'adapter la PRODUCTIS au niveau des apprenants tout en conservant une vision globale de l'installation.

A partir de l'analyse du système, dix dysfonctionnements aux conséquences plus ou moins critiques, peuvent apparaître suite à une mauvaise commande :

- Dp_1 : Mettre une bille alors qu'il n'y a pas de palette sur le poste ou si la palette est en mouvement ;
- Dp_2 : Libérer une palette du poste *l*, alors qu'il y a déjà une palette sur le poste suivant *l*+1. L'arrivée d'une nouvelle palette sur le poste suivant va provoquer un choc sur l'autre palette ;
- Dp_3 : Libérer une palette alors que le préhenseur du poste suivant n'est pas en position haute, peut provoquer une collision entre le flacon et le préhenseur ;
- Dp_4 : Le mouvement vertical peut se faire seulement en position rentrée ou avancée pour éviter la collision entre le flacon ou le bouchon et le préhenseur

- Dp_5 : L'activation de l'aspiration peut se faire seulement en position rentrée et basse pour éviter d'aspirer le flacon
- Dp_6 : La désactivation de l'aspiration ou l'activation de l'éjection peut se faire seulement position avancée et basse pour éviter de perdre le bouchon
- Dp_7 : Le mouvement horizontal peut se faire seulement en position haute pour éviter la collision entre le flacon ou le bouchon et le préhenseur
- Dp_8 : Le mouvement de monter en position avancée et l'aspiration maintenue, peut provoquer le déplacement du flacon, en même temps que le bouchon
- Dp_9 : Libérer une palette alors que le préhenseur n'est pas en position haute, peut provoquer une collision entre le flacon et le préhenseur ;
- Dp_{10} : La descente de la pince fermée en position rentrée peut provoquer la collision entre la pince et le flacon.

À ces dysfonctionnements s'ajoutent les erreurs qui peuvent être commises par l'apprenant, n'ayant pas d'effets sur la sécurité du système, mais qui doivent être définies pour apporter une explication :

- Envoi de deux ordres inverses sur un même ElPO
- Envoi d'un ordre alors que les conditions de désactivation sont déjà présentes
- Non désactivation d'un ordre lorsque les conditions de désactivation deviennent vraies

L'analyse du système permet aussi de définir les interactions (cf. chapitre 3) qui existent dans le système définissant ainsi les contraintes et les modèles nécessaires à la vérification de celles-ci :

- Poste de distribution de billes (poste 1 et poste 3) :
  - Zone commune entre la palette et le plateau de distribution
  - Zone commune entre la palette et la palette du poste suivant
  - Zone commune entre la palette et le préhenseur du poste suivant
- Poste de bouchonnage (poste 2 et poste 4) :
  - Interaction physique entre le vérin horizontal et vertical formant le préhenseur
  - Interaction physique entre l'aspiration et le préhenseur

- Interaction physique entre l'éjection et le préhenseur
- Zone commune entre la palette et le préhenseur
- Zone commune entre la palette et la palette du poste suivant

A partir de cette analyse, les contraintes nécessaires pour assurer la sécurité ont été définies et vérifiées sous UPPAAL suivant l'approche proposée dans le chapitre 4.

2.3 Définition des contraintes

Pour chaque poste et chaque dysfonctionnement, un ensemble de contraintes a été défini en tenant compte de la technologie. Avant cela, les contraintes prenant en compte les erreurs pouvant être faites par le concepteur sur un ElPO, sont présentées.

2.3.1 Contraintes pour les ElPO pris indépendamment

Il a fallu définir des contraintes de sécurité pour les vérins du préhenseur, les plateaux de distribution des billes et pour la pince. Les contraintes sont énoncées rapidement sans détail car ce sont les mêmes que dans les exemples traités dans le chapitre 4 :

Vérins préhenseur

- Interdire l'envoi des deux commandes en même temps sur le poste *j* où *j* = 2 ou 4

$$Av_j\_auto \wedge \mathrm{Re}_j\_auto = 0 \qquad \textit{(5.1, 5.2)}$$

- Interdire la commande Av_j, si le vérin horizontal est déjà en a_j

$$\uparrow Av_j\_auto \wedge a_j\_auto = 0 \qquad \textit{(5.3, 5.4)}$$

- Interdire la commande Re_j, si le vérin horizontal est déjà en r_j

$$\uparrow \mathrm{Re}_j\_auto \wedge r_j\_auto = 0 \qquad \textit{(5.5, 5.6)}$$

- Désactiver la commande Re_j, si le vérin horizontal est arrivé en r_j

$$\uparrow r_j\_auto \wedge \mathrm{Re}_j\_auto = 0 \qquad \textit{(5.7, 5.8)}$$

- Désactiver la commande Av_j, si le vérin horizontal est arrivé en a_j

$$\uparrow a_j\_auto \wedge Av_j\_auto = 0 \qquad \textit{(5.9, 5.10)}$$

- Interdire l'envoi des deux commandes en même temps

$$Mo_j\_auto \wedge De_j\_auto = 0 \qquad \textit{(5.11, 5.12)}$$

- Interdire la commande Mo_j, si le vérin vertical est déjà en h_j

$$\uparrow Mo_j\_auto \wedge h_j\_auto = 0 \qquad (5.13, 5.14)$$

- Interdire la commande De_j, si le vérin vertical est déjà en b_j

$$\uparrow De_j\_auto \wedge b_j\_auto = 0 \qquad (5.15, 5.16)$$

- Désactiver la commande Mo_j, si le vérin vertical est arrivé en h_j

$$\uparrow h_j\_auto \wedge Mo_j\_auto = 0 \qquad (5.17, 5.18)$$

- Désactiver la commande De_j, si le vérin vertical est arrivé en b_j

$$\uparrow b_j\_auto \wedge De_j\_auto = 0 \qquad (5.19, 5.20)$$

Vérins du plateau de distribution des billes

- Interdire l'envoi des deux commandes en même temps sur le poste i où i = 1 ou 3

$$Rot_i + \_auto \wedge Rot_i - \_auto = 0 \qquad (5.21, 5.22)$$

- Interdire la commande Rot_i+, si le vérin plateau est déjà en $prot_i+$

$$\uparrow Rot_i + \_auto \wedge prot_i + \_auto = 0 \qquad (5.23, 5.24)$$

- Interdire la commande Rot_i-, si le vérin plateau est déjà en $prot_i-$

$$\uparrow Rot_i - \_auto \wedge prot_i - \_auto = 0 \qquad (5.25, 5.26)$$

- Désactiver la commande Rot_i-, si le vérin plateau est arrivé en $prot_i-$

$$\uparrow prot_i - \_auto \wedge Rot_i - \_auto = 0 \qquad (5.27, 5.28)$$

- Désactiver la commande Rot_i+, si le vérin plateau est arrivé en $prot_i+$

$$\uparrow prot_i + \_auto \wedge Rot_i + \_auto = 0 \qquad (5.29, 5.30)$$

Vérin de la pince

- Interdire l'envoi des deux commandes en même temps

$$Ouv\_auto \wedge Fer\_auto = 0 \qquad (5.31)$$

- Interdire la commande *Ouv*, si le vérin plateau est déjà en *pouv*

$$\uparrow Ouv\_auto \wedge pouv\_auto = 0 \qquad (5.32)$$

- Interdire la commande *Fer*, si la pince est déjà fermée

$$\uparrow Fer\_auto \wedge \neg pouv\_auto = 0 \qquad (5.33)$$

- Désactiver la commande *Ouv* quand la pince est ouverte,

$$\uparrow pouv\_\text{auto} \wedge Ouv\_auto = 0 \qquad (5.34)$$

- Désactiver la commande *Fer*, quand la pince est fermée

$$\downarrow pouv\_\text{auto} \wedge Fer\_auto = 0 \qquad (5.35)$$

Pour ces contraintes, il faut reconstruire l'information pince ouverte *pouv_auto*, car il n'y a pas de capteurs pour indiquer la position de la pince. La reconstruction se fait simplement en supposant que l'ouverture ou la fermeture est instantanée (figure 5.6.).

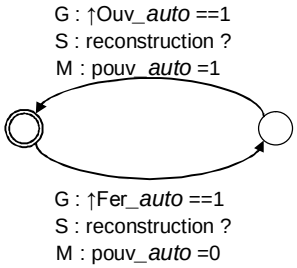


Figure 5.6. Reconstruction de l'information de la position de la pince

Pour les autres ElPO qui composent la machine PRODUCTIS, il n'y a pas de contraintes à définir pour les ElPO pris indépendamment. Nous rappelons que ces contraintes ne sont pas utiles pour assurer la sécurité du système mais elles sont importantes pour apporter des explications à l'étudiant.

2.3.2 Contraintes pour les interactions entre ElPO et produit

2.3.2.1 Poste 1 : distribution de billes vertes et Poste 3 : distribution de billes blanches

Le dysfonctionnement possible sur ces postes, est la distribution d'une bille alors qu'il n'y a pas de palette (Dp1). Pour éviter cela, quatre contraintes ont été définies pour chacun des postes 1 et 3 (i=1, 3) :

- Interdire la commande Rot_i+ s'il n'y a pas de palette sur le poste i (ppp_i)

$$Rot_i + \_auto \wedge \neg ppp_i\_auto = 0 \qquad (5.36, 5.37)$$

- Interdire la commande Rot_i- s'il n'y a pas de palette sur le poste i (ppp_i)

$$Rot_i - \_auto \wedge \neg ppp_i\_auto = 0 \qquad (5.38, 5.39)$$

- Interdire l'envoi des commandes Rot_i+ et B_i en même temps

$$Rot_i + \_auto \wedge B_i \_auto = 0 \qquad (5.40, 5.41)$$

- Interdire l'envoi des commandes Rot_i+ et B_i en même temps

$$Rot_i + \_auto \wedge B_i \_auto = 0 \qquad (5.42, 5.43)$$

La vérification de la suffisance de cet ensemble se fait en vérifiant s'il est possible d'atteindre la position

$$\neg ppp_i \wedge \neg prot_i + \wedge \neg prot_i - . \qquad (5.44)$$

En effet si cette position est atteignable, cela signifie que le plateau peut quitter sa position de repos pour libérer une bille sans qu'il y ait une palette. La propriété P1 (Chapitre 4 §4.2) a été vérifiée sous UPPAAL.

2.3.3 Poste 2 : Petit bouchonnage et Poste 4 : Gros bouchonnage

Sur les postes 2 et 4, cinq dysfonctionnements peuvent apparaitre si le mouvement horizontal ne se fait pas en haut (Dp_7), si le mouvement vertical ne se fait pas en position avancée ou rentrée (Dp_4), si l'activation (Dp_5) ou la désactivation (Dp_6) de l'aspiration ou l'activation de l'éjection ne se font pas au bon endroit, et si le flacon est remonté (Dp_8).

La définition des contraintes pour palier aux deux premiers dysfonctionnements se fait rapidement, car le cas du préhenseur piloté par des distributeurs 5/2 a été traité et les contraintes vérifiées dans le chapitre 4 §5.1. Nous utilisons donc le principe de la bibliothèque de contraintes qui a été proposée. La vérification des trois derniers dysfonctionnements s'effectue en même temps car ils sont liés par le produit.

2.3.3.1 Dysfonctionnements Dp_4 et Dp_7

Les huit contraintes vérifiées pour assurer le mouvement en U du préhenseur sont les suivantes pour le poste 2 et 4 :

- Interdire la commande Av_j si le préhenseur n'est pas en position h_j

$$Av_j \_ auto \wedge \neg h_j \_ auto = 0 \qquad (5.45, 5.46)$$

- Interdire la commande Re_j si le préhenseur n'est pas en position h_j

$$\mathrm{Re}_j \_ auto \wedge \neg h_j \_ auto = 0 \qquad (5.47, 5.48)$$

- Interdire l'envoi des deux commandes en même temps

$$Av_j\_auto \wedge De_j\_auto = 0 \qquad \textit{(5.49, 5.50)}$$

- Interdire l'envoi des deux commandes en même temps

$$\mathrm{Re}_j\_auto \wedge De_j\_auto = 0 \qquad \textit{(5.51, 5.52)}$$

- Interdire de descendre si le vérin vertical n'est pas en a ou en r

$$De_j\_auto \wedge \neg a_j\_auto \wedge \neg r_j\_auto = 0 \qquad \textit{(5.53, 5.54)}$$

Pour assurer l'équivalence entre la commande et le sens, il est nécessaire de contraindre plus fortement la commande en définissant par exemple les trois contraintes suivantes (cf. chapitre 4 §5.1) :

- Interdire de désactiver Av_j si le préhenseur n'est pas en a_j

$$\downarrow Av_j\_auto \wedge \neg a_j\_auto = 0 \qquad \textit{(5.55, 5.56)}$$

- Interdire de désactiver Re_j si le préhenseur n'est pas en r_j

$$\downarrow \mathrm{Re}_j\_auto \wedge \neg r_j\_auto = 0 \qquad \textit{(5.57, 5.58)}$$

- Interdire de désactiver De_j si le préhenseur n'est pas en b_j

$$\downarrow De_j\_auto \wedge \neg b_j\_auto = 0 \qquad \textit{(5.59, 5.60)}$$

2.3.3.2 Dysfonctionnements Dp_5, Dp_6, Dp_8

Il y a six contraintes à définir pour assurer que l'activation et la désactivation de l'aspiration et de l'éjection se font au bon endroit :

- L'activation de Asp_j ne peut pas se faire si le préhenseur n'est pas en position b_j et r_j

$$\uparrow Asp_j\_auto \wedge (\neg b_j\_auto \vee \neg r_j\_auto) = 0 \qquad \textit{(5.61, 5.62)}$$

- La désactivation de Asp_j ne peut pas se faire si le préhenseur n'est pas en position b_j et a_j

$$\downarrow Asp_j\_auto \wedge (\neg b_j\_auto \vee \neg a_j\_auto) = 0 \qquad \textit{(5.63, 5.64)}$$

- L'activation de Asp_j ne peut pas se faire si le préhenseur est en train de monter

$$\uparrow Asp_j\_auto \wedge Mo_j\_auto = 0 \qquad \textit{(5.65, 5.66)}$$

- La désactivation de Asp_j ne peut pas se faire si le préhenseur est en train de monter

$$\downarrow Asp_j \_ auto \wedge Mo_j \_ auto = 0 \qquad (5.67, 5.68)$$

- L'activation de $Eject_j$ ne peut pas se faire si le préhenseur n'est pas en position b_j et a_j

$$\uparrow Eject_j \_ auto \wedge (\neg b_j \_ auto \vee \neg a_j \_ auto) = 0 \qquad (5.69, 5.70)$$

- L'activation de $Eject_j$ ne peut pas se faire si le préhenseur est en train de monter

$$\uparrow Eject_j \_ auto \wedge Mo_j \_ auto = 0 \qquad (5.71, 5.72)$$

A cela, il faut ajouter la contrainte qui permet l'équivalence entre la commande et le sens de *Mo* :

$$\downarrow Mo_j \_ auto \wedge \neg h_j \_ auto = 0 \qquad (5.73)$$

Pour vérifier la suffisance de cet ensemble de contraintes, deux modèles de produit sont définis *Produit_aspiré* et *Produit_éjecté* (figure 5.7.) et par l'intermédiaire de la propriété P2 (chapitre 4 §4.3.), la suffisance de cet ensemble est démontrée.

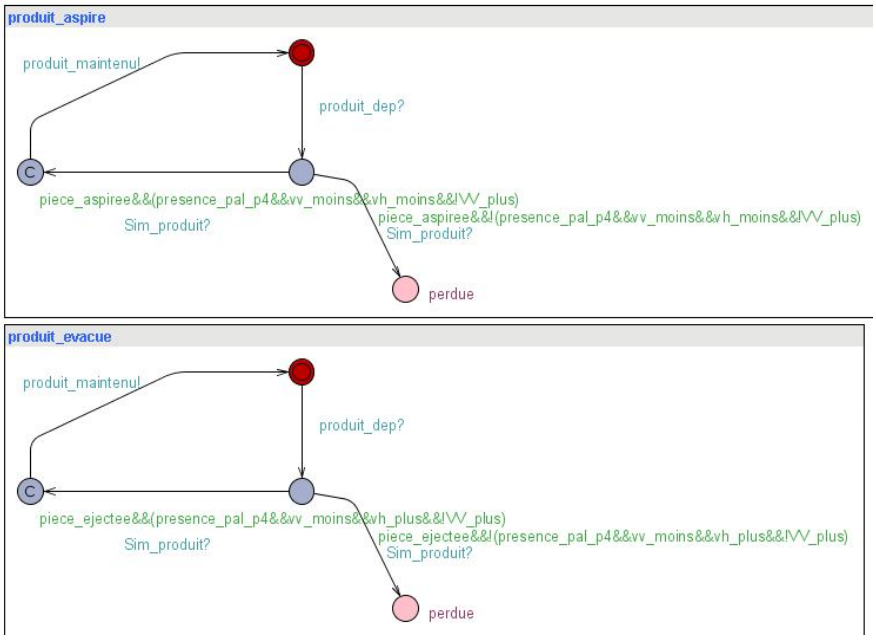


Figure 5.7. Modèles de produit

Pour éviter de remonter le flacon en même temps que la tête aspirante, il faut ajouter une contrainte qui interdit d'avoir Mo_j et Asp_j si le préhenseur est avancé :

$$Mo_j \_ auto \wedge Asp_j \_ auto \wedge a_j \_ auto = 0 \qquad (5.74)$$

Pour vérifier la suffisance de cette contrainte associée à la contrainte 5.73, la propriété P1 est utilisée avec la position interdite :

$$Asp_j \wedge a_j \wedge Mo_j \_ auto . \qquad (5.75)$$

2.3.4 Poste 4 : Évacuation du flacon

2.3.4.1 Dysfonctionnements Dp_5, Dp_6

Sur le poste 4, les dysfonctionnements liés à la prise du flacon par la pince sont les même que ceux pour la prise du bouchon (Dp_5, Dp_6). Les contraintes associées à ces dysfonctionnement ne sont pas à vérifiées car elles l'ont été pour le déplacement du bouchon. Les contraintes sont les suivantes :

- L'activation de *Fer* peut se faire seulement si le préhenseur est en position b_4 et r_4

  $$\uparrow Fer \_ auto \wedge (\neg b_4 \_ auto \vee \neg r_4 \_ auto) = 0 \qquad (5.76)$$

- La désactivation de *Ouv* peut se faire seulement si le préhenseur est en position b_4 et a_4

  $$\uparrow Ouv \_ auto \wedge (\neg b_4 \_ auto \vee \neg a_4 \_ auto) = 0 \qquad (5.77)$$

- L'activation de *Fer* ne peut pas se faire si le préhenseur est en train de monter

  $$\uparrow Fer \_ auto \wedge Mo_4 \_ auto = 0 \qquad (5.78)$$

2.3.4.2 Dysfonctionnement Dp_{10}

Lors de la prise du flacon, il faut s'assurer que la pince n'est pas fermée pour descendre la pince en position rentrée, sinon il peut y avoir collision entre le flacon et la pince (Dp_{10}). Les contraintes pour éviter que la pince fermée descende en position rentrée sont les suivantes :

- Interdire l'activation de De_4 en position r_4 si la pince n'est pas ouverte.

  $$\uparrow De_4 \_ auto \wedge \neg pouv \_ auto \wedge r_4 \_ auto = 0 \qquad (5.79)$$

- Interdire l'envoi des deux commandes en même temps

  $$Fer \_ auto \wedge De_4 \_ auto = 0 \qquad (5.80)$$

Avec la contrainte 5.60 pour rendre le sens équivalent à la commande, la suffisance des trois contraintes est vérifiée sur la propriété P1 avec la position interdite suivante :

$$\neg pouv \wedge \neg h_4 \wedge \neg b_4 \qquad (5.81)$$

2.3.5 Convoyage des palettes

Le déplacement des palettes sur le convoyeur peut aussi entrainer des dysfonctionnements par collision avec le préhenseur (Dp_3, Dp_9) ou entre les palettes (Dp_2).

2.3.5.1 Dysfonctionnements Dp_3, Dp_9

La collision avec le préhenseur peut avoir lieu soit par libération d'une palette alors que le préhenseur du poste suivant n'est pas en position haute, soit par libération d'une palette alors que le préhenseur n'est pas en position haute. Les deux premières contraintes sont définies pour éviter la collision entre la palette du poste i et le préhenseur du poste suivant :

- Interdire d'enlever la butée B_i si le préhenseur du poste suivant n'est pas en position h_j où $(i, j) = \{(1, 2) ; (3, 4)\}$

  $$B_i\_auto \wedge \neg h_j\_auto = 0 \qquad (5.82, 5.83)$$

- Interdire l'envoi des deux commandes en même temps

  $$B_i\_auto \wedge De_j\_auto = 0 \qquad (5.84, 5.85)$$

Les deux contraintes suivantes sont définies pour éviter la collision entre la palette et le préhenseur du même poste :

- Interdire d'enlever la butée B_j si le préhenseur du poste suivant n'est pas en position h_j où j=2, 4

  $$B_j\_auto \wedge \neg h_j\_auto = 0 \qquad (5.86, 5.87)$$

- Interdire l'envoi des deux commandes en même temps

  $$B_j\_auto \wedge De_j\_auto = 0 \qquad (5.88, 5.89)$$

- La vérification des contraintes passent par la vérification de l'atteignabilité de la position

$$pp_{ij} \wedge \neg h_j \wedge \neg b_j, \qquad (5.90, 5.91, 5.92)$$

Pour les poste 2 et 4 et pour les postes 1et 2 et les postes 3 et 4 dans la propriété P1. L'information pp_{ij} est générée par le modèle de PO.

2.3.5.2 Dysfonctionnement Dp_2

Le dernier dysfonctionnement à éviter, est la collision entre les palettes. Pour cela, une seule contrainte est définie pour chaque poste. En effet avant de libérer la palette, il faut vérifier s'il y a une palette sur le poste suivant :

- Interdire d'enlever la butée B_i s'il y a une palette sur le poste i et sur le poste suivant j où $(i, j) = \{(1, 2) ; (2, 3) ; (3, 4)\}$

$$B_i \_ auto \wedge ppp_i \_ auto \wedge ppp_{i+1} \_ auto = 0 \qquad (5.93, 5.94, 5.95)$$

Pour vérifier la suffisance de cette contrainte, la position définie par les équations 5.96 ne doit pas être atteinte.

$$pp_{ij} \wedge ppp_j \qquad (5.96)$$

Cette propriété a été vérifiée sous UPPAAL.

L'information pp_{ij}_auto est à reconstruire comme indiquer sur la figure suivante.

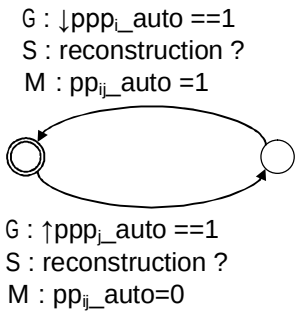


Figure 5.8. Reconstruction de l'information de la palette entre les postes i et j

Les 89 contraintes définies ici permettent d'assurer la sécurité du système et sont placées dans le filtre de validation système. En fonction de l'application, il faut ajouter des contraintes pour prendre en compte les spécifications du cahier des charges. La partie suivante présente les activités proposées autour de la machine PRODUCTIS lors de manifestations à destination des enfants ou lors de TP pour des étudiants débutants en commande.

2.4 Applications réalisées sur la PRODUCTIS

Le système sécurisé au niveau le plus bas est utilisé par des étudiants automaticiens « experts » en Master Pro « Automatisation et Supervision » dans le cadre de projets. Le filtre

robuste de commandes installé dans l'API permet aux étudiants de programmer en toute sécurité et de façon autonome la machine PRODUCTIS.

Une idée importante de la thèse, est la mise à disposition de systèmes automatisés pour des apprenants novices tout en conservant une vision globale de l'installation. Pour cela, l'enseignant doit définir le champ d'actions adapté à l'apprenant et les contraintes à définir au niveau du filtre fonctionnel.

Les trois expérimentations présentées rentrent dans ce cadre. Les objectifs pédagogiques qui ont conduit à la définition du champ d'actions, les contraintes à ajouter au niveau fonctionnel et les retours d'expérience sont successivement détaillés. Les deux premières activités sont à destination d'automaticiens « en herbe » car il s'agit d'enfants et la troisième est une utilisation dans le cadre de TP avec des étudiants de GCE qui débutent en commande.

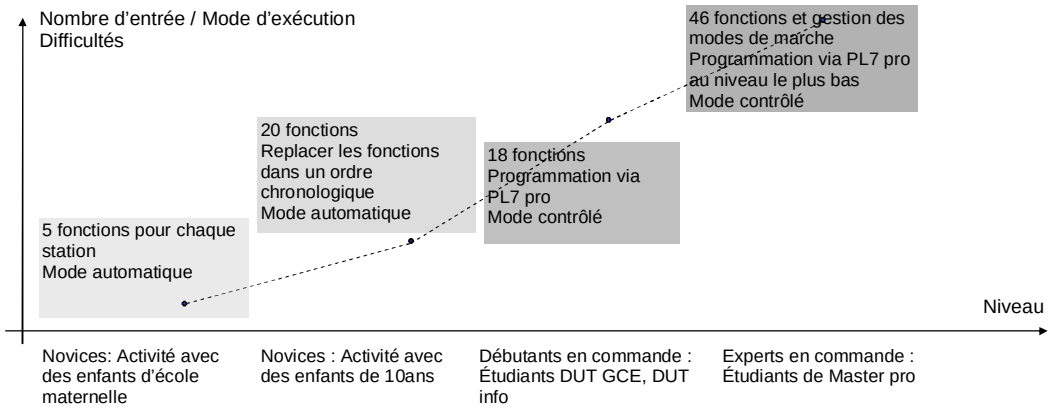


Figure 5.9. Différentes activités avec le degré de difficulté

Pour chaque activité, les connaissances de l'apprenant ont été prises en compte pour adapter le degré de difficulté (figure 5.9.). La première activité s'adresse à des enfants de 4-5 ans où ils découvrent simplement l'interaction entre une action sur un PC et la réaction engendrée sur le système (Riera et al., 2005). La seconde activité a été proposée à des enfants de 8 – 10 ans et l'activité va plus loin dans la découverte des systèmes automatisés et de leur commande (Marangé et al., 2007 b). Cette fois-ci, l'enfant est capable de replacer dans l'ordre chronologique une succession d'opérations. Pour cela, il utilise une interface lui permettant de créer une séquence pour fabriquer un flacon de médicaments. Cette expérience a été enrichissante tant pour les étudiants que pour les enfants. Cette approche est complémentaire au travail de l'association «la main à la pâte». En faisant venir les enfants dans nos salles de TP pour leur faire découvrir les sciences et la technique, on démystifie le monde universitaire et celui de l'EEA (Électronique, Électrotechnique, Automatique). De plus, ces événements ont

été suivis par la presse locale et plusieurs articles élogieux ont été publiés (cf. figure 5.13.), donnant ainsi une image originale et positive de nos formations.

En revanche, les deux autres activités relèvent de l'enseignement universitaire. Il s'agit de TP avec des étudiants de DUT GCE qui débutent en commande et d'autres en fin d'études.

Le but du TP avec les étudiants de GCE est de travailler sur différentes structures de Grafcet et sur les notions de synchronisation entre Grafcet, via un logiciel dédié à la programmation d'API.

Les étudiants en fin d'études d'automatisme, doivent quant à eux réaliser la commande complète, avec ses différents modes de marche et d'arrêt, de la machine PRODUCTIS.

2.4.1 Activités pour les 4 - 5 ans

Les objectifs pédagogiques (Riera et al., 2005) ont été définis par les enseignantes en écoles maternelles et peuvent être résumés de la façon suivante :

- Faire découvrir aux enfants un objet technique complexe, son usage et son utilisation,
- Préciser les parties, les gestes, l'organisation, les enchaînements de l'objet technique,
- Explorer des objets graphiques « programmés»,
- Faire prendre conscience aux enfants que derrière l'IHM (Interface Homme-Machine) peut se trouver le monde réel,
- Distinguer des formes, des couleurs et des nombres.

Les enseignantes ont aussi insisté sur la nécessité de scénariser les activités avec des enfants de 4–5 ans. Le travail pour définir le champ d'actions de l'enfant s'est fait en collaboration avec une institutrice de maternelle pour utiliser un vocabulaire adéquat et adapté à l'enfant, par exemple parler de barrière au lieu de butée, et définir une activité adaptée à leurs connaissances. Suite à ces discussions, Il a été décidé de définir 9 fonctions (figure 5.10.) qui s'exécutent en mode automatique :

- Mettre N billes vertes ($N \in \{1, 6\}$) (F_1), Mettre M billes blanches ($M \in \{1, 6\}$) (F_2)
- Mettre un gros bouchon (F_3),
- Mettre un petit bouchon (F_4),
- Enlever la barrière poste i ($i \in \{1, 4\}$) (F_5, F_6, F_7, F_8),

- Évacuer le flacon (F_9).

Ces 9 fonctions définissent 9 Grafcets qui s'exécutent lorsque l'enfant le demande et si les conditions d'activation sont présentes et s'arrêtent automatiquement.

Niveau 0	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Conditionner des médicaments dans des flacons	Mettre N billes vertes	Distribuer N billes vertes (F_1)	Distribuer 1 bille verte	Rot_1+	
				Rot_1-	
		Libérer palette (F_5)	B_1		
	Mettre M billes blanches	Distribuer M billes blanches (F_2)	Distribuer 1 bille blanche	Rot_3+	
				Rot_3-	
		Libérer palette (F_6)	B_3		
	Mettre un gros bouchon (F_3)	Positionner le préhenseur au dessus du bouchon	De_2		
		Maintenir le bouchon	Asp_2		↑Asp_2
					↓Asp_2
		Positionner le préhenseur au dessus du flacon	Mo_2		
			Av_2		
			De_2		
			Re_2		
		Ejecter le bouchon	$Eject_2$		
	Libérer palette (F_7)	B_2			
	Mettre un petit bouchon (F_4)	Positionner le préhenseur au dessus du bouchon	De_4		
		Maintenir le bouchon	Asp_4		↑Asp_4
					↓Asp_4
		Positionner le préhenseur au dessus du flacon	Mo_4		
			Av_4		
			De_4		
			Re_4		
		Ejecter le bouchon	$Eject_4$		
	Libérer palette (F_8)	B_4			
	Évacuer le flacon (F_9)	Positionner le préhenseur sur le flacon	De_4		
		Fermer la pince	Fer		
		Positionner le préhenseur au dessus de l'évacuation	Mo_4		
			Av_4		
			De_4		
			Re_4		
		Ouvrir la pince	Ouv		
		Libérer palette	B_4		

Figure 5.10. Champs d'actions pour les enfants de 4-5ans

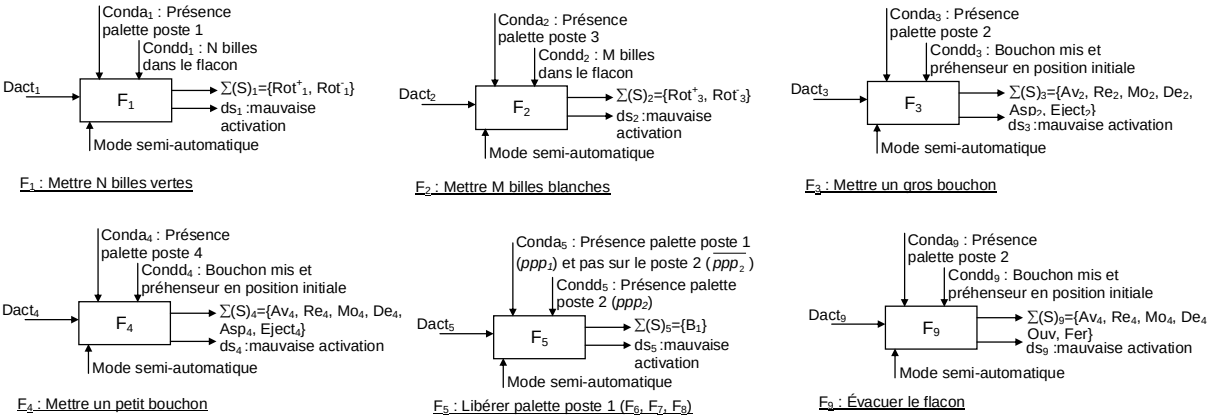


Figure 5.11. Blocs de fonction pour l'activité avec les enfants de 4-5 ans

La figure 5.11. donne la définition de chaque bloc fonction pour l'activité avec les enfants de 4 - 5 ans.

Cette activité a été mise en place durant plusieurs fêtes de la science avec des écoles maternelles et primaires de Reims. C'est l'occasion de mettre en place un projet avec des étudiants de licence professionnelle « Supervision et Traçabilité » pour développer les morceaux de commandes nécessaires aux fonctions ainsi que les Interfaces Homme-Machine (IHM) de supervision à destination des enfants. L'interface s'adresse à des enfants de 4-5 ans qui ne savent encore pas lire, les étudiants ont fait un effort important pour rendre l'interface la plus didactyle possible. Lorsque l'interface contient du texte, ce sont les animateurs qui le lisent aux enfants. La figure 5.12. donne plusieurs exemples d'interfaces proposées. Chaque année, un thème nouveau (des crocodiles malades, des ours, des fées, …) est choisi et il faut chaque fois trouver un scénario pour rendre l'activité ludique pour les enfants qui découvrent ainsi la « magie » de l'automatique.

a) Interfaces pour distribuer les billes

b) Interfaces pour mettre le bouchon

c) Interface pour évacuer le flacon d) Interface pour libérer les palettes

Figure 5.12. Interfaces

Pour cette activité, aucune contrainte n'est à ajouter car les fonctions s'exécutent en mode semi automatique et les conditions d'exécution sont simplement vérifiées lors de la transition initiale des Grafcets.

REIMS

Une idée de Génie

Des élèves de maternelle ont découvert l'intérêt, voire la magie, de la technique scientifique : ils ont piloté une ligne automatisée... pour soigner des crocodiles. Un concept du département « Génie du conditionnement et de l'emballage » à l'IUT de Reims.

Brevet de pilote

A disposition

Observation du cheminement des flacons.

Au secours des crocodiles

Figure 5.13. Article de presse

2.4.2 Activités pour les 8 - 10 ans

Pour s'approcher un peu plus du pilotage de systèmes automatisés de production, une seconde activité a été développée pour des enfants de 8 à 10 ans (Marangé et al. 2007 b) qui doivent réaliser un programme séquentiel simple pour commander la PRODUCTIS. Une fois encore, l'idée est de permettre à des novices de piloter la machine dans sa globalité. En collaboration avec une institutrice, à partir de l'analyse fonctionnelle de la machine, vingt fonctions ont été définies (figure 5.14.), avec un vocabulaire adéquat :

- Pour distribuer des billes : Mettre une bille verte (F_1) et Mettre une bille blanche (F_2)

- Pour mettre le bouchon : Avancer la tête (F_3, F_8), Reculer la tête (F_4, F_9), Monter la tête (F_5, F_{10}), Descendre la tête (F_6, F_{11}), Aspirer le bouchon (F_7, F_{12}) et Éjecter le bouchon (F_{19}, F_{20})
- Pour évacuer le flacon : Ouvrir la pince (F_{13}) et Fermer la pince (F_{14})
- Pour libérer la palette : Retirer l'obstacle du poste i (i∈{1, 4}) (F_{15}, F_{16}, F_{17}, F_{18})

Niveau 0	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Conditionner des médicaments dans des flacons	Mettre N billes vertes	Distribuer N billes vertes	Distribuer 1 bille verte (F_1)	Rot_1+	
				Rot_1-	
		Libérer palette	B_1 (F_{15})		
	Mettre M billes blanches	Distribuer M billes blanches	Distribuer 1 bille blanche (F_2)	Rot_3+	
				Rot_3-	
		Libérer palette	B_3 (F_{17})		
	Mettre un gros bouchon	Positionner le préhenseur au dessus du bouchon	De_2		
		Maintenir le bouchon	Asp_2		↑Asp_2 (F_7)
					↓Asp_2 (F_{19})
		Positionner le préhenseur au dessus du flacon	Mo_2 (F_5)		
			Av_2 (F_3)		
			De_2 (F_6)		
			Re_2 (F_4)		
		Éjecter le bouchon	$Eject_2$		
	Libérer palette	B_2 (F_{16})			
	Mettre un petit bouchon	Positionner le préhenseur au dessus du bouchon	De_4		
		Maintenir le bouchon	Asp_4		↑Asp_4 (F_{12})
					↓Asp_4 (F_{20})
		Positionner le préhenseur au dessus du flacon	Mo_4 (F_{10})		
			Av_4 (F_8)		
			De_4 (F_{11})		
			Re_4 (F_9)		
		Éjecter le bouchon	$Eject_4$		
	Libérer palette	B_4 (F_{18})			
	Évacuer le flacon	Positionner le préhenseur sur le flacon	De_4		
		Fermer la pince	Fer (F_{14})		
		Positionner le préhenseur au dessus de l'évacuation	Mo_4		
			Av_4		
			De_4		
			Re_4		
		Ouvrir la pince	Ouv (F_{13})		
		Libérer palette	B_4		

Figure 5.14. Champs d'actions pour les 8-10 ans

La fonction d'aspiration *Asp* a été décomposée car il est difficilement imaginable et compréhensible pour un enfant de cet âge de faire des actions en parallèle. Donc la fonction « Aspirer le bouchon » représente l'activation de *Asp* et « Ejecter le bouchon » représente la désactivation et l'envoi de l'ordre éjecter. Pour l'enfant le fait que l'actionneur *Asp* soit monostable et qu'il doit être maintenu, est complètement transparent. De même pour l'action *Eject*, elle est transparente pour l'enfant car il est difficile pour lui de distinguer la conséquence d'arrêter d'aspirer et de demander d'éjecter.

La définition de chaque bloc fonction est donnée dans la figure 5.15. L'enfant gère seulement l'activation de la fonction car elle s'exécute en mode semi-automatique

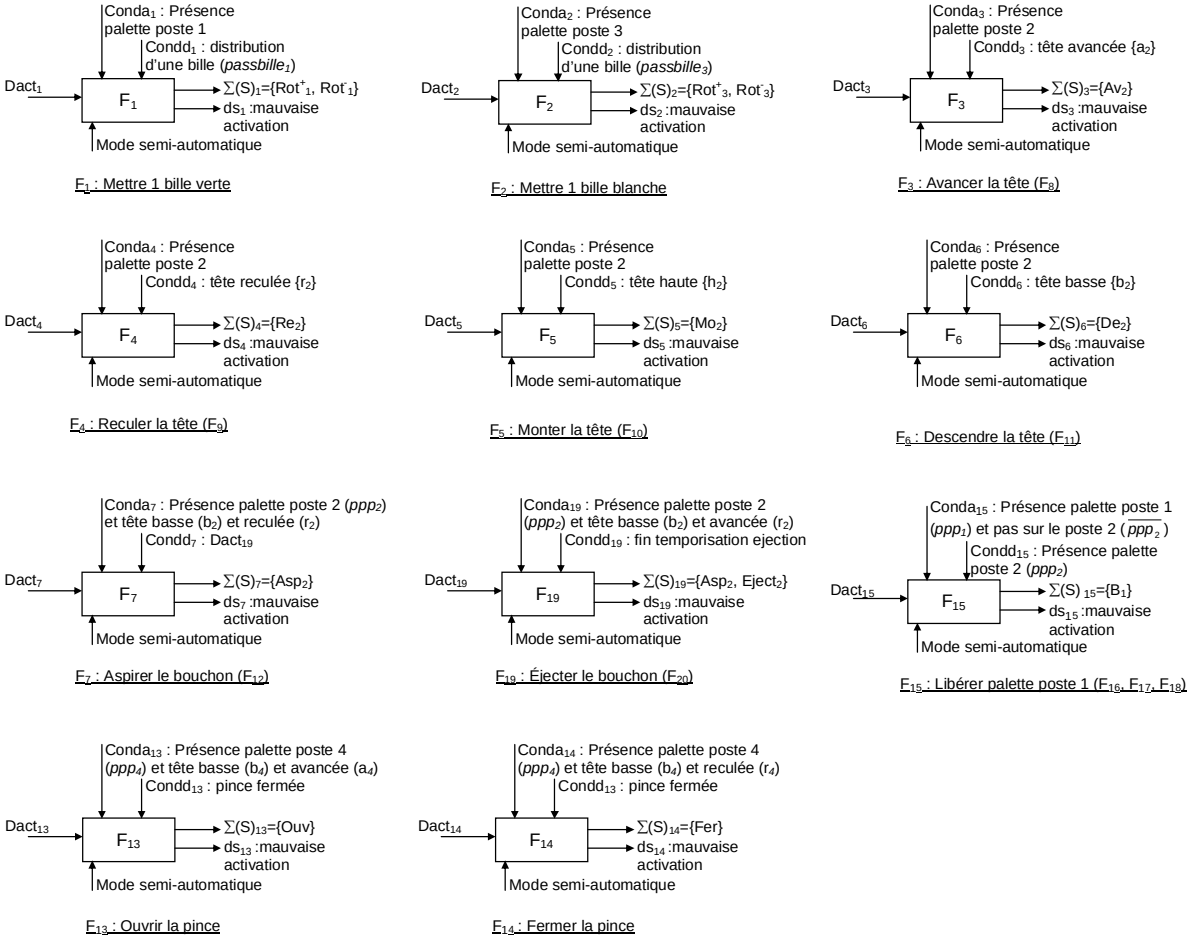
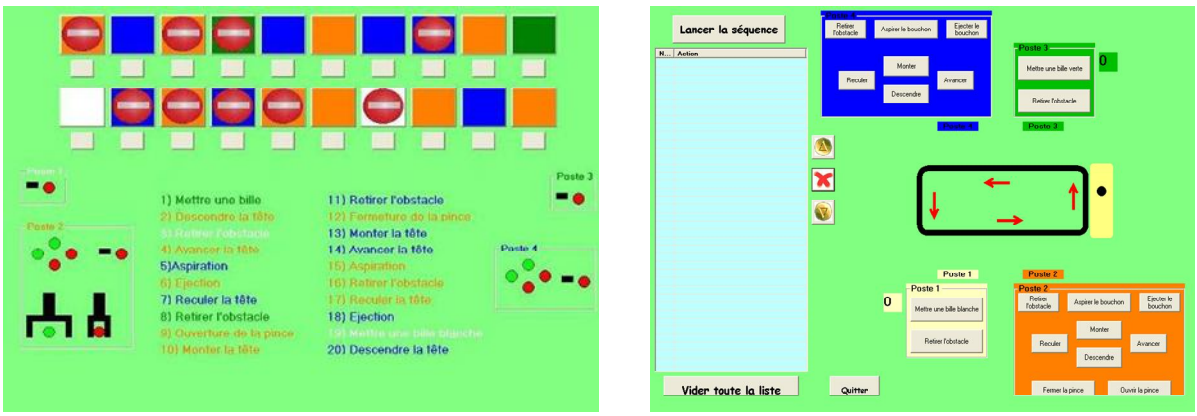


Figure 5.15. Blocs fonction pour l'activité avec les enfants de 8-10 ans

L'activité proposée aux enfants se déroule en deux temps : un premier où il découvre le système et un second où il propose une séquence d'actions pour constituer un flacon suivant divers scénarii. La première interface (figure 5.16.a) doit permettre à l'enfant de comprendre le rôle (i.e. la fonction) caché derrière chacun des boutons. L'enfant appuie sur un bouton de l'IHM et provoque un mouvement sur la machine. Ainsi, l'enfant peut associer une fonction à un bouton et comprendre l'effet de celui-ci sur le système. Dans cette activité, il faut indiquer à l'enfant les actions qu'il ne peut pas faire en fonction de la position du système. Cela se fait par l'intermédiaire d'un panneau « sens interdit » sur les boutons dont l'activation n'a aucun effet sur le système au vu de l'état du système. Par exemple, si le préhenseur est en position haute, l'activation du bouton monter n'aura aucun effet, donc un sens interdit apparaitra sur le bouton représentant « Monter la tête ».

Une fois que l'enfant a compris l'ensemble des fonctions, la seconde interface (figure 5.16.b) lui permet de programmer une séquence de mouvements pour réaliser un flacon de médicaments. Lors du déroulement de la séquence, avant l'exécution de chaque action, le filtre valide si les contraintes sont respectées ou non. Si elles le sont, l'action est exécutée sur le système et le filtre passe à l'action suivante. Sinon, le système est arrêté et l'enfant est informé de la ou les contraintes non respectées et, avec ou sans l'aide d'un encadrant, modifie sa séquence pour réaliser le flacon.



a) Mode « Pas à Pas »

b) Mode « Séquence »

Figure 5.16. Interfaces en Visual basic

L'implémentation de cette activité a été faite lors d'un projet de fin d'études par des étudiants de DUT Informatique. Les étudiants ont choisi de programmer les interfaces en Visual basic et la commande et le filtre sous PL7 Pro.

2.4.3 TP en GCE

La machine PRODUCTIS a aussi été utilisée dans le cadre de Travaux Pratiques (TP) conventionnels avec des étudiants de DUT GCE. Pour ce TP, l'étudiant doit apprendre à utiliser un logiciel dédié à la programmation d'API (PL7 Pro), à utiliser des structures de Grafcet vues en cours : ET, OU, synchronisation entre Grafcet. Pour ce TP, 18 fonctions ont été mises à sa disposition (figure 5.17.) :

- Pour les postes de distribution de billes : Mettre une bille verte (F_1) et Mettre une bille blanche (F_2)
- Pour mettre le bouchon : Avancer la tête (F_3, F_8), Reculer la tête (F_4, F_9), Monter la tête (F_5, F_{10}), Descendre la tête (F_6, F_{11}), Aspirer (F_7, F_{12})
- Pour évacuer le flacon : Ouvrir la pince (F_{13}) et Fermer la pince (F_{14})

- Pour libérer la palette (F_{15}, F_{16}, F_{17}, F_{18})

Niveau 0	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Conditionner des médicaments dans des flacons	Mettre N billes vertes	Distribuer N billes vertes	Distribuer 1 bille verte (F_1)	Rot_1+	
				Rot_1-	
		Libérer palette (F_{15})	B_1		
	Mettre M billes blanches	Distribuer M billes blanches	Distribuer 1 bille blanche (F_2)	Rot_3+	
				Rot_3-	
		Libérer palette (F_{16})	B_3		
	Mettre un gros bouchon	Positionner le préhenseur au dessus du bouchon	De_2		
		Maintenir le bouchon	Asp_2 (F_7)		↑Asp_2
					↓Asp_2
		Positionner le préhenseur au dessus du flacon	Mo_2 (F_5)		
			Av_2 (F_3)		
			De_2 (F_6)		
			Re_2 (F_4)		
		Ejecter le bouchon	$Eject_2$		
	Libérer palette	B_2 (F_{17})			
	Mettre un petit bouchon	Positionner le préhenseur au dessus du bouchon	De_4		
		Maintenir le bouchon	Asp_4 (F_{12})		↑Asp_4
					↓Asp_4
		Positionner le préhenseur au dessus du flacon	Mo_4 (F_{10})		
			Av_4 (F_8)		
			De_4 (F_{11})		
			Re_2 (F_9)		
		Ejecter le bouchon	$Eject_4$		
	Libérer palette	B_4 (F_{18})			
	Evacuer le flacon	Positionner le préhenseur sur le flacon	De_4		
		Fermer la pince	Fer (F_{14})		
		Positionner le préhenseur au dessus de l'évacuation	Mo_4		
			Av_4		
			De_4		
			Re_4		
		Ouvrir la pince	Ouv (F_{13})		
		Libérer palette	B_4		

Figure 5.17. Champs d'actions pour le TP en GCE

L'étudiant ne gère pas l'actionneur permettant l'éjection du bouchon, car la dimension du système est suffisante avec 18 fonctions. Il pourrait aussi paraître surprenant qu'il y ait moins de fonctions que pour l'activité avec les enfants de 8 – 10 ans, mais la difficulté est plus importante. Cela vient du fait de la gestion de la structure en ET entre l'aspiration et le mouvement du préhenseur. Dans le cas de l'activité avec les enfants, la commande est une succession linéaire d'actions. De plus le logiciel de programmation des API est plus compliqué et moins ludique. L'étudiant est informé de ses erreurs en lui remontant la ou les contraintes non respectées. Toutefois, la présence de l'enseignant reste nécessaire car les explications fournies concernant la contrainte qui n'est pas respectée sont relatives aux capteurs et aux actionneurs et ne suffisent pas forcément pour corriger le programme de commande.

La définition des blocs fonction est donnée par la figure 5.18. Les fonctions sont exécutées en mode contrôlé, c'est-à-dire que l'étudiant doit gérer l'activation et la désactivation de la fonction.

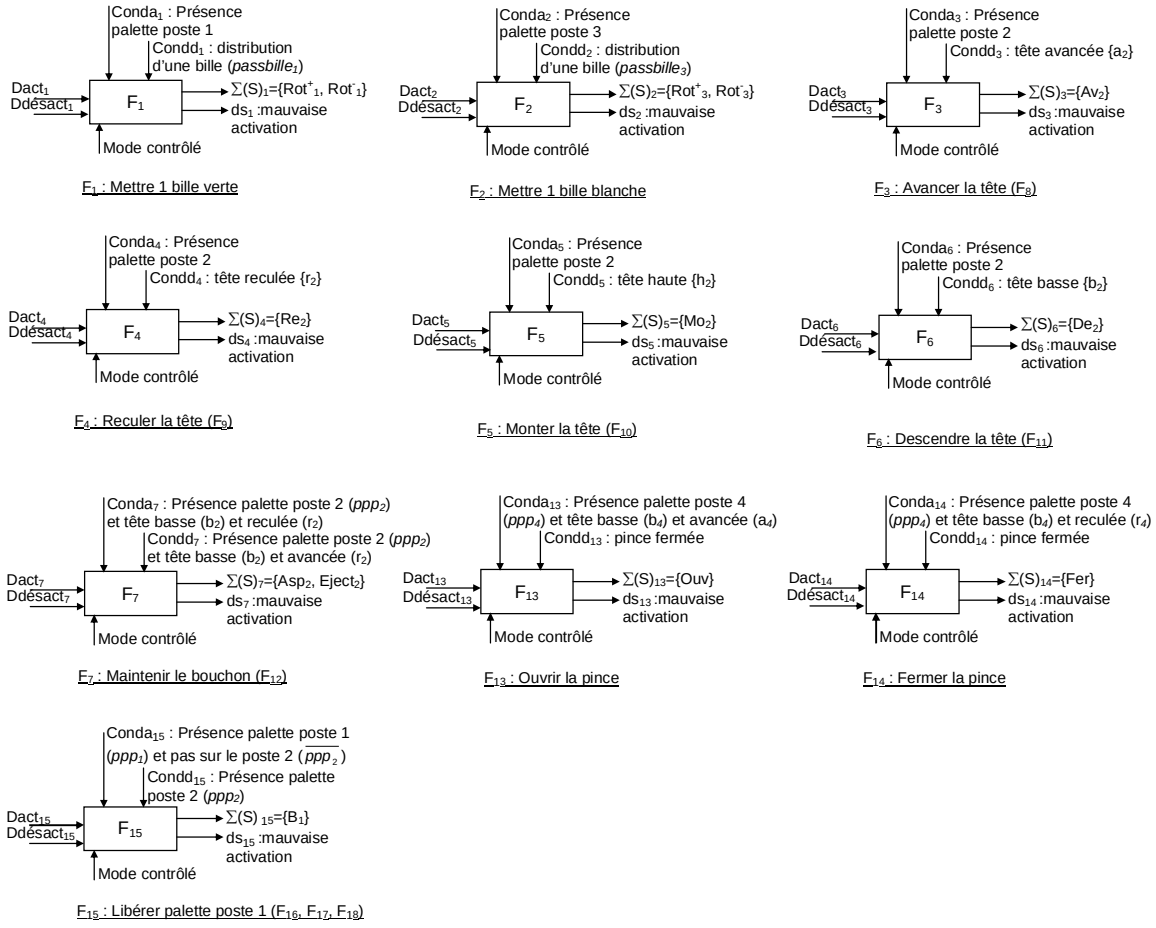


Figure 5.18. Blocs fonction pour le TP avec les étudiants de GCE

Dans le cadre de cette application, le filtre de validation fonctionnelle n'a pas été implémenté car cela ne nous est pas apparu indispensable.

L'implémentation du filtre de validation système s'est faite sous PL7 Pro : logiciel qui est utilisé par les étudiants pour programmer leur commande. Un fichier contenant la configuration matérielle du système et le filtre de validation est donné à l'étudiant lors du TP. La figure 5.19. montre le positionnement du filtre de validation. Les programmes des fonctions (F_1, F_2) et l'ensemble des contraintes sont programmées en Ladder dans une section à part appelée « *contraintes* » et qui s'exécute après le programme de l'étudiant (figure 5.20.). Chaque contrainte est représentée au moyen d'un bit interne de l'API.

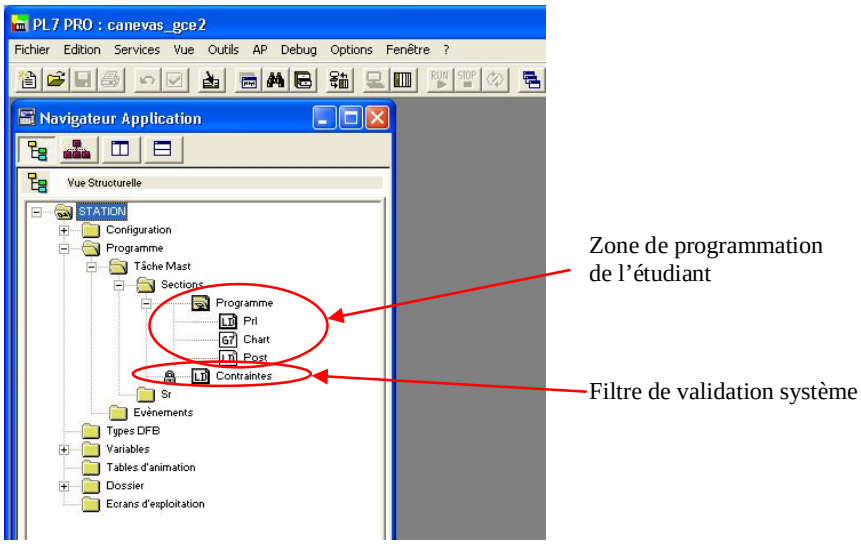


Figure 5.19. Implémentation filtre de sécurité sous PL7 Pro

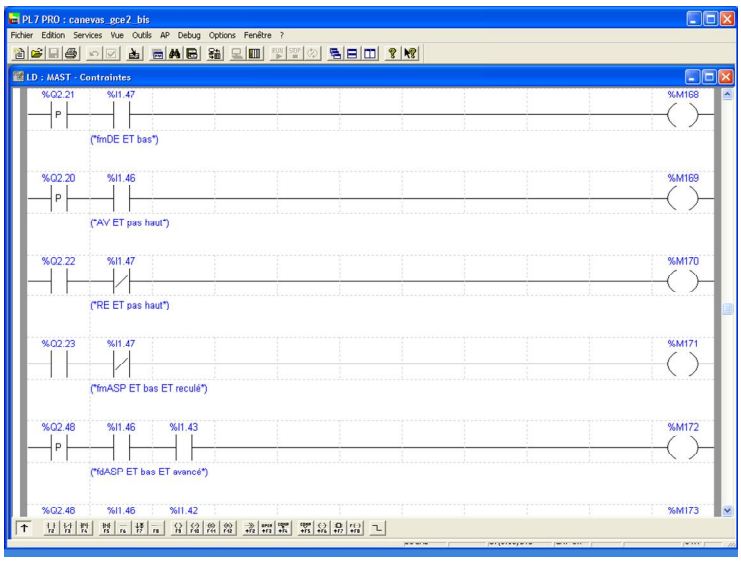


Figure 5.20. Implémentation des contraintes dans le filtre sous PL7 Pro

De cette manière, l'étudiant travaille directement sur les variables de sortie (%Qi) et si l'évolution qu'il propose n'est pas correcte par rapport aux contraintes, à la fin du bloc « *contraintes* » toutes les sorties sont forcées à 0 et un bit d'erreur est mis à 1 (figure 5.21.).

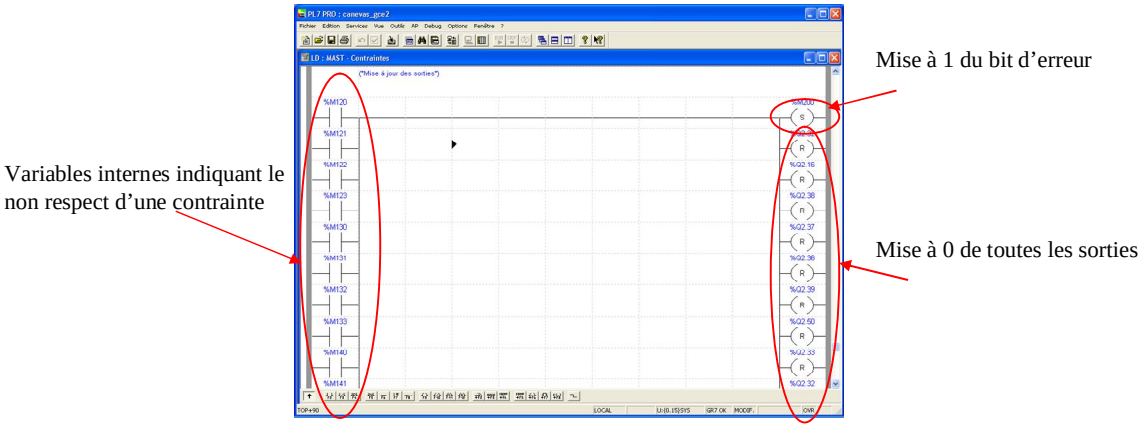


Figure 5.21. Mise à jour des sorties en cas d'erreurs sous PL7 Pro

Une interface de supervision (figure 5.22.) a été développée pour apporter une explication à l'étudiant en cas d'erreurs. Lorsqu'une contrainte n'est pas respectée une variable interne (%Mi) est mise à 1. Cette variable est associée à une LED sur l'interface de supervision, indiquant ainsi la contrainte qui est violée.

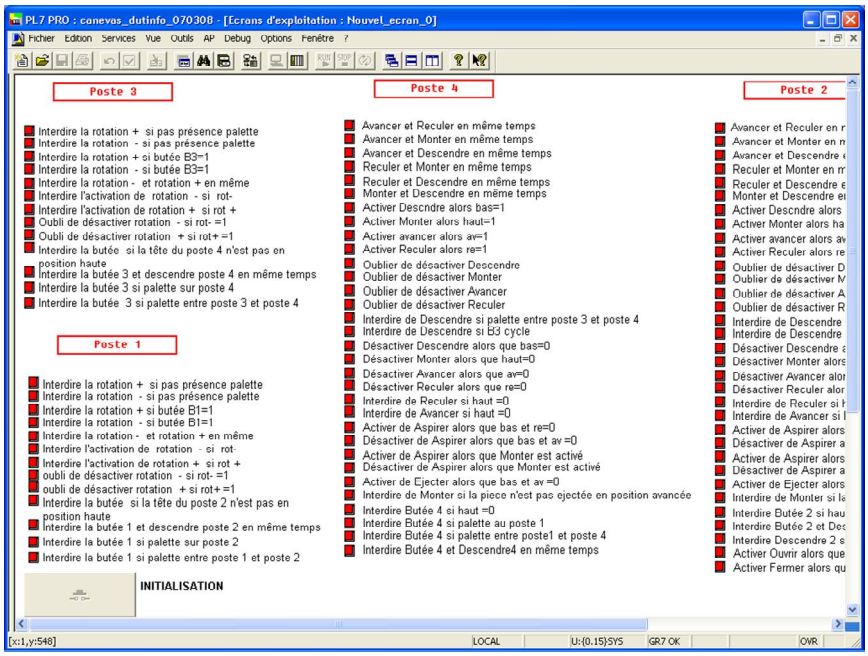


Figure 5.22. Interface pour expliquer les erreurs sous PL7 Pro

2.5 Conclusion sur les applicatifs avec la machine PRODUCTIS

L'application sur des maquettes réelles dans le cadre pédagogique, permet de faire prendre conscience à l'apprenant de la réalité des systèmes pilotés derrière une commande. Que cela soit avec les enfants ou avec les étudiants, les premiers retours sont très intéressants.

Au niveau des étudiants, ils sont moins anxieux de proposer une commande en sachant que leurs erreurs sont filtrées et ils sont plus intéressés car ils travaillent sur un système réel complet qui correspond aux installations qu'ils sont susceptibles de retrouver dans leur vie professionnelle.

Le filtre de commande robuste est donc opérationnel. Il permet à toutes les personnes travaillant sur la commande de la PRODUCTIS de le faire en toute sécurité. C'est le cas des projets de Master pro où les étudiants travaillent en autonomie complète.

3 Application sur un système virtuel

L'approche proposée pour valider la commande peut être aussi utilisée sur un système simulé ou virtuel. Son rôle premier n'est plus de sécuriser le système, mais d'apporter une explication en cas d'erreur. Une application a été faite sur un système virtuel de magasin automatique du logiciel ITS PLC (Real Games, 2008).

ITS PLC Professional Edition est un logiciel éducatif adapté à l'apprentissage de la programmation des API. Basé sur les dernières technologies informatiques, ITS PLC propose cinq PO simulées : système de tri, mélangeur « batch » 3 cuves, robot « Pick & Place », palettiseur et magasin automatisé

Il s'agit en fait d'environnements virtuels très réalistes grâce d'une part à une totale interactivité, et d'autre part aux animations graphiques 3D en temps réel, des dynamiques et des sons. Le résultat est un environnement simulé composé de plusieurs systèmes très réalistes pouvant être connectés à un API bien réel.

Les informations sont échangées entre l'API et la PO virtuelle au moyen d'un module USB d'acquisition de données TOR disposant de 16 entrées et de 16 sorties (figure 5.23.) connectées respectivement au coupleur de sorties et au coupleur d'entrées de l'API. Cette solution permet un temps de réponse rapide de la simulation et surtout une transparence pour l'étudiant qui perçoit la PO simulée au travers d'entrées/sorties bien réelles de l'API.

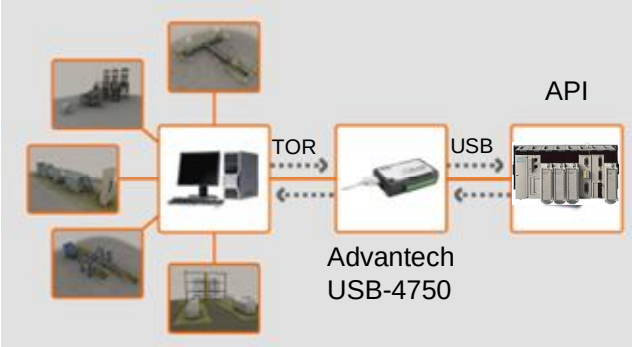


Figure 5.23. ITS PLC

3.1 Description du système

Ce système de magasin automatique est composé d'un transtockeur, d'un rack, d'un quai d'alimentation et d'un quai de sortie. Le quai d'alimentation apporte des caisses par un monorail automatique au transtockeur. Les caisses sont ensuite déposées ou récupérées par les fourches dans une des 50 positions du rack. Lorsque le transtockeur récupère une caisse, il va la déposer sur le quai de sortie.

Figure 5.24. Magasin automatique

Le système est constitué de 8 entrées et 8 sorties énumérées dans le tableau suivant :

	Variables système	Variables API
Transstockeur	I0, I1	I0_auto, I1_auto
	O0, O1, O2, O3, O4, O5	O0_auto, O1_auto, O2_auto, O3_auto, O4_auto, O5_auto
Fourche	I2, I3, I4, I5	I2_auto, I3_auto, I4_auto, I5_auto
	O6, O7	O6, _auto O7_auto
Monorail d'entrée	I6	I6_auto
Monorail de sortie	I7	I7_auto

Figure 5.25. Variables système/ API du magasin automatique

Les entrées du système sont :

- Transtockeur devant le quai d'entrée : *I0*
- Capteur de non mouvement du transtockeur : *I1*
- Fourches sorties vers les quais : *I2*
- Fourches en position rentrée : *I3*
- Fourches sorties vers le rack: *I4*
- Capteur de mouvement vertical des fourches: *I5*
- Monorail sur le quai d'entrée : *I6*
- Monorail sur le quai de sortie : *I7*

Le transtockeur et les fourches sont pilotés par les sorties suivantes :

- 6 encodeurs pour définir la position à atteindre : *O0*, *O1*, *O2*, *O3*, *O4*, *O5*
- Sortir les fourches vers les quais : *O6*
- Sortir les fourches vers le rack : *O7*

Le transtockeur est piloté par 5 encodeurs qui peuvent être vus comme une fonction selon la définition donnée dans ce manuscrit. En effet, le concepteur ne pilote pas directement le déplacement du transtockeur en lui disant de se déplacer vers la droite, vers la gauche, vers le haut ou vers le bas, mais en indiquant la position finale désirée.

3.2 Analyse du système

La fonction globale du système est de gérer un stock automatiquement, c'est-à-dire qu'il faut stocker et déstocker des paquets. Pour cela, il faut pour stocker, déplacer le transtockeur vers la position d'entrée, charger le produit, déplacer le transtockeur vers la position désirée et enfin décharger le paquet. Pour déstocker, c'est la même succession d'étapes pour déstocker.

Pour ce système, l'analyse fonctionnelle (figure 5.26.) amène à considérer sept dysfonctionnements dangereux :

- Dp1 : Déplacement du transtockeur alors que la fourche n'est pas rentrée, cela peut produire la détérioration des fourches.
- Dp2 : Chargement d'un nouveau produit sur le monorail d'entrée, alors qu'il y a déjà un produit dans le transtockeur, risquant de faire tomber l'un des deux produits.

- Dp3 : Mouvement des fourches du mauvais côté soit vers le rack au lieu du monorail, soit vers le monorail au lieu du rack, entrainant une perte du produit.
- Dp4 : Mouvement du transtockeur pendant le chargement du produit entrainant le mauvais chargement du produit.
- Dp5 : Arrêt du transtockeur entre deux racks qui va provoquer lors de la sortie des fourches une détérioration de celles-ci. Il est à noter qu'il n'y a pas de capteur pour indiquer en face de quel rack, le transtockeur est arrêté.
- Dp6 : Déchargement d'un produit sur un rack où un produit est déjà présent entrainant la chute de l'un des deux produits.
- Dp7 : Déchargement d'un produit sur le monorail d'entrée, ce qui peut entrainer la chute d'un produit.

Décomposition			Actionneurs	Défaillances possibles
Gérer un magasin automatique	Stocker un produit dans le magasin	Aller à la position d'entrée	O0, O1, O2, O3, O4, O5,	Dp1 : Mouvement avec les fourches pas rentrées
		Charger le produit du monorail d'entrée sur le transstockeur	O6	Dp2 : Un produit est déjà présent sur le transstockeur Dp3 : Les fourches sont sorties dans le mauvais sens Dp4 : Le transstockeur est en mouvement pendant le chargement
		Aller à la position désirée	O0, O1, O2, O3, O4, O5,	Dp5 : Transstockeur arrête en position intermédiaire entre 2 positions du rack (Dp1)
		Déposer le produit sur le rack	O7	Dp6 : Un produit est déjà présent à cet emplacement (Dp4, Dp5)
	Déstocker un produit du magasin	Aller à la position désirée	O0, O1, O2, O3, O4, O5,	(Dp1, Dp2)
		Charger le produit du rack sur le transstockeur	O7	(Dp3, Dp4, Dp6)
		Aller à la position de sortie	O0, O1, O2, O3, O4, O5,	(Dp1, Dp2)
		Décharger le produit	O6	Dp7 : Décharger le produit sur le quai d'entrée (Dp3, Dp4, Dp5)

Figure 5.26. Analyse du magasin automatique

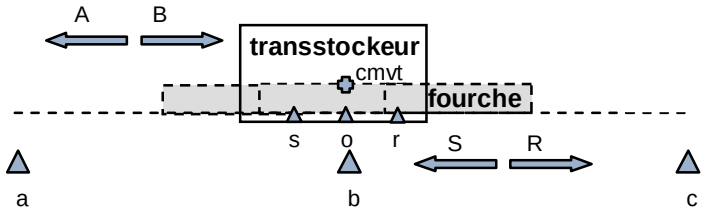
Pour éviter ces sept dysfonctionnements, un ensemble de contraintes a été défini en considérant le problème sur un modèle simplifié. En effet, si les contraintes sont correctement définies pour un emplacement et pour une seule dimension de déplacement du transtockeur, elles le seront aussi une fois généralisées à tous les emplacements et aux deux dimensions.

3.3 Définition des contraintes

L'étude de ce système peut se faire plus simplement, en ne considérant qu'un seul axe de déplacement pour le transtockeur et une seule position dans le stock. Le magasin simplifié est vu de la manière suivante (figure 5.27.) :

- Le mouvement du transtockeur est réduit à une dimension
- Une position dans le rack est définie : *b*
- Les positions d'entrées et de sorties sont représentées par deux capteurs : *a* et *c*
- La fourche sort vers r (*R*) pour charger le produit en *a* et le décharger en *c*
- La fourche sort vers s (*S*) pour charger ou décharger un produit du rack (*b*)
- Le capteur de mouvement du transtockeur est représenté par *cmvt*
- Le déplacement vers a s'effectue par la commande *A* et vers *c* par *B*
- Le capteur de mouvement vertical de la fourche n'est utile que pour la commande

La figure 5.27. présente le modèle simplifié du magasin automatique.



a) schéma simplifié du magasin automatique

	Variables système	Variables API
Transstockeur	a, b, c, cmvt, A, B	a_auto, b_auto, c_auto, cmvt_auto A_auto, B_auto
Fourche	s, o, r S, R	s_auto, o_auto, r_auto S_auto R_auto

b) variables utilisées

Figure 5.27. Modèle simplifié du magasin automatique

Pour définir les contraintes, chaque défaillance possible va être considérée, ensuite la suffisance de ces contraintes sera vérifiée.

3.3.1 Dp1 : Mouvement avec les fourches pas rentrées

Les six contraintes suivantes permettent d'éviter le déplacement du transtockeur alors que les fourches ne sont pas rentrées :

- Interdiction d'activer *A* si les fourches ne sont pas en *o*

$$\uparrow A\_auto \wedge \neg o\_auto = 0 \qquad (5.97)$$

- Interdiction d'activer *B* si les fourches ne sont pas en *o*

$$\uparrow B\_auto \wedge \neg o\_auto = 0 \qquad (5.98)$$

- Interdiction de désactiver *S* si les fourches ne sont arrivées en *s,* ce qui rend équivalent le sens et la commande

$$\downarrow S\_auto \wedge \neg s\_auto = 0 \qquad (5.99)$$

- Interdiction de désactiver *R* si les fourches ne sont arrives en *r* ce qui rend équivalent le sens et la commande

$$\downarrow R\_auto \wedge \neg r\_auto = 0 \qquad (5.100)$$

- Interdiction d'avoir *A* en même temps que *R* ou *S*

$$A\_auto \wedge (S\_auto \vee R\_auto) \qquad (5.101)$$

- Interdiction d'avoir *B* en même temps que *R* ou *S*

$$B\_auto \wedge (S\_auto \vee R\_auto) \qquad (5.102)$$

3.3.2 Dp2 : Chargement de deux produits dans le transtockeur

Pour empêcher de charger un nouveau produit du monorail d'entrée, alors qu'il y a déjà un produit dans le transtockeur, il faut définir la contrainte suivante : Interdiction d'activer *S* si le transtockeur est en position *a* (quai d'alimentation) et si un produit est déjà présent sur le transtockeur :

$$\uparrow S\_auto \wedge a\_auto \wedge produit\_ds\_trans\_auto = 0 \qquad (5.103)$$

A cette contrainte, il faut aussi ajouter l'interdiction d'activer *R* s'il y a un produit dans le transtockeur, et un produit à la position *b* et si le transtockeur se trouve en *b* :

$$\uparrow R\_auto \wedge b\_auto \wedge produit\_en\_b\_auto \wedge produit\_ds\_trans\_auto = 0 \qquad (5.104)$$

Il faut reconstruire deux informations :

- la présence d'un produit dans le transtockeur *produit_ds_trans_auto* : peut se faire soit par un chargement ($\uparrow s$) sur le quai d'alimentation (*a*), soit par le chargement ($\uparrow r$) d'un produit (*produit_en_b_auto*) dans le rack (*b*). Il n'y aura plus de produit dans le transtockeur soit parce qu'il aura été déposé ($\uparrow r$) en *b* ou ($\uparrow s$) sur le quai de sortie (*c*)
- la présence d'un produit en b (*produit_en_b_auto*) : est issue de la dépose ($\uparrow r$) d'un produit présent sur le transtockeur (*produit_ds_trans_auto*) lorsqu'il est en *b*. Il n'y aura plus de produit lorsque le transtockeur chargera ($\uparrow r$) le produit en *b*.

3.3.3 Dp3 : Mouvement des fourches dans le mauvais sens

Pour éviter un mouvement des fourches du mauvais côté soit vers le rack au lieu du monorail, soit vers le monorail au lieu du rack, deux contraintes sont définies :

- Interdiction d'activer *S* si le transtockeur n'est ni en *a* ni en *c*

$$\uparrow S\_auto \wedge \neg a\_auto \wedge \neg c\_auto = 0 \qquad (5.105)$$

- Interdiction d'activer *R* si le transtockeur n'est pas en *b*

$$\uparrow R\_auto \wedge \neg b\_auto = 0 \qquad (5.106)$$

3.3.4 DP4 : Mouvement du transtockeur pendant le chargement

Pour éviter de charger ou de décharger, lorsque le transtockeur est en mouvement, il faut définir, les contraintes suivantes qui testent si le capteur de mouvement *cmvt* est à 0 :

- Interdiction d'activer *S* si le capteur de mouvement *cmvt* est à 1

$$\uparrow S\_auto \wedge cmvt\_auto = 0 \qquad (5.107)$$

- Interdiction d'activer *R* si le capteur de mouvement *cmvt* est à 1

$$\uparrow R\_auto \wedge cmvt\_auto = 0 \qquad (5.108)$$

3.3.5 Dp5 : Le transtockeur est arrêté en position intermédiaire

Il n'y a pas de capteur pour indiquer les emplacements sur le rack. Pour éviter que le transtockeur s'arrête entre deux positions, il faut lui permette de s'arrêter seulement lorsque le capteur de mouvement est à 0 :

- Interdiction de désactiver *A* si le capteur de mouvement *cmvt* est à 1

$$\downarrow A\_auto \wedge cmvt\_auto = 0 \qquad (5.109)$$

- Interdiction de désactiver *B* si le capteur de mouvement *cmvt* est à 1

$$\downarrow B\_auto \wedge cmvt\_auto = 0 \qquad (5.110)$$

3.3.6 Dp6 : Un produit est déjà présent à cet emplacement

Le déchargement d'un produit sur un rack où un produit est déjà présent est interdit pour éviter la chute de l'un des deux. Pour cela, la contrainte suivante est définie :

$$\uparrow R\_auto \wedge b\_auto \wedge produit\_en\_b\_auto = 0 \qquad (5.111)$$

3.3.7 Dp7 : Décharger un produit sur le quai d'alimentation

La contrainte pour éviter de décharger un produit sur le monorail d'entrée revient à lui interdire de recharger un second produit (Dp2). La contrainte interdisant d'activer *S* si le transtockeur est en position *a* (quai d'alimentation) et si un produit est présent sur le transtockeur, interdit aussi de décharger sur le quai d'alimentation.

$$\uparrow S\_auto \wedge a\_auto \wedge produit\_ds\_trans\_auto = 0 \qquad (5.112)$$

Une fois ces 15 contraintes définies, elles doivent être vérifiées en fonction des situations qui ne doivent pas être atteintes.

3.4 Vérification de la suffisance des contraintes

Pour vérifier la suffisance de l'ensemble de contraintes, trois points doivent être traités :

- Les positions atteignables par le transtockeur pour assurer l'impossibilité des dysfonctionnements (Dp1, Dp3, Dp4, Dp5)
- L'impossibilité d'avoir deux produits dans le transtockeur (Dp2, Dp7)
- L'impossibilité d'avoir deux produits à l'emplacement *b* (Dp6)

Pour vérifier le premier point, la propriété P1 est utilisée avec l'équation des états interdits (figure 5.28.): où *z* représente la position intermédiaire entre *o* et *r*

$$m = ((\neg s \wedge \neg o \wedge \neg z \wedge b) \vee (\neg r \wedge \neg o \wedge z \wedge a) \vee (\neg r \wedge \neg o \wedge z \wedge c) \vee (\neg o \wedge \neg a \wedge \neg b \wedge \neg c) \qquad (5.113)$$

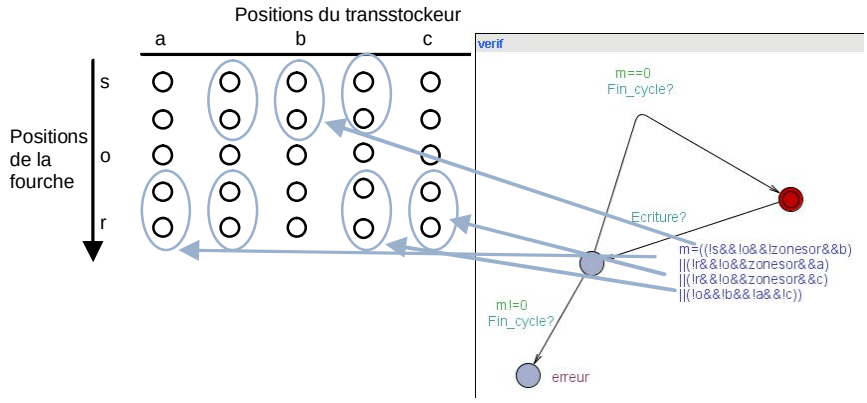


Figure 5.28. Modèle de vérification sous UPPAAL

Pour vérifier le second point, la seconde propriété est utilisée avec les modèles de produit suivant : *pièce_en_a*, *pièce_en_déplacement_depuis_a*, *suivi_pièce_b*, *pièce_en_déplacement_depuis_b*, *suivi_ pièce_c* et pour effectuer la vérification : le modèle de produit *suivi_2produits_en_b* est défini. Dans ce dernier modèle, un état a été défini comme interdit (figure 5.29.) et représente le fait de valider les conditions pour charger un second produit.

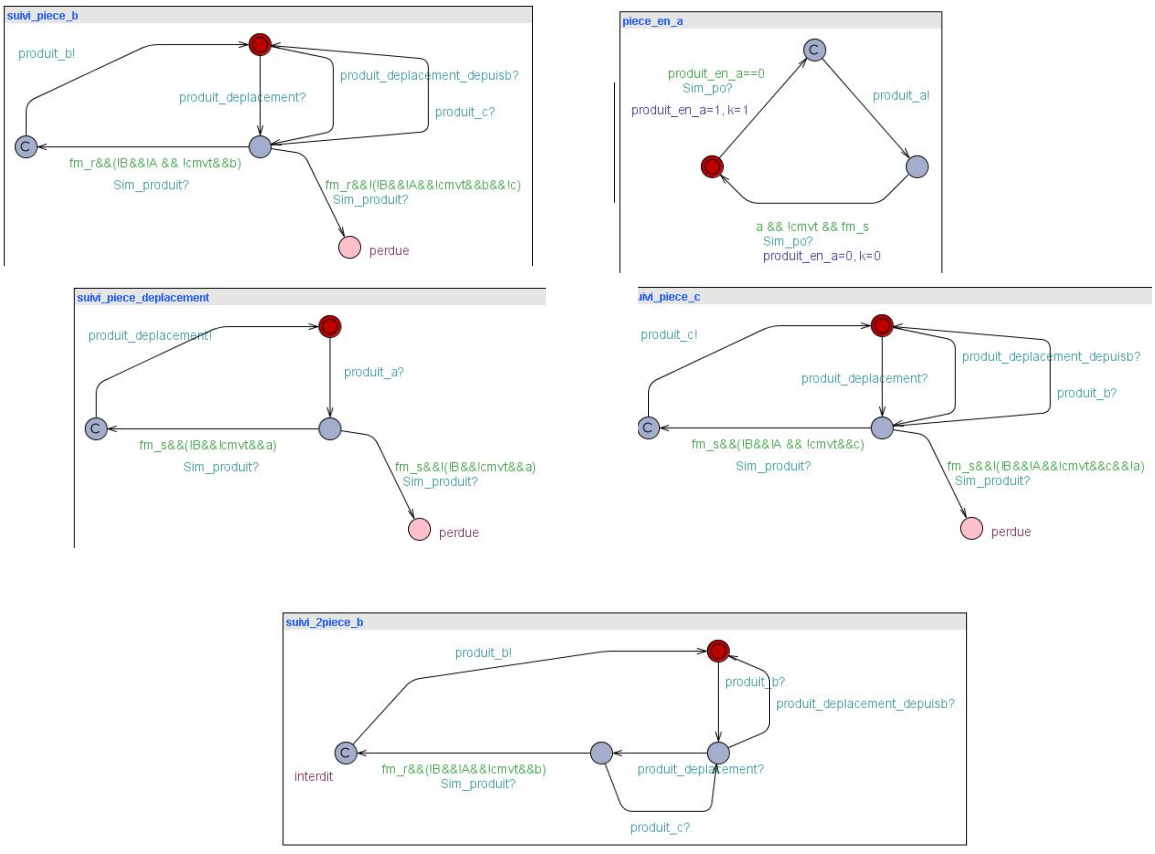


Figure 5.29. Modèle de produit sous UPPAAL

Le troisième point est vérifié de la même manière que le point précédent avec la propriété P2. La possibilité de mettre plusieurs produits sur le transtockeur et la possibilité de déposer un produit sur le quai d'alimentation, sont vérifiées en même temps. Les mêmes modèles de produit sont utilisés, seul le modèle pour effectuer la vérification change. Cette fois-ci, il est testé s'il est possible de mettre 2 produits sur le transtockeur avec le modèle *2produits_en_déplacement* suivant

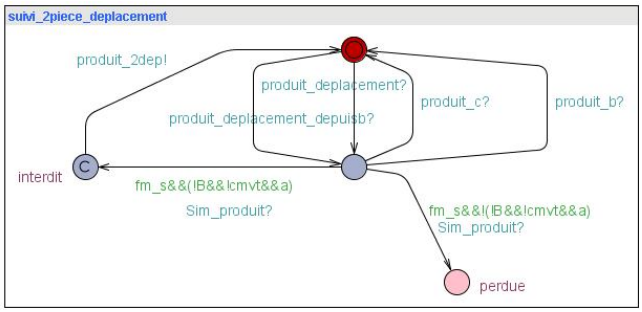


Figure 5.30. Modèle de produit sous UPPAAL

3.5 Généralisation des contraintes

Les contraintes ont été vérifiées pour le modèle simplifié, elles sont maintenant généralisées au système global pour être implémentées :

- $\uparrow A\_auto \wedge \neg o\_auto = 0$: Interdiction d'activer A si les fourches ne sont pas en o $\uparrow B\_auto \wedge \neg o\_auto = 0$: Interdiction d'activer B si les fourches ne sont pas en o

$\uparrow O0\_auto \wedge \neg I3\_auto = 0$ *(5.114);* $\uparrow O1\_auto \wedge \neg I3\_auto = 0$ *(5.115);*

$\uparrow O2\_auto \wedge \neg I3\_auto = 0$ *(5.116);* $\uparrow O3\_auto \wedge \neg I3\_auto = 0$ *(5.117);*

$\uparrow O4\_auto \wedge \neg I3\_auto = 0$ *(5.118) ;* $\uparrow O5\_auto \wedge \neg I3\_auto = 0$ *(5.119)*

- $\downarrow S\_auto \wedge \neg s\_auto = 0$: Interdiction de désactiver S si les fourches ne sont pas arrivées en s

$\downarrow O6\_auto \wedge \neg I2\_auto = 0$ *(5.120)*

- $\downarrow R\_auto \wedge \neg r\_auto = 0$: Interdiction de désactiver R si les fourches ne sont pas arrivées en r

$\downarrow O7\_auto \wedge \neg I4\_auto = 0$ *(5.121)*

- $A\_auto \wedge (S\_auto \vee R\_auto)$: Interdiction d'avoir *A* en même temps que *R* ou *S* et $B\_auto \wedge (S\_auto \vee R\_auto)$: Interdiction d'avoir *B* en même temps que *R* ou *S*

$$O0\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.122)$$

$$O1\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.123)$$

$$O2\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.124)$$

$$O3\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.125)$$

$$O4\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.126)$$

$$O5\_auto \wedge (O6\_auto \vee O7\_auto) \quad (5.127)$$

- $\downarrow A\_auto \wedge cmvt\_auto = 0$: Interdiction de désactiver *A* si le capteur *cmvt* est à 1 et $\downarrow B\_auto \wedge cmvt\_auto = 0$: Interdiction de désactiver *B* si le capteur *cmvt* est à 1

$$\downarrow O0\_auto \wedge I1\_auto = 0 \quad (5.128); \qquad \downarrow O1\_auto \wedge I1\_auto = 0 \quad (5.129)$$

$$\downarrow O2\_auto \wedge I1\_auto = 0 \quad (5.130); \qquad \downarrow O3\_auto \wedge I1\_auto = 0 \quad (5.131)$$

$$\downarrow O4\_auto \wedge I1\_auto = 0 \quad (5.132); \qquad \downarrow O5\_auto \wedge I1\_auto = 0 \quad (5.133)$$

- $\uparrow S\_auto \wedge a\_auto \wedge produit\_ds\_trans\_auto = 0$: Interdiction d'activer *S* si le transtockeur est en position *a* (quai d'alimentation) et si un produit est déjà présent sur le transtockeur

$$\uparrow O6\_auto \wedge I0\_auto \wedge produit\_ds\_trans\_auto = 0 \quad (5.134)$$

- $\uparrow S\_auto \wedge (\neg c\_auto \wedge \neg a\_auto) = 0$: Interdiction d'activer *S* si le transtockeur n'est ni en *a* ni en *c* et $\uparrow R\_auto \wedge \neg b\_auto = 0$: Interdiction d'activer *R* si le transtockeur n'est pas en *b*

$$\uparrow O6\_auto \wedge (\neg p10\_auto \wedge \neg IO\_auto) = 0 \quad (5.135)$$

$$\uparrow O7\_auto \wedge (\neg p1\_auto \wedge \neg p2\_auto \wedge ... \wedge \neg p50\_auto) = 0 \quad (5.136)$$

- $\uparrow S\_auto \wedge cmvt\_auto = 0$: Interdiction d'activer *S* si le capteur de mouvement *cmvt* est à 1 et $\uparrow R\_auto \wedge cmvt\_auto = 0$: Interdiction d'activer *R* si le capteur de mouvement *cmvt* est à 1

$$\uparrow O6\_auto \wedge cmvt\_auto = 0 \quad (5.137)$$

$$\uparrow O7\_auto \wedge cmvt\_auto = 0 \quad (5.138)$$

- $\uparrow R\_auto \wedge b\_auto \wedge produit\_en\_b\_auto = 0$: Interdire d'activer R si le transtockeur est en position b et si un produit est déjà présent en b

$$\uparrow O7\_auto \wedge p1\_auto \wedge produit\_en\_p1\_auto = 0 \quad (5.139)$$

$$\uparrow O7\_auto \wedge p50\_auto \wedge produit\_en\_p50\_auto = 0 \quad (5.140)$$

La reconstruction des informations pour cet exemple peut paraître assez lourde. En effet, il faut reconstruire les 50 positions avec ou sans produit (*produit_en_pi_auto*), les 50 positions (*pi_auto*) et l'information de la présence d'un produit dans le transtockeur.

Pour cet exemple, il faut au total définir 75 contraintes. Celui-ci montre une limite dans la reconstruction d'informations et la lourdeur de l'approche, dans le cas de la gestion de stock si le nombre de capteurs est insuffisant. Cependant, l'approche reste simple dans la détermination des informations à reconstruire et dans la définition des contraintes. Le filtre de commande robuste a été implémenté dans un automate Premium de la même façon que pour l'application sur la PRODUCTIS. Nous l'avons testé avec succès au moyen de commandes erronées. Il sera complètement validé car utilisé à partir de septembre 2008 en TP et en auto-formation par les étudiants.

4 Conclusion

Ce chapitre a présenté deux applicatifs, dans un cadre pédagogique, de l'approche de validation de la commande par filtre, ainsi que la vérification de la robustesse de celui-ci vis-à-vis de la commande la plus permissive possible. Le premier applicatif porte sur l'utilisation d'un système réel de conditionnement de médicaments. Le filtre assure son rôle premier de protection du système face aux erreurs pouvant occasionner des défaillances, mais il permet aussi d'apporter une explication à l'apprenant en cas d'erreur. Plusieurs applications ont été proposées aussi bien à de jeunes enfants de 4-5 ans qu'à des étudiants de BAC+2 à BAC+5. Chaque fois, une perception globale de la machine a été proposée.

Le second applicatif concerne une partie opérative simulée. Il s'agit du magasin automatisé du logiciel ITS PLC. Cette fois-ci, le filtre a juste un rôle explicatif en cas d'erreur. Cet exemple est intéressant dans la gestion du produit et dans la définition des sorties permettant le déplacement du transtockeur. Toutefois, il montre les limites de la méthode proposée en particulier pour la gestion de stock.

Ces 2 exemples ont clairement montré l'intérêt, l'originalité et la faisabilité de la validation de commande par filtre robuste. Cependant, le travail réalisé nécessite d'être poursuivi et fait l'objet de perspectives qui vont être présentées avant de conclure ce mémoire.

Conclusion générale et perspectives

Les travaux présentés dans ce manuscrit se sont intéressés aux problématiques posées lorsqu'un système manufacturier est mis à la disposition d'automaticiens non nécessairement expérimentés et/ou susceptibles de commettre des erreurs dans la conception de la commande pouvant mettre en danger l'installation. Les questions essentielles auxquelles nous avons tenté de répondre sont les suivantes :

- Comment assurer la sécurité face aux erreurs de commande ?
- Comment prendre en compte la composante humaine (automaticien) ?
- Comment garantir que l'approche mise en place est sûre de fonctionnement ?

La figure 1 présente un résumé de notre contribution qui a tenté d'apporter des éléments de réponses théoriques, méthodologiques et pratiques à chacune de ces 3 questions.

Pour assurer la sécurité du système, deux approches de validation ont été proposées. La première utilise les concepts de synthèse. En effet, à partir des spécifications qui doivent être respectées et du modèle de la PO, le comportement contraint est obtenu. La validation de la commande se fait alors par comparaison entre le comportement précédemment obtenu et le comportement décrit par la commande proposée par le concepteur. La deuxième approche utilise les concepts de surveillance de système par filtre. Celui-ci est placé entre la PO et l'API, et à chaque évolution de la commande, on vérifie que les spécifications fonctionnelles et de sécurité sont validées. Dans cette thèse nous avons principalement travaillé sur la proposition d'un filtre robuste de commande garantissant la sécurité de la PO indépendamment du cahier des charges.

L'utilisation pédagogique de systèmes industriels nous a conduit à prendre en compte la composante humaine (automaticien, concepteur, apprenant, expert, enseignant, …) en ayant une vision systémique de l'automatisation. L'idée n'a pas été de retirer l'expert ou encore d'assister le concepteur en réalisant la commande à sa place, mais plutôt de leur fournir des outils méthodologiques ou explicatifs adaptés à leur tâche. Une des idées majeures a été de considérer qu'il était important, quel que soit l'apprenti automaticien, de conserver une vision globale de la PO. Pour cela, une vision fonctionnelle hiérarchique permet la définition par l'expert du champ d'actions du concepteur. La structure du filtre proposé offre cette possibilité.

La sécurité du système dans les deux approches est basée sur la formalisation des spécifications. Si celles-ci ne sont pas correctement définies, la détection des erreurs sera erronée et pourra entraîner des détériorations du système. Pour s'assurer que les spécifications sont correctement définies, une approche de vérification de la suffisance pour la sécurité de l'ensemble de contraintes a été proposée. Celle-ci est définie de telle manière à ne faire aucune hypothèse sur la commande proposée et revient à concevoir un filtre « robuste » aux erreurs de conception. Pour cela, une approche de vérification formelle par model-checking a été développée. Le principe consiste à vérifier qu'un modèle de PO (basé sur la chaîne fonctionnelle) n'atteint jamais de situations dangereuses quelle que soit la commande. Le fonctionnement cyclique de l'API ainsi que le retard de causalité induit ont été pris en compte au travers de différents modèles représentés sous la forme d'automates à états communicants. En vue d'éviter une explosion combinatoire, la PO est analysée au moyen d'une approche modulaire où la PO est vue comme des ensembles d'éléments de PO et de produits en interaction, indépendants les uns des autres.

L'adaptation du système au concepteur et le filtrage « robuste » de la commande ont été illustrés au moyen de deux applications. La première concerne un système « réel » de conditionnement de médicaments appelé PRODUCTIS. La seconde a été réalisée en utilisant une simulation « virtuelle » de magasin automatisé. Ces réalisations ont permis de tester et de montrer l'intérêt, mais aussi les limites, des propositions faites dans la thèse concernant la sécurisation et l'adaptation.

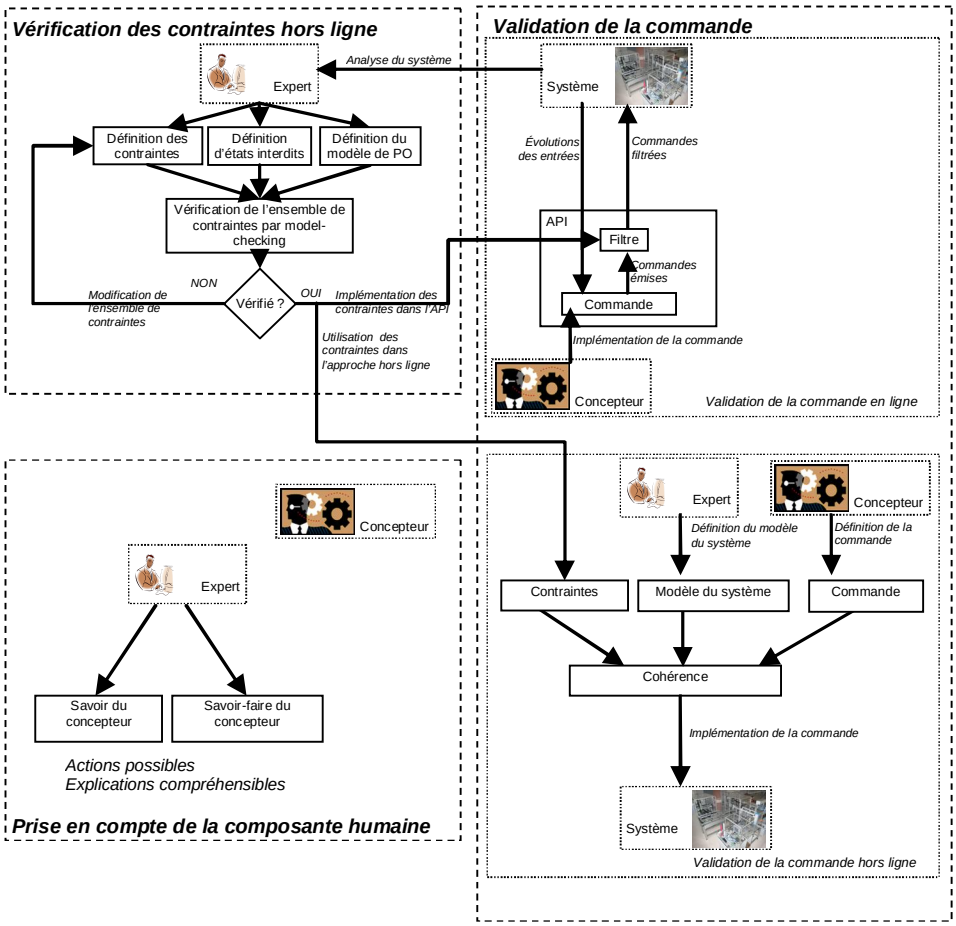


Figure 1. Résumé de notre contribution

Le travail réalisé nécessite d'être poursuivi en vue de lever les limites que nous avons soulignées tout au long du manuscrit.

L'approche de validation de la commande par synthèse est difficilement applicable sur des systèmes réels du fait de l'explosion combinatoire et de la modélisation du système global. De plus, dans sa version actuelle, elle ne tient pas compte de la granularité de la PO pour prendre en compte les connaissances du concepteur. En vue de lever ces limitations, dans la suite des travaux pour cette approche, il serait envisageable de :

- Proposer un modèle de PO non plus au niveau capteurs/actionneurs mais au niveau fonctionnel permettant d'obtenir ainsi le comportement maximum admissible du cahier des charges et non, le comportement maximal admissible du procédé, avec la vision du concepteur. Les contraintes à définir représenteraient les spécifications du cahier des charges.

- Effectuer la synthèse de certaines parties du système peu complexes. Ces parties seraient implantées dans le système et les évolutions n'apparaissant pas dans ces modèles ne seront pas envoyées vers la PO.

Dans les travaux présentés dans cette thèse, uniquement le filtre « système » lié à la sécurité est vérifié formellement. Il serait intéressant par la suite de poursuivre les recherches sur la partie fonctionnelle du filtre et ses capacités explicatives. Pour cela, il faudrait :

- Prendre en compte les spécifications du cahier des charges. Ceci nécessite de modéliser des séquences d'évolution et leurs interactions. Ce point pourrait être pris en compte par exemple en utilisant l'algèbre d'Allen (Allen, 1983) spécifiant treize relations entre deux intervalles. Ce point a commencé à être étudié dans (Marangé et al., 2007 d) où il est proposé de transcrire les relations d'Allen par des sous-programmes de commande indiquant ainsi l'ordre d'exécution des séquences.
- Analyser plus finement les traces du model-checker UPPAAL pour fournir une explication facilement interprétable par l'automaticien. En effet, comme nous l'avons noté dans le chapitre 4 (paragraphe 2), une contrainte non respectée est une source d'explications car elle correspond à un ou plusieurs chemins conduisant à un état dangereux. Les traces fournies par UPPAAL doivent être synthétisées pour devenir un véritable outil de diagnostic de la commande pour le concepteur.
- Apporter une explication en adéquation avec le champ d'actions du concepteur. En effet, les explications remontées en cas d'erreur de sécurité, proviennent du filtre système et sont définies au niveau le plus bas. Une méthode pourrait regrouper les contraintes relatives à une même fonction pour générer une explication adéquate.

La recherche du sous-ensemble minimal de contraintes a été évoquée dans le chapitre 4 mais pas développée. De plus, nous avons fait plusieurs hypothèses concernant la dynamique des éléments de PO (ElPO) et l'unicité du produit. Ces deux points ouvrent les perspectives suivantes :

- Définir l'ensemble de contraintes nécessaires à la validation de la commande. En effet, l'approche de vérification de l'écriture des contraintes permet d'assurer que l'ensemble de contraintes est suffisant pour ne pas atteindre des états interdits.

Implémenter l'ensemble minimum de contraintes permet d'obtenir la solution la plus légère pour l'API en termes d'utilisation mémoire.

- Tenir compte des effets d'inertie dans la définition des contraintes. En effet, les travaux actuels font l'hypothèse que les capteurs sont à contact long pour que l'ElPO puisse s'arrêter si la demande d'arrêt est faite en début de déclenchement du capteur. La difficulté se place au niveau de la reconstruction du sens de déplacement. Il nous semble que la solution passe par une modélisation plus fine des phénomènes d'inertie et par là même, la prise en compte du temps.
- Lever l'hypothèse d'un seul produit par état produit. Pour cela, un nouveau modèle de produit devra être proposé utilisant un autre outil de modélisation. L'exemple du magasin automatique a montré que la modélisation de chaque produit dans le stock était lourde.
- Proposer une approche pour accompagner l'expert dans la définition des contraintes à partir du canevas. Nous avons proposé une première approche où l'expert définit, pour chaque interaction, les positions interdites dans le graphe de positions (Marangé et al., 2007 e, 2008 d). A chaque transition conduisant à un état interdit, une contrainte est définie. L'inconvénient majeur de cette approche, est qu'elle demande la construction d'un nombre important de modèles et n'est pas didactique.
- Développer une bibliothèque d'ensemble de contraintes pour des ElPO, des interactions d'ElPO ou des interactions avec le produit. Il serait envisageable de développer une base de données mettant à disposition des ElPO pour reconstruire le système. L'expert pourrait spécifier les interactions existantes et le logiciel générerait l'ensemble de contraintes.

Les applicatifs de cette thèse ont concerné la formation aux automatismes, de la découverte à la spécialisation. Il serait intéressant d'appliquer les approches de validation de la commande et de vérification sur un problème industriel pour justifier la faisabilité sur des systèmes complexes. A ce sujet, l'atelier flexible CellFlex (SIMATIC, 2008), opérationnel depuis octobre 2007 à l'Université de Reims Champagne-Ardenne est un démonstrateur idéal.

Bibliographie

A

Abrial J.R., *The B-Book, Assigning Programs to Meanings*, Cambridge University Press, ISBN 0521496195, 1996

Achour Z., Rezg N., *Commande par supervision des systèmes à événements discrets basée sur l'utilisation du Grafcet et la théorie des régions*, Journal européen des systèmes automatisés RS série JESA Vol. 38 N° 1, pp37-58, 2004

Adjallah K., Niang B., Ndiaye P., *Gestion de tâches de diagnostic à base d'un système distribué*, Colloque Francophone sur le thème : Performances et Nouvelles Technologies en Maintenance (PENTOM), 2005

AFNOR, *Terminologie de la maintenance – NF EN 13306*, 2001

AFNOR NF X 50-151, *Analyse de la Valeur, Analyse Fonctionnelle - Expression fonctionnelle du besoin et cahier des charges fonctionnel*, 1991

Alanche P, Lhoste P., Morel G., Roesh M., Salim M., Salvi P., *Application de la modélisation de la Partie opérative à la structuration de la commande*, Journée AFCET, Montpellier, 1986

Aliane N., Javier F.A., Martinez A., Fraile A., Ortiz J., *An internet-based control Engineering loboratory*, 7th IFAC Symposium on Advances in control Education (ACE'06), paper 120, Madrid, 2006

B

Bainbridge L., *Ironies of automation*, Automatica, vol. 19, N°6, pp 775-779, 1983

Bajic E., Les automatismes industriels : images, évolutions et perspectives, GESI, 2006

Balemi S., Hoffmann G.J., Gyugyi P., Wong-Toi H., Franklin G.F., *Supervisory control of a rapid thermal multiprocessor*, IEEE Transactions on Automatic Control, vol. 38, n°7, 1993, pp1040-1059.

Barragan Santiago I., *Élaboration de propriétés formelles de contrôleurs logiques à partir d'analyse prévisionnelle par arbre de défaillances*, Thèse de doctorat de l'ENS de Cachan, 2007

Behrmann G., Bengtsson J., David A., Larsen K.G., Pettersson P., Yi W., *Uppaal implementation secrets*, In Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems, 2002.

Belhimeur A., *Contribution à l'étude d'une méthode de conception des automatismes des systèmes de conduite des processus industriels*, Thèse de Doctorat en Automatique, Université des Sciences et Techniques de Lille, 1989

Bellmunt G.O., Montesinos Miracle D., Galcern Arellano S., Sudria Andreu A., *A distance PLC programming Course employing a remote laboratory based on a flexible manufacturing cell*, IEEE Trans. on Educ., vol.49, n°2, may 2006

Beney M., *Contribution à l'analyse des phases manipulatoires des travaux pratiques de physique en Premier Cycle universitaire. Les apprentissages possibles à travers la conduite de l'action*. Thèse de doctorat de l'Université Paris 11 Orsay, 1998

Berard B., Bidoit M., Finkel A., Laroussinie F., Petit A., Petrucci L., Schnoebelen P., *Systems and software verification: Model-checking techniques and tools*, Heidelberg, Springer-Verlag Edition, 1999.

Berry G., *Real time programming: special purpose or general purpose languages*, rapport de recherche INRIA, n°1065, Août 1989.

Bodart A., Carré-Ménétrier V., De Loor P., Deluche J.B., Dupont J., Gendreau D., Hancq J., Kril A., Nido J., *7 facettes du Grafcet*, collection Automatisation et production, Cépaduès édition, ISBN 2-8542-8461-5, mai 2000

Boehm B.W., *A spiral model of software development*, ACM, SIGSOFT, vol.11, n°4, 1986

Booch G., Rumbaugh J., Jacobson I., *The Unified Modelling Language User Guide*, The Addison-Wesley Object Technology Series, Addison-Wesley Publishing Company, ISBN 0-201-57168-4, 1998

Bossy J.C., Brard P., Faugère P., Merlaud C., *Le Grafcet, sa pratique et ses applications*, El educalivre, Edition Casteilla A., ISBN 2-71351-5513, 1989

Boulanger J.L., *Expression et validation des propriétés de sécurité logique et physique pour les systèmes informatiques critiques*, Thèse de doctorat de l'Université de Technologie de Compiègne, mai 2006

C

Carré-Ménétrier V., Zaytoon J., *Grafcet: behavioural issues and control synthesis*, European Journal of Control (EJC), vol. 8, n°4, pp375-401, 2002

Carré-Ménétrier V., Zaytoon J., *Démarche progressive d'implantation d'une commande sûre dans un API. A partir d'une spécification Grafcet*, Revue de l'Électricité et de l'Électronique R.E.E., n°3, pp69-79, mars 2000

Chaillet-Subias A., *Approche multi modèles pour la commande et la surveillance en temps réel des systèmes à événements discrets*, Thèse de doctorat de l'Université Paul Sabatier, Toulouse, décembre 1995

Chafik S., *Proposition d'une structure de contrôle par supervision hiérarchique et distribuée: application à la coordination*, Thèse de doctorat de l'Institut National des Sciences Appliquées de Lyon, décembre 2000

Chafik S., Niel. E., *Hierarchical decentralized solutions of supervisory control*, 3rd MATHMOD, volume2, pp787-790, Vienna, Austria, February 2000

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, volume 8, n° 1, pp49-73, 2002

Chandra V., Kumar R., *A Discrete Event Systems Modeling Formalism Based one event Occurrences Rules and Precedences*, IEEE Transactions one Robotics and Automation, vol.17, n°6, 2001

Chatelet E., *Sûreté de fonctionnement : méthodes et outils de base*, Université de technologie de Troyes édition, 2000.

Chatelet P., Jouga B., *Les systèmes d'automatismes et Internet, J'automatise*, chapitre 3, pp53-55., 1999

Chen P. P., *Entity-relationship approach to data modelling*, In System and Software Requirements Engineering, Thayer RH, Dorfman M (eds). Washington: IEEE.; 238-243, 1990

Chen P. P., *The entity-relationship model: towards a unified view of data*, ACM TODS, 1:9—36, 1976

Coad P., Yourdon E., *Object-Oriented Analysis*, Englewood Cliffs, New Jersey : Yourdon Press, Prentice Hall. 2 edition, 1991.

Colace F., De Santo M., Pietrosanto A., *Work in Progress - Virtual Lab for Electronic Engineering Curricula*, 34th ASEE/IEEE Frontiers in Education Conference, Savannah, October 2004

Combacau M., *Commande et surveillance des systèmes à événements discrets complexes : application aux ateliers flexibles*, thèse de doctorat de l'Université Paul Sabatier de Toulouse, 1991

Cruette D., *Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et à la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière*, thèse de doctorat de l'Université de Lille, février 1991

D

De Bonneval A., *Mécanismes de Reprise dans les Systèmes de Commande à Événements Discrets*, Thèse de doctorat de l'Université Paul Sabatier, Toulouse, Septembre 1993

Delaval G., Rutten E., *A Domain-Specific Language for Multitask Systems, Applying Discrete Controller Synthesis*, EURASIP Journal on Embedded Systems, vol. 2007, Article ID 84192, 17 pages, 2007.

Deschamps E., *Diagnostic de services pour la reconfiguration dynamique de systèmes à événements discrets complexes*, Thèses, Institut National Polytechnique de Grenoble – INPG, novembre 2007

E

El-Khattabi S., *Intégration de la surveillance de bas niveau dans la conception des systèmes à événements discrets : application aux systèmes de production flexibles*, Thèse de docteur de l'Université de Lille I, Septembre 1993

Endsley E. W., Almeida E. E., Tilbury D. M., *Modular finite state machines: development and application to reconfigurable manufacturing cell controller generation*, Control Engineering Practice, Vol. 14, pp1127-1142, 2006

Evrot D, *Contribution à la vérification d'exigences de sécurité : Application au domaine de la machine industrielle*, Thèse de doctorat de l'Université de Henri Poincaré, Nancy 1, juillet 2008

F

Faucher J., *Pratique de l'AMDEC*, Edition DUNOD, ISBN 2-100-06710-9, Paris 2004.

Faure J-M., Lesage J-J., *Methods for safe control systems design and implementations, 10th IFAC Symposium on Information Control Problems in Manufacturing*, INCOM'2001, Vienna (Austria), CD-Rom paper, 6 pages, September 2001

Forsberg K., Cotterman H., Mooz H., V*isualizing project management: Models and frameworks for mastering complex systems*, Wiley; 2 edition, ISBN 0-471-64848-5, 2005

Forsberg K., Mooz H., *The relationship of system engineering to the project cycle*, proceeding of the INCOSE Symposium, 1991

Frey G., Litz. L., *Formal methods in PLC programming*, Proceedings of the IEEE SMC 2000, pages 2431 2436, October 2000

G

Gaffé D., *Modélisation et vérification d'un système mécatronique par SyncCharts, MSR'03 Modélisation des systèmes réactifs*, pp 95-107, Metz, octobre 2003

Ghaffari A., Rezg N., Xie X., *Algebraic and geometric characterization of Petri net controllers using the theory of region*, Proceeding of 6th International Workshop on discrete Event Systems, WODES'02, Saragoza, Espagne, 2-4 octobre 2002

Ginestie J., *Contribution à la didactique des disciplines technologiques : acquisition et utilisation d'un langage d'automatisme*, Thèse de doctorat de l'Université d'Aix-Marseille 1, 1992

Gourguff V., *Représentations formelles efficaces pour l'aide à la certification de contrôleurs logiques industriels*, thèse de doctorat de l'ENS de Cachan, novembre 2007

Gouyon D., Pétin J.F., Gouin A., *A pragmatic approach for modular control synthesis and implementation*, International Journal of Production Research, vol. 42, n° 14, pp. 2839-2858, Taylor & Francis Publisher, ISSN 0020-7543, juillet 2004.

Guillon A., *Étude épistémologique et didactique de l'activité expérimentale en vue de l'enseignement et de l'apprentissage des démarches du physicien, dans le cadre des travaux pratiques de première et deuxième années d'Université*, Thèse de doctorat de l'Université de Paris 11 Orsay, 1996

H

Hassapis G., *An interactive electronic book approach for teaching computer implementation of industrial control systems*, IEEE Trans. on Educ., vol.46, n°1, February 2003

Hatley D.J., Pirbhai I.A., *Stratégies de spécification des systèmes temps réel (SA-RT)*, Editions Masson, Paris, ISBN 2-225-82229-8, 1991

Hellgren A., Lennartson B., Fabian M., *Modelling and PLC-based implementation of modular supervisory control*, Proceeding of 6th International Workshop on discrete Event Systems, WODES'02, Saragoza, Espagne, 2-4 octobre 2002

Hoc J.M, S*upervision et contrôle de processus : la cognition en situation dynamique*, Presse Universitaire de Grenoble, Collection Sciences et Technologie de la connaissance, 1996

Holloway L.E., Krogh B.H., *Fault detection and diagnosis in manufacturing systems: a behavioural model approach*, IEEE International Conference on computer Integrated Manufacturing, mai 1990

I

IEC *Automates programmables – Partie 3 : Langages de programmation - Programmable controllers – Part 3: Programming languages*, Norme Internationale / International Standard, CEI/IEC 61131-3, 2003.

IEC 61131-3 *Langage de spécification GRAFCET pour diagrammes fonctionnels en séquence – GRAFCET specification language for sequential function charts*, Norme Internationale / International Standard, CEI/IEC 60848, 94 pages, février 2002 a

IEC 61508, *Sécurité fonctionnelle des systèmes électriques / électroniques / électronique programmable relatifs à la sécurité*, 2002 b

IEC *Etablissement des diagrammes fonctionnels pour systèmes de commande - Preparation of function charts for control systems*, Norme Internationale / International Standard, CEI/IEC 60848, 1988

I.G.L. Technologie, *SADT - un langage pour communiquer*, ISBN-10: 2.21208.185.5 Paris : Eyrolles, 2006.

ISO/CEI Guide 51, *Aspects lies à la sécurité – Principes directeurs pour les inclure dans les normes*, 1999

ISO 8402, *Management de la qualité et assurance de la qualité*, vocabulaire, 1995

Ivanov A., Varnier C., Zerhouni N., *Ordonnancement des activités de maintenance dans un contexte distribué*, Conférence Francophone de Modélisation et Simulation MOSIM, 2003

J

Johannsen G., *25 years of Human-Machine Systems in IFAC*, 10th IFAC/IFIP/IFORS/IEA Symposium on analysis, Design, and Evaluation of human-Machine Systems, Seoul, September 2007

Julliand J., *Automates finis et applications*, Cour de DEA IAP, 2003

K

Kamach O., Chafik S., Pietrac L., Niel E., *Representation of a reactive system with different models*, IEEE International Conference on Systems, Man and Cybernetics (SMC'02), Hammamet, Tunisia, octobre2002

Khatab A., *Contrôle et contrôle stabilisant des systèmes à événements discrets : application au recouvrement des défaillances*, Thèse de doctorat de l'Institut National des Sciences Appliquées de Lyon, décembre 2000

Kowalewski S. and Preuig J., *Verification of sequential controllers with timing functions for chemical processes*, IFAC 13th World Congress, volume J, pages 419-424, San Francisco, USA, 1996

Kumar R., *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems*, Thesis for PhD. Degree, University of Texas, 1991

L

Laprie J.C., *Sûreté de fonctionnement des systèmes : concepts de base et terminologie*, Rapport LAAS N°04520, 2004.

Laprie J.C., Arlat J., Blanquart J-P., Costes A., Crouzet Y., Deswarte Y., Fabre J-C., Guillermain H., Kaaniche M., Mazet C., Powell D., Rabéjac C. et Thévenod. P., *Guide de la Sûreté de Fonctionnement*, 2ème édition (Cépaduès), ISBN : 9782854283822, 1995

Le Moigne, *La modélisation des systèmes complexes*, Edition Dunod, Paris, ISBN : 9782100043828, 1999

Lhoste P., *Contribution au génie automatique : concepts, modèles*, méthodes et outils, Habilitation à diriger des recherches de l'Université de Nancy, février 1994

Li Y., *Control of vector discrete-events systems*, PhD thesis, Department of electrical and computer engineering, University of Toronto, 1991

Lind, M., *Modelling Goals and Functions of Complex Industrial Plant*, Applied Artificial Intelligence, Vol.8 no.2, April-June 1994

Lottin J., Morteau S., *Le pendule inverse : un système hybride ?*, CETSIS'05, Nancy, France, octobre 2005

Lunt B.M., Helps H.G., Carter P., Red E., *Systems and automation education through Web-based labs*, ICEE 2000, Taiwan, august 2000

M

Machado J., Denis B., Lesage J.-J., Faure J.M., Ferreira Da Silva J.C.L, *Logic controllers dependability verification using a plant model*, DESDes06, pages 37-42, Rydzyna (Poland), September 2006 a

Machado J., *Influence de la prise en compte d'un modèle de processus en vérification formelle des systèmes à événements discrets*, Thèse de doctorat de l'école normale supérieure de Cachan et de l'Université de Minho (Portugal), juin 2006 b

Makungun M., Barbeau M., Saint Denis R., *Synthesis of controllers of processes modelled as colored Petri nets discrete event dynamics systems: theory and application*, Vol.9, pp147-169, Khuwer academic Publishers, 1999

Marangé P., Gellot F., Riera B., *Human-Machine Systems concepts applied to Control Engineering Education*, Ifac World Congress, Seoul, juillet 2008 a

Marangé P., Gellot F., Riera B., *Vérification des propriétés de sécurité pour la commande des systèmes à événements discrets*, GIS'08, juin 2008 b

Marangé P., Tajer A., Gellot F., Carré-Ménétrier V., *Étude du comportement global d'un SED en vue de la validation de sa commande spécifiée par Grafcet*, Hermès/Lavoisier, Journal Européen des Systèmes Automatisés JESA, 42(1):63-94, mars 2008 c

Marangé P., Gellot F., Riera B., *Application of the control validation to D.E.S. Teaching*, IJOE : International Journal of Online Engineering, February 2008 d

Marangé P., Gellot F., Riera B., *Remote control of automation systems for D.E.S. course*, revue IEEE Transaction on Industrial Electronics Special Section, pp3103-3111, December 2007 a

Marangé P., Gellot F., Riera B., Chemla J.P., *Utilisation adaptée au niveau des apprenants et validation de la commande des Systèmes Automatisés de Production*, CETSIS'07, Bordeaux, septembre 2007 b

Marangé P., Gellot F., Riera B., *Remote control of Automation Systems for training*, ADEHMS'07, Séoul Septembre 2007 c

Marangé P., Gellot F., Carré-Ménétrier V., Riera B., *Validation de commande des Systèmes à événements Discrets*, MSR'07, Lyon, octobre 2007 d

Marangé P., Gellot F., Riera B., *Control Validation of manufacturing systems: Application to remote laboratories*, WoRD'07, Lyon, October 2007 e

Marangé P., Gellot F., Riera B., *Synthesis of safety system for discrete event control*, 4th international conference on informatics in control, automation and robotics Icinco'07, IFAC ICINCO'2007 Angers, mai 2007 f

Marangé P., Gellot F., Chemla J.P., Riera B., *Requirements and Use for remote teaching of Discrete Events Systems*, Proceeding of 7th IFAC symposium on Advances in control Education, ACE'06, Paper WeP02.1sur le CD-ROM, Madrid, Spain, June 21-23, 2006 a

Marangé P., Tajer A., Gellot F., Carré Ménétrier V., *Synthesis of supervised controller based on Boolean constraints and Boolean automata*, 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2006), pp299-304. Saint-Etienne, France, mai 2006 b

Marca D.A., McGowan C.L., *SADT -- Structured Analysis and Design Technique*, New York : McGraw-Hill, 1988

Mendez H., Zamaï,E., Descotes-Genon B., *Quality, productivity, security and ecological constraints for synthesis of monitoring laws*, 5éme Congrès International Pluridisciplinaire Qualité et Sûreté de Fonctionnement (QUALITA2003), Nancy, 2003

Merle G., Roussel J.M., *Algebraic modelling of fault trees with priority and Gates*, 1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07), ENS Cachan, France, June 2007

Metzger M., *Agent-based virtual control systems for DCS education via Internet*, S3b_4, IFAC Workshop IBCE'04, Grenoble, France, September 2004

Muller P.A, Gaertner N., *Modélisation objet avec UML*, Eyrolles, Paris, ISBN : 2-21211-397-8, 2004

N

Ndjab C.H., *Synthèse de la commande des SED par Grafcet*, Thèse de doctorat de l'Université de Reims, 1999

Niel E., Pietrac L., Regimbal L., *Advantages and drawbacks of the logic program synthesis using supervisory control theory*, 10th IFAC Symposium on information Control Problems in Manufacturing (INCOM 2001), Vienna, Austria, september 2001.

Nourelfath M., Niel E., *Contribution à la surveillance des systèmes automatisés de production*, Journal Européen des Systèmes Automatisés (JESA) Vol. 14, n°2-3, pp105-124, Avril 2000.

P

Parasuraman R, Sheridan T.B, Wickens C.D, *Model for types and levels of human interaction with automation*, IEEE Transaction on Systems, Man, and Cybernetics, Part A: Systems and Humans, Volume 30, Issues 3, pp286-297, 2000

Perrin J., Binet F., Dumery J.J., Merlaud C., Trichard J.P., *Automatique et informatique industrielle - Bases théoriques, méthodologiques et Techniques (livre de l'élève)*, Éditeur : Nathan, ISBN 2-09-179452-X, 2006

Petin J.F., *Méthodes et Modèles pour un processus sur d'automatisation*, Habilitation à diriger les recherches, Université de Nancy, décembre 2007

Peyrucat J.F., *Les outils évolués de conception d'automatismes existent, mais trop peu s'y intéressent*, …, Mesures 778, pp24-26, 2005

Peyrucat J.F., *Il reste trop d'a peu près dans la réalisation des applications d'automatismes !*, Mesures 760, pp34-37, 2003

Philippot A., *Contribution au diagnostic décentralisé des systèmes à événements discrets : Application aux systèmes manufacturiers*, thèse de doctorat de l'Université de Reims Champagne-Ardenne, juillet 2006

Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., *Méthodologie de modélisation dans le cadre de la synthèse formelle des SED*, Douz, Tunisie, papier n°234, 6 pages, 2004 a

Philippot A., Tajer A., Gellot F., Carré-Ménétrier V., *On-line synthesis approach based on a structured plant modelling*, In 7th International Workshop On Discrete Event Systems WODES'04, Reims, France, pp397-402, September 2004 b

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, pp61-75, octobre 2003.

Pourcel C., *Systèmes automatisés de production*, Cepandues - Editions, Toulouse, France, Décembre 1986.

Pujo P., Kieffer J.P., *Fondements du pilotage des systèmes de production*, Traité IC2 productique édition, ISBN : 2-7462-0513-0, 2002

R

Ramadge G., Wonham W. M., *The control of discrete event systems*, Proc. IEEE, Special issue on DEDSs, 77, pp.81-98, 1989.

Real Games Lda, *Its PLC professional edition: User Guide 1.1*, 2008

Riera B., Vigario B., Chemla J-P., Correia L ., Gellot F., *10ans de Maquettes virtuelles pour l'enseignement des automatismes : de WINSIM en 1998 à ITS PLC Professional Edition en 2008*, CETSIS'07, Bruxelles, octobre 2008

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*, J3eA, 5, Hors Série, 2006 a

Riera B., Marangé P., Gellot F, *Ingénierie collaborative en enseignement pour l'apprentissage distribué de l'apprentissage, rapport final*, rapport pour la région Champagne-Ardenne, octobre 2006 b

Riera B., Gellot F., Marangé P., Chemla J.P. et 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 en Électronique, Électrotechnique et Automatique (CETSIS-EEA 2005). Nancy, France, October 2005 a

Riera B., Gellot F, *Ingénierie collaborative en enseignement pour l'apprentissage distribué de l'apprentissage, 2ème rapport intermédiaire*, rapport pour la région Champagne-Ardenne, juillet 2005 b

Riera B., Gellot F, *Ingénierie collaborative en enseignement pour l'apprentissage distribué de l'apprentissage, 1er rapport intermédiaire*, rapport pour la région Champagne-Ardenne, février 2005 c

Riera B, Debernard S, *Basic cognitive principles applied to the design of advanced supervisory systems for process control*, Handbook of cognitive task design, Chapter 12, Ed. Erik Hollnagel, pp255-281, Lawrence Erlbaum Associates Inc., 2003

Riera B., *Specification, design and evaluation of an advanced human adapted supervisory system*, Cognition, Technology and Work, Springer-Verlag London Limited, n°3,pp53-65, 2001

Riera B., Martel G., Angué JC, *La simulation : outil pédagogique pour l'enseignement de l'automatique dans les formations professionnalisées*, Sciences et techniques éducatives, vol.6, n°1, pp37-60, Hermès Sciences, décembre 1999

Rohee B., Riera B., Carré-Ménétrier V. et Roussel J.M., *Outil d'aide à l'élaboration de modèles hybrides de simulation pour les systèmes manufacturiers*, JN-JD-MACS 2007. Reims, France, juillet 2007.

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 DESDes 2006, Rydzyna Castle (Poland), September 2006

Roussel J.M., A Medina., J.M. Faure., *Synthèse d'un programme de commande d'un système logique à partir de l'expression algébrique de ses spécifications, Colloque Francophone sur la modélisation des systèmes réactifs*, MSR'03, Metz, pp77-93, 2003.

Roussel J.M., Denis B., *Safety properties verification of ladder diagram programs*, Journal Européen des Systèmes Automatisés, Hermès Editions, 36(7), pp905–917, ISSN 1269-6935, 2002 a

Roussel J.M., Faure J.M., *An algebraic approach for PLC programs verification*, In Proceedings of 6th international workshop on discrete event systems (WODES'02), Zaragoza, Spain, pp303–308, 2–4 October 2002 b

Roussel J.M., *Analyse de Grafcet par génération logique de l'automate équivalent*, Thèse de doctorat de l'École Normale Supérieure de Cachan, France, 1994.

Ross D, *Structured Analysis (SA): a language for communicating ideas*, IEEE Trans Soft Engineering, SE-3 (1): pp16-34, 1977

Royce W.W., *Managing the development of large software systems*, IEEE WESCON, 1970

Rushby J., *Theorem proving for verification*, In Franck Cassez, editor, Modelling and Verification of Parallel Processes : MoVEP 2k, Nantes, France, June 2000

Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-Oriented Modeling and Design*, Relié - Prentice Hall, ISBN 0136298419, mars 1993

S

Saliah H., *Design of a Generic Interactive Virtual and Remote Electrical Engineering Laboratory*, 29ème conference ASEE/IEEE Frontiers in Education, San Juan, Puerto Rico, 1999

Sarri P, *Stabilisation optimale des systèmes à événements discrets à structure vectorielle: application à la sécurité opérationnelle des systèmes de production*, Thèse de doctorat de l'institut national des sciences de Lyon, janvier1999

Schmidt K., *Cooperative work and its articulation: requirements for computer support*, Le travail humain, tome 57 n°4, pp345-366, 1994

Schmidt K., *Cooperative work: a conceptual framework*, In J. Rasmussen, B. Brehmer, & J. Laplat (Eds), Distributed Decision-Marking: Cognitive Models for Cooperative Work, p.75-110, Chichester, UK: John Willey and Sons, 1991

Schneider Electric, *PRODUCTIS, Notice technique*, II-S-F, MD1AD901, juin 2001

Schnoebelen P., Bérard B., Bidoit M., Laroussinie F. Petit A., *Vérification de logiciels : Techniques et outils du model-checking*, Vuibert, Paris, ISBN 2-7117-8646-3, 1999

Sefiane H., et Senechal O., *Contribution au développement d'une ingénierie du diagnostic pour la télémaintenance*, Congrès International de Génie Industriel(GI), 2005

Simatic, *L'embouteillage et le conditionnement portés sur le CBA*, simatic 09/2008, pp23-25, 2008

T

Tajer A., Marangé P., Gellot F., Carré Ménétrier V., *Synthèse d'une commande supervisée à base de contraintes booléennes*, CIFA'2006, article n°314 sur CD-ROM, Bordeaux, 2006

Tajer A., *Contribution aux approches formelles de synthèse de commande spécifiée par Grafcet*, Thèse de doctorat de l'Université de Reims Champagne Ardenne, Spécialité génie informatique et traitement du signal, novembre 2005

Tardieu H., A. Rochfeld, R. Colletti, *La méthode Merise – Principes et outils*, Édition d'Organisation, ISBN-10: 2-70812-473-0, 2000

Tauvel A., Flety E., *Régulateur de vitesse pour MCC: Choix d'un contrôle à distanc*e, CETSIS'05, Nancy, France, 25-27 octobre 2005

Théron A., *Sciences de l'ingénieur automatique : logique*, collection taupe - niveau Édition Ellipses, ISBN 2-7298-2518-5, 2005

Toguyeni A.K.A., *Surveillance et diagnostic en ligne dans les ateliers flexibles de l'industrie manufacturière*, thèse de doctorat de l'Université de Lille, novembre 1992

V

Vahidi A., Fabian M., Lennartson B., *Efficient supervisory synthesis of large systems*, Control Engineering Practice, Vol. 14(10), pp1157-1167, 2006

Villemeur A., *Sûreté de fonctionnement des systèmes industriels*, collection de la Direction des Études et Recherche d'Électricité de France, Eyrolles, ISBN-10: 2.21201.615.8, 1988.

Volker N. Kramer B.J., *Automated verification of function block-based industrial control systems*, Science of Computer Programming, 42(1):101 113, 2002

W

Wang Y, *Supervisory control Boolean discrete-events systems*, Master's thesis of University of Toronto, June 2000

Wong R.K., *State-based discrete-event modelling and control of concurrent systems*, Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, 1990

Wonham W.M., Ramadge P.J., *On the supermall controllable sublanguage of a given language*, SIAM J. Control Optimization, 25, pp 637-659, 1987

Z

Zamaï E., *Architecture de surveillance – commande pour les systèmes à événements discrets complexes*, thèse de doctorat de l'Université Paul Sabatier, Toulouse, septembre 1997

Zaytoon J., *A contribution to the validation of Grafcet controlled systems, European Journal of control*, vol.7, 2001

Zaytoon J., Carré Ménétrier V., *Grafcet et Graphe d'états : comportement, raffinement, vérification et validation*, Journal Européen des Systèmes Automatisés, vol.33, n°7, pp751-782, 1999

Annexe 1 : Illustration de la synthèse de commande sur un exemple

Pour illustrer ces travaux, nous avons choisi une application dont l'objectif est de trier automatiquement des caisses de deux tailles différentes (figure A1.1). Le processus industriel se compose du convoyeur 1 qui apporte les caisses les unes après les autres, d'un axe composé de deux vérins double effet V1 et V2, de deux vérins simple effet V3 et V4, et de deux convoyeurs (convoyeur 2 et 3) permettant l'évacuation des caisses. Selon la taille de la caisse, l'axe composé de V1 et de V2, place les caisses devant le vérin V3 ou devant le vérin V4.

La notation est la suivante : v_{i0} correspond à la position rentrée du vérin V_i, v_{i1} à la position sortie du vérin V_i, cp_{11} correspond au capteur de petite caisse et cp_{12} au capteur de grande caisse. Les sorties du système sont définies par : *TAPIS* correspond à l'ordre d'activer les trois convoyeurs, *AVANCER1* (*AVANCER2*) à l'ordre de sortie du vérin V1 (resp.V2), *RECULER1* (*RECULER2*) à l'ordre de rentrée du vérin V1 (resp. V2), *AVANCER3* (*AVANCER4*) à l'ordre de sortie du vérin simple effet V3 (resp. V4). Du point de vue de la commande, les vecteurs d'entrées et de sorties correspondant au système de tri de caisses, sont définis de la façon suivante : E = {v_{10}, v_{11}, v_{20}, v_{21}, v_{30}, v_{31}, v_{40}, v_{41}, cp_{11}, cp_{12}}, Z = {*TAPIS*, *AVANCER1*, *AVANCER2*, *RECULER1*, *RECULER2*, *AVANCER3*, *AVANCER4*}.

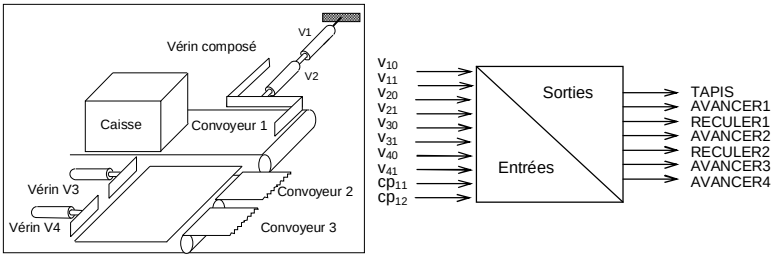


Figure A1.1. Système du tri de caisses

1 Modélisation de la partie opérative

Le modèle de la partie opérative se compose de 4 automates décrivant le mouvement de chacun des 4 vérins avec une prise en compte de la technologie du vérin. Nous ne détaillons dans cette partie que le mouvement associé au vérin V1.

1.1 Établissement des règles d'occurrence et de précédence

Le mouvement du vérin double effet V1 peut être modélisé selon les quatre événements commandables $\downarrow AVANCER1$, $\uparrow AVANCER1$, $\downarrow RECULER1$, et $\uparrow RECULER1$ et les quatre événements non commandables $\uparrow v_{10}$, $\downarrow v_{10}$, $\uparrow v_{11}$ et $\downarrow v_{11}$. L'ensemble des paramètres décrivant le mouvement du vérin V1 est présenté dans le tableau 1.

Conditions initiales	Règles d'occurrence	Relations de précédence
$\overline{AVANCER1}$	$v_{10} \wedge \uparrow AVANCER1 \Rightarrow \downarrow v_{10}$ $\neg v_{10} \wedge \neg v_{11} \wedge \uparrow AVANCER1 \Rightarrow \uparrow v_{11}$	$\downarrow v_{10}$ précède $\uparrow v_{11}$
$\overline{RECULER1}$	$v_{11} \wedge \uparrow RECULER1 \Rightarrow \downarrow v_{11}$ $\neg v_{10} \wedge \neg v_{11} \wedge \uparrow RECULER1 \Rightarrow \uparrow v_{10}$	$\downarrow v_{11}$ précède $\uparrow v_{10}$
$\overline{v_{11}}$	$\neg v_{10} \wedge \downarrow AVANCER1 \Rightarrow \uparrow v_{11}$	Aucune
v_{10}	$\neg v_{11} \wedge \downarrow RECULER1 \Rightarrow \uparrow v_{10}$	Aucune

Tableau 1. Paramètre du mouvement du vérin V1

- – Nous fixons comme état initial que le vérin est en position rentrée et qu'aucun ordre n'est envoyé à la partie opérative
- – Pour définir les règles d'occurrence (colonne 2 du tableau 1), il est nécessaire d'observer les conséquences des ordres envoyés pour un état donné du système. L'ordre *AVANCER1* rend possible le déplacement du vérin et lui permet de quitter

sa position rentrée v_{10}. Par conséquent, la désactivation de v_{10} est une conséquence de l'activation de *AVANCER1*. De même, l'activation de v_{11} est une conséquence de l'activation de *AVANCER1* lorsque le vérin est entre v_{10} et v_{11}.

- – Pour établir les règles de précédence (colonne 3 du tableau 1), il faut définir la chronologie des événements non commandables. Par exemple, dans le cadre de *AVANCER1*, la désactivation de v_{10} précède l'activation de v_{11}
- – La règle d'occurrence liée à la désactivation de l'ordre *AVANCER1* (ligne 3 du tableau 1) traduit le fait qu'à partir du moment où, sous l'influence de l'ordre, le vérin quitte sa position de départ formulé par un front descendant sur v_{10}, la désactivation de l'ordre devient inopérante et le vérin continue sa course jusqu'à ce que v_{11} soit vrai. Le raisonnement est identique pour la ligne suivante du tableau 1 avec l'ordre *RECULER1*.

1.2 Construction de l'automate de partie opérative

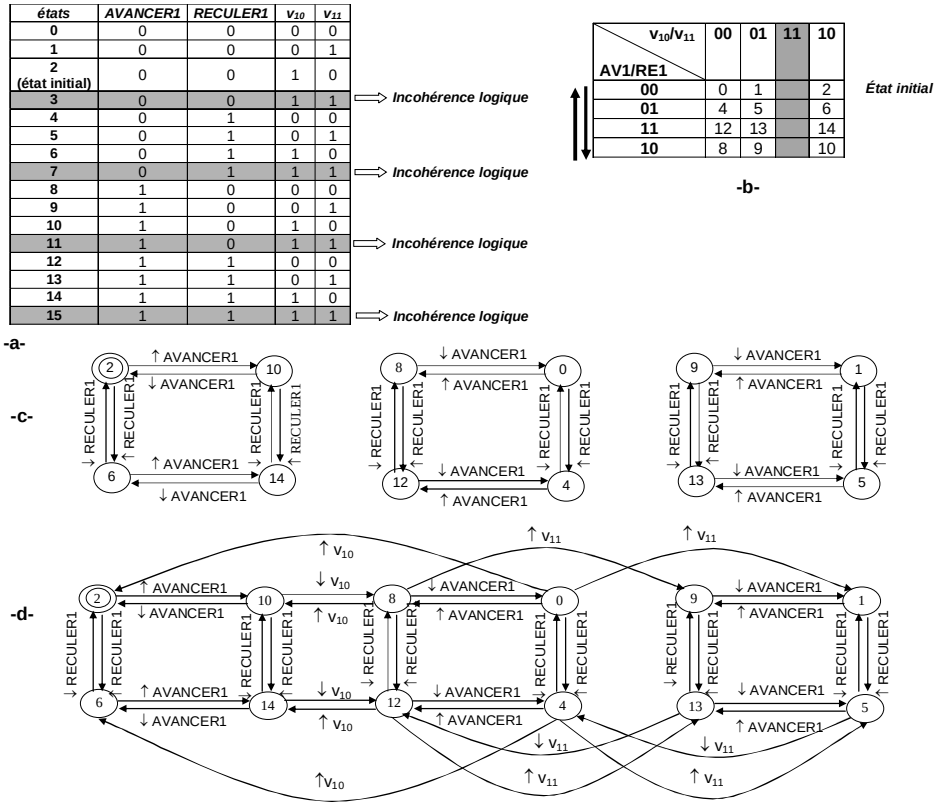
Pour le mouvement du vérin V1, la première étape consiste à établir les 2^4 combinaisons entre *AVANCER1*, *RECULER1*, v_{11} et v_{10} (figure A1.2.a). La deuxième étape consiste à supprimer les 4 incohérences logiques d'événements liées au fait que v_{10} et v_{11} ne peuvent pas être vrais en même temps, nous en déduisons alors le diagramme des évolutions commandables (figure A1.2.b). L'automate des évolutions commandables est ensuite déterminé à partir du diagramme des évolutions exprimant de ce fait toutes les évolutions possibles entre les états. Ces évolutions sont définies uniquement par les événements commandables : $\uparrow$*AVANCER1*, $\downarrow$*AVANCER1*, $\uparrow$*RECULER1*, $\downarrow$*RECULER1*. Sachant que *AVANCER1* et *RECULER1* peuvent soit prendre la valeur 1 pour l'activation, soit 0 pour la désactivation (figure A1.2.c).

L'application des règles d'occurrence et des relations de précédence engendre ensuite l'automate final présenté figure A1.2.d et permet de faire la jonction entre les trois blocs de la figure A1.2.c.

A partir de l'état initial (état 2 figure A1.2.d), les événements commandables possibles sortants sont $\uparrow$*AVANCER1* et $\uparrow$*RECULER1*. L'occurrence de l'événement commandable $\uparrow$*AVANCER1* conduit alors l'automate à l'état 10. La relation d'occurrence correspondant à cet événement est définie par l'événement non commandable $\downarrow v_{10}$ permettant ainsi de faire la jonction avec l'état 8. A partir de cet état, deux cas peuvent se produire : occurrence de $\downarrow v_{10}$

qui entraîne l'apparition de l'événement non commandable $\uparrow v_{11}$ (cf tableau 1) conduisant l'évolution vers l'état 9, ou occurrence de $\downarrow AVANCER1$ conduisant l'évolution vers l'état 0. A partir de l'état 9, l'occurrence de $\downarrow AVANCER1$ conduit vers l'état 1 et à partir de l'état 0, l'occurrence de $\uparrow v_{11}$ conduit vers l'état 1 également. Donc, quel que soit l'ordre d'arrivée des événements commandables ou non, l'état d'arrivée est le même.

Les évolutions non commandables vont venir s'ajouter de la même manière entre les trois automates des évolutions commandables (figure A1.2.b). Il est à noter que ces évolutions ne sont possibles que lorsqu'une seule valeur des entrées v_{11} ou v_{10} change à la fois.



états	AVANCER1	RECULER1	v_{10}	v_{11}	
0	0	0	0	0	
1	0	0	0	1	
2 (état initial)	0	0	1	0	
3	0	0	1	1	⇨ Incohérence logique
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	⇨ Incohérence logique
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	⇨ Incohérence logique
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	⇨ Incohérence logique

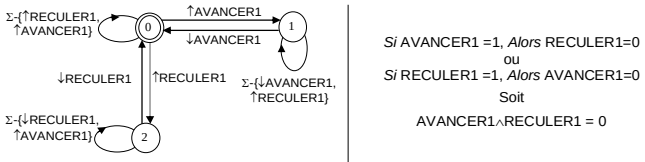
v_{10}/v_{11} / AV1/RE1	00	01	11	10
00	0	1		2
01	4	5		6
11	12	13		14
10	8	9		10

Figure A1.2. Étapes de la construction de l'automate du mouvement horizontal

Pour obtenir l'automate complet de la PO, nous appliquons la même méthode pour chacun des éléments du système et nous effectuons le produit asynchrone entre les modèles élémentaires. Pour cet exemple du tri de caisses, nous obtenons au final un modèle à 15552 états et à 163295 transitions.

2 Modélisation des contraintes

Dans cet exemple, nous nous intéressons à des contraintes au niveau des vérins eux-mêmes et à leur interaction avec les autres vérins ou avec les caisses manipulées par le système. Le vérin V1 ne peut pas rentrer et sortir en même temps, c'est-à-dire que l'envoi simultané des deux ordres *AVANCER1* et *RECULER1* doit être interdit. Cette spécification s'exprime par un modèle automates à trois états (figure 4.343.a) qui est à comparer à l'équation logique de la figure 4.34.b obtenue à travers l'implication suivante : « Si *AVANCER1=1* Alors *RECULER1 = 0* » ou « Si *RECULER1=1* Alors *AVANCER1=0* » qui n'est autre que le complément du ET logique entre *AVANCER1* et *RECULER1*.



a) Automate à états b) Équation logique

Figure A1.3. Contrainte « Ne pas AVANCER1 et RECULER1 en même temps »

La liste ci-dessous présente les cinq contraintes à respecter par le système : la première contrainte comme la deuxième s'applique au vérin V1 et au vérin V2 pour lesquels il est interdit d'effectuer des mouvements en sortie et en entrée simultanément. La troisième et quatrième contrainte concernent exclusivement le vérin V3 pour lequel il est interdit de sortir si (i) le vérin V1 est sorti et/ou si (ii) le vérin V2 est sorti. La cinquième contrainte interdit la sortie du vérin V2 lorsqu'une petite caisse se présente. La sixième et la septième contrainte interdissent de sortir V3 en même temps que V1 ou en même temps que V4. Enfin, la huitième contrainte interdit de sortir le vérin V4 s'il n'y a pas de grandes caisses. L'implantation des capteurs cp11 et cp12 conditionne fortement la véracité de cette contrainte. Par la suite (Chapitre 2), nous verrons cette problématique apparaître.

- Interdiction d'avancer et de reculer le vérin V_1 (1) : $RECULER1 \wedge AVANCER1 = 0$
- Interdiction d'avancer et de rentrer le vérin V_2 (2) : $RECULER2 \wedge AVANCER2 = 0$
- Interdiction d'avancer V_3 si le vérin V_1 est en position v_{11} (3) : $AVANCER3 \wedge v_{11} = 0$
- Interdiction d'avancer V_3 si le vérin V_2 est en position v_{21} (4) : $AVANCER3 \wedge v_{21} = 0$

- Interdiction d'avancer le vérin V_2, si une petite caisse est présente (5) : $AVANCER2 \wedge cp11 = 0$
- Interdiction d'avancer les vérins V_3 et V_1 en même temps (6) : $AVANCER3 \wedge AVANCER1 = 0$
- Interdiction d'avancer les vérins V_3 et V_4 en même temps (7) : $AVANCER3 \wedge AVANCER4 = 0$
- Interdiction d'avancer le vérin V_4 si pas de grandes caisses (8) : $AVANCER4 \wedge \neg cp12 = 0$

2.1 Application à l'exemple

Pour obtenir le comportement maximal admissible du procédé par rapport aux contraintes, il faut utiliser l'algorithme décrit précédemment. Les conditions initiales sont définies par les vecteurs d'entrées et de sorties : E_0=[1,0,1,0,1,0,1,0,0,0] et Z_0=[1,0,0,0,0,0,0].

Depuis l'état p_1 de l'automate de la partie opérative (figure A1..a), une des transitions de sorties est définie par l'événement $\uparrow RECULER1$. Cette évolution est défendue parce que l'occurrence de cet événement ne vérifie pas la contrainte (1). L'évolution (q_1, $\uparrow RECULER1$, q_9) sera ainsi enlevée de l'automate booléen du superviseur.

L'occurrence de l'événement non commandable $\uparrow v_{11}$ de l'état p_{51} peut conduire à l'état p_{60}. Cependant, dans cet état la contrainte (3) n'est pas respectée. En conséquence, l'état p_{51} est un état défendu dans l'automate du superviseur.

De l'état p_{13} (E_{13} = [1,0,1,0,1,0,1,0,0,0], Z_{13} = [1,1,0,0,0,1,0]), l'événement non commandable $\downarrow v_{10}$ conduit vers l'état défendu p_{51}, en conséquence, l'état q_{13} sera défini comme un état faiblement défendu.

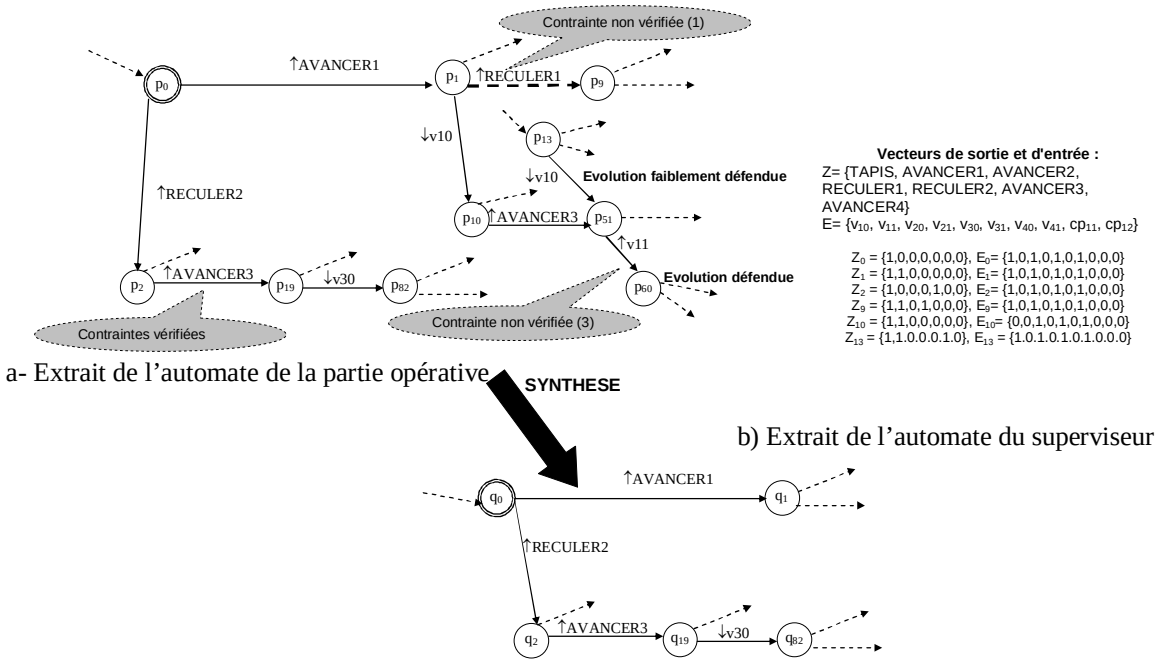


Figure A1.4. Application de la synthèse sur l'exemple du tri de caisses

Après suppression des états défendus, faiblement défendus et non accessibles, nous obtenons un automate booléen représentant le comportement maximal admissible du procédé complet composé de 1920 états et de 26015 transitions ; un extrait de cet automate est donné dans la figure A1.4.b. Si la modélisation des contraintes avait été réalisée par des automates comme ceux préconisés par la théorie de supervision, l'automate du superviseur serait alors composé de 16524 états et 153432 transitions. Le gain en termes d'états et de transitions apparaît ainsi clairement sur cet exemple simple.

2.2 Corrections apportées à la définition des contraintes :

Suite à nos travaux sur la vérification de l'ensemble de contraintes, la définition des contraintes au niveau de cet exemple a été retraitée. Ceci nous a permis de nous rendre compte que cet ensemble est insuffisant et que certaines contraintes étaient mal définies, voir inutiles. En effet, la condition être en v_{11} pour ne pas sortir le vérin n'est pas suffisante, dés que le vérin V1 n'est plus en v_{10}, le vérin V3 ne doit pas sortir, de même avec le vérin V2. La contrainte 7 n'apporte rien au niveau de la sécurité du système ou au fonctionnement même de celui-ci.

L'ensemble de contraintes à définir est le suivant :

- Si l'on considère les éléments de PO pris indépendamment, il y a 5 contraintes à définir pour le vérin V1 et le vérin V2 du fait qu'ils sont bistables, au niveau des vérins V3 et V4, il n'y a aucune contrainte à définir du fait qu'ils sont monostables :
  - $AVANCER1 \wedge RECULER1 = 0$ (1) : Interdire l'envoi des deux commandes en même temps
  - $\uparrow AVANCER1 \wedge v_{10} = 0$ (2) : Interdire la commande *AVANCER1*, si le vérin V1 est déjà en v_{10}
  - $\uparrow RECULER1 \wedge v_{11} = 0$ (3) : Interdire la commande *RECULER1*, si le vérin V1 est déjà en v_{11}
  - $\uparrow v_{11} \wedge RECULER1 = 0$ (4) : Désactiver la commande *RECULER1*, si le vérin V1 est arrivé en v_{11}
  - $\uparrow v_{10} \wedge AVANCER1 = 0$ (5) : Désactiver la commande *AVANCER1*, si le vérin V1 est arrivé en v_{10}
  - $AVANCER2 \wedge RECULER2 = 0$ (6) : Interdire l'envoi des deux commandes en même temps
  - $\uparrow AVANCER2 \wedge v_{20} = 0$ (7) : Interdire la commande *AVANCER2*, si le vérin V2 est déjà en v_{20}
  - $\uparrow RECULER2 \wedge v_{21} = 0$ (8) : Interdire la commande *RECULER2*, si le vérin V2est déjà en v_{21}
  - $\uparrow v_{21} \wedge RECULER2 = 0$ (9) : Désactiver la commande *RECULER2*, si le vérin V2 est arrivé en v_{21}
  - $\uparrow v_{20} \wedge AVANCER2 = 0$ (10) : Désactiver la commande *AVANCER2*, si le vérin V2 est arrivé en v_{20}
- Si l'on considère les interactions entre les éléments de PO, il faut considérer les interactions entre les vérins V1 et V3, les vérins V2 et V3 et entre les vérins V1, V2, et V4 :
  - $AVANCER1 \wedge AVANCER3 = 0$ (11) : Interdire l'envoi des deux commandes d'avancer en même temps
  - $AVANCER3 \wedge \neg v_{10} = 0$ (12) : Interdire la commande *AVANCER3*, si le vérin V1 n'est pas en v_{10}

- $\downarrow AVANCER1 \wedge \neg v_{11} = 0$ (13) : Interdire la désactivation de la commande *AVANCER1*, si le vérin V1 n'est pas en v_{11} permettant ainsi de faire un lien direct entre le sens de déplacement et la commande
- $AVANCER2 \wedge AVANCER3 = 0$ (14) : Interdire l'envoi des deux commandes d'avancer en même temps
- $AVANCER3 \wedge \neg v_{20} = 0$ (15) : Interdire la commande *AVANCER3*, si le vérin V2 n'est pas en v_{20}
- $\downarrow AVANCER2 \wedge \neg v_{21} = 0$ (16) : Interdire la désactivation de la commande *AVANCER2*, si le vérin V2 n'est pas en v_{21}
- $AVANCER1 \wedge AVANCER2 \wedge AVANCER4 = 0$ (17) : Interdire l'envoi des trois commandes d'avancer en même temps
- $AVANCER4 \wedge AVANCER1 \wedge v_{21} = 0$ (18) : Interdire les commandes *AVANCER4 et AVANCER1*, si le vérin V2 est en v_{21}
- $AVANCER4 \wedge AVANCER2 \wedge v_{11} = 0$ (19) : Interdire les commandes *AVANCER4 et AVANCER2*, si le vérin V1 est en v_{11}
- $AVANCER4 \wedge \neg v_{20} \wedge v_{11} = 0$ (20) : Interdire la commande *AVANCER4*, si le vérin V2 n'est pas en v_{20} et le vérin V1 est en v_{11}
- $AVANCER4 \wedge \neg v_{10} \wedge v_{21} = 0$ (21) : Interdire la commande *AVANCER4*, si le vérin V1 n'est pas en v_{10} et le vérin V2 est en v_{21}
- $AVANCER4 \wedge \neg v_{10} \wedge \neg v_{11} \wedge \neg v_{20} \wedge \neg v_{21} = 0$ (22) : Interdire la commande *AVANCER4*, si le vérin V1 n'est pas en v_{10} et le vérin V2 n'est pas en v_{20}

- Si l'on considère les interactions avec le produit, il faut considérer les interactions entre le vérin V3 et la petite caisse, entre le vérin V4 et la grande caisse et entre le vérin V2 et la grande caisse :
  - $AVANCER3 \wedge \neg cp_{11} = 0$ (23) : Interdire la commande *AVANCER3*, s'il n'y a pas de petites caisses
  - $AVANCER4 \wedge \neg cp_{12} = 0$ (24) : Interdire la commande *AVANCER4*, s'il n'y a pas de grandes caisses
  - $AVANCER2 \wedge \neg cp_{12} = 0$ (25) : Interdire la commande *AVANCER2*, s'il n'y a pas de grandes caisses

Annexe 2 :
Bibliothèque de modèles pour la vérification de contraintes

Nous avons pu le voir dans le chapitre 4, la vérification des contraintes se fait sur un ensemble d'élément de PO sous le logiciel UPPAAL. Nous proposons dans cette annexe de rappeler brièvement le fonctionnement des automates à états et de regrouper un ensemble de modèles d'environnement, commande, sens et positions pour des éléments de PO couramment rencontrés dans les systèmes manufacturiers et pouvant être utiliser pour effectuer de nouvelles vérifications.

1 Rappel sur les différents automates

L'automate à états finis présenté dans la figure A2.1.a, définit l'évolution suivante :

- A l'instant initial, l'automate se trouve dans l'état 1
- Suite à l'occurrence de l'événement a, l'automate passe instantanément dans l'état 2. Puis suite à l'occurrence de l'événement b, l'automate revient instantanément dans l'état 2.

Le chronogramme des évolutions de l'automate est illustré dans la figure A2.1.b

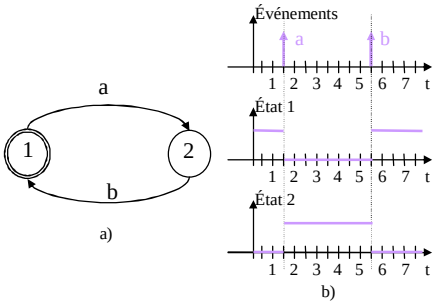


Figure A2.1. Automate à états finis

Dans le cas de l'automate temporisé, l'évolution de l'automate n'est pas toujours instantanée. En effet, dans l'automate de la figure A2.2.a :

- A l'instant initial, l'automate est dans l'état 1,
- Sous l'occurrence de a, l'automate passe instantanément dans l'état 2 et l'horloge h est mise à 0.
- Par contre, lors de l'occurrence de b, deux cas sont possibles.
- Soit b occure après 5 unités de temps, dans ce cas, l'automate évolue instantanément (figure A2.2.b)
- Soit b occure avant 5 unités de temps, dans ce cas, l'automate ne pourra évoluer que lorsque 5 unités de temps se seront écoulées (figure A2.2.c).

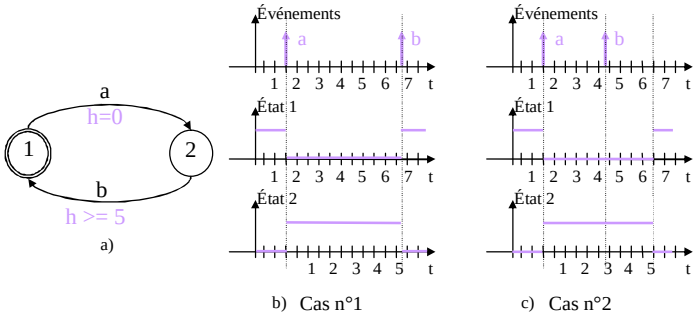


Figure A2.2. Automate temporisé

Les automates communicants (figure A2.3.a) permettent de synchroniser plusieurs automates entre eux. L'évolution donnée dans la figure A2.3.b décrit les évolutions suivantes :

- A l'instant initial, les automates sont dans les états 1 et 3,
- Sous l'occurrence de a, le premier automate passe instantanément dans l'état 2, l'horloge h est remise à 0 et le second automate reste en 3.

- Lors de l'occurrence de b, deux cas sont possibles (exemple précédent). Par contre, lors de la transition 1→ 2, un message est émis (noté mess1 ! : figure A2.3.a) qui permet de faire évolué simultanément le second automate vers l'état 4 par la réception du message (noté mess1 ?)
- Le second automate revient en position 3, suite à l'occurrence de c.

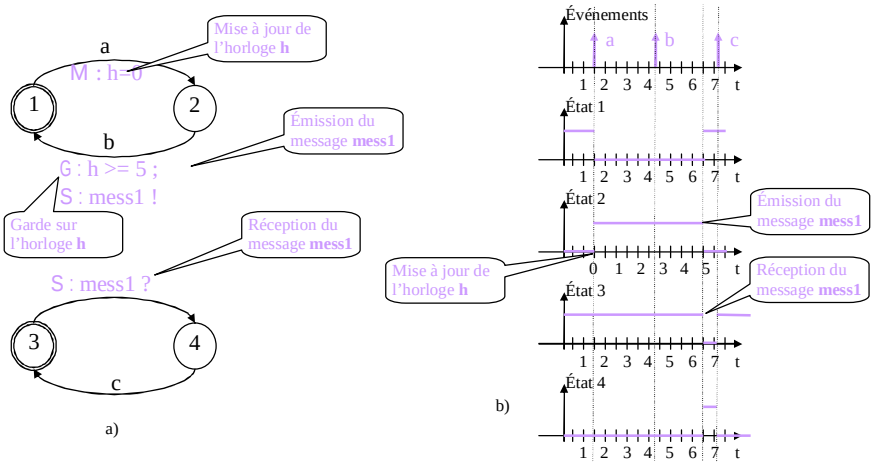


Figure A2.3. Automates communicants

2 Modèle d'environnement

Le modèle d'environnement est modulable en fonction des besoins du système et des contraintes.

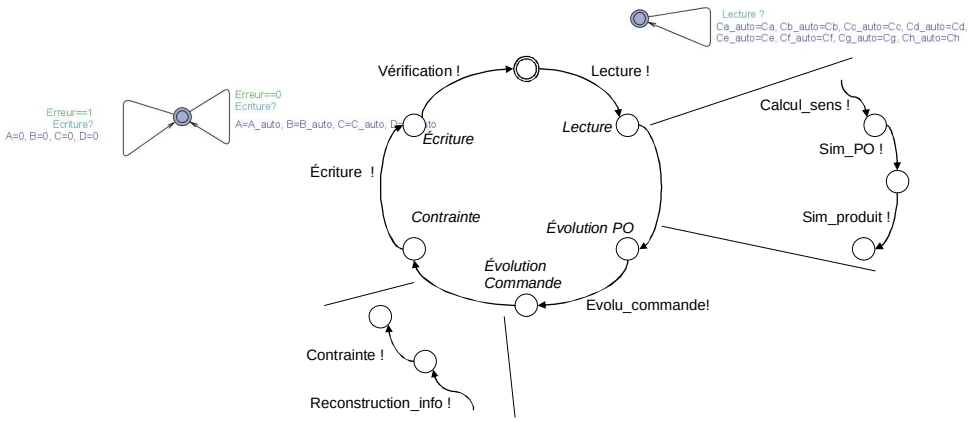


Figure A2.4. Modèle d'environnement

Lors de la validation des contraintes, si les contraintes utilisent seulement les informations capteur, il est inutile d'ajouter la transition pour émettre le message « *reconstruction _info* ». De même, pour la validation de contraintes lors d'une interaction entre élément de PO, il n'est pas nécessaire d'émettre le message « *Sim_produit* ». Le modèle d'environnement est composé aussi d'un modèle de lecture qui recopie les valeurs du modèle de position sur les variables API et d'écriture qui met à jour les variables de commande par rapport aux évolutions de l'API.

3 Modèle de commande

Deux modèles de commande sont définis pour une commande monostable ou bistable.

3.1 Commande monostable

Lorsque la commande est monostable, elle ne peut être activée ou désactivée entre l'instant de réception du message « *évolu_commande* » et « *calcul_front* ». L'activation est représentée par la variable *AB_auto* et la désactivation par *DB_auto*. À chaque évolution, la variable *B_auto* associée est mise à jour. Pour empêcher l'évolution de la commande en cas d'erreur, une condition/garde *erreur* est mise à 1, interdisant ainsi toutes évolutions.

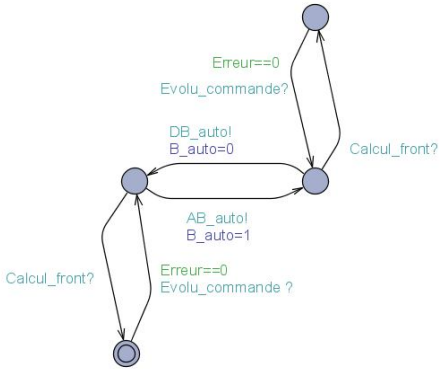


Figure A2.5. Modèle de commande monostable

3.2 Commande bistable

Lorsque la commande est bistable, pendant le temps d'évolution de la commande, une seule commande peut être activée (*UAA_auto*, *UAB_auto*) ou désactivée (*UDA_auto*, *UDB_auto*) ou les deux commandes sont inversées en même temps : Activation de A et désactivation de B (*UAC_auto*) ou Activation de B et désactivation de A (*UDC_auto*). À chaque évolution les variables de commande (*UA_auto*, *UB_auto*) sont mises à jour.

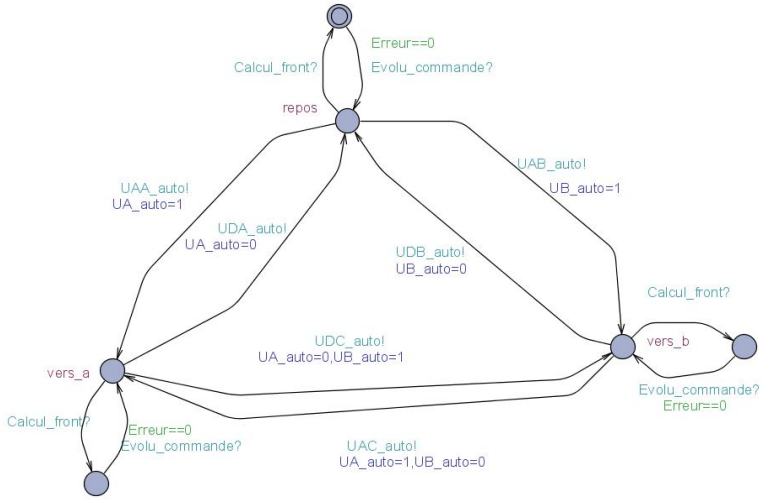


Figure A2.6. Modèle de commande bistable

4 Modèle de sens

Les modèles de sens permettent de faire à paraître la technologie et les effets physiques (retards commande - déplacement, inertie). Cette partie reprend les éléments les plus couramment rencontrés : différents distributeurs, moteurs.

4.1 Distributeurs 5/2

Pour représenter un distributeur 5/2, il faut tenir compte qu'après émission d'une commande, le distributeur reste actif jusqu'à l'envoi de la commande inverse. Cela est représenter par le maintient de la force à sa valeur précédente. Pour un distributeur 5/2, il y a seulement 3 valeurs possibles pour la force -1, 0 ou 1 et le passage d'une valeur à l'autre se fait en 1seul pas $f = 1$. Le sens de déplacement s'obtient simplement en comparent la force F

aux valeurs seuils : *frp* = 1 et f*rm*=-1. La figure A2.7. donne le modèle de sens d'un distributeur 5/2.

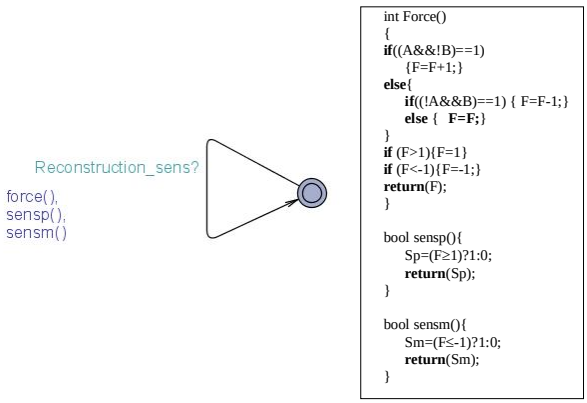


Figure A2.7. Modèle de sens d'un distributeur 5/2

Il est à noter que pour un élément monostable, c'est le même modèle sauf qu'il faut retirer la commande B.

4.2 Distributeurs 5/3

Pour représenter un distributeur 5/3, lorsqu'aucune commande n'est envoyée sur le distributeur, celui-ci revient en position de repos. Cela se modélise en faisant revenir la valeur de la force F vers 0, sinon le fonctionnement est le même que précédemment.

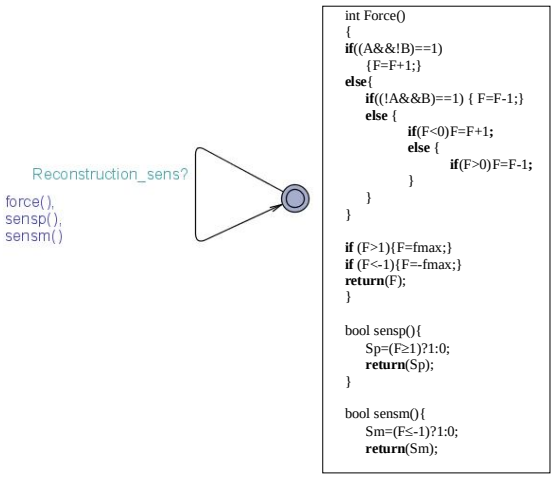


Figure A2.8. Modèle de sens d'un distributeur 5/3

4.3 Moteurs à 1 sens de rotation

Le modèle de sens d'un moteur doit modéliser le laps de temps pour se mettre en marche et pour s'arrêter après évolution de la commande. Pour cela, lorsque la commande est active, la force *F* est incrémentée de la valeur *f*, et lorsque la commande est désactivée, la force *F* est décrémentée de *f*. C'est sur ce paramètre *f*, la valeur de seuil *frp* et la valeur maximum *fmax* qu'il est possible de jouer pour moduler le laps de temps de mise en, marche ou d'arrêt.

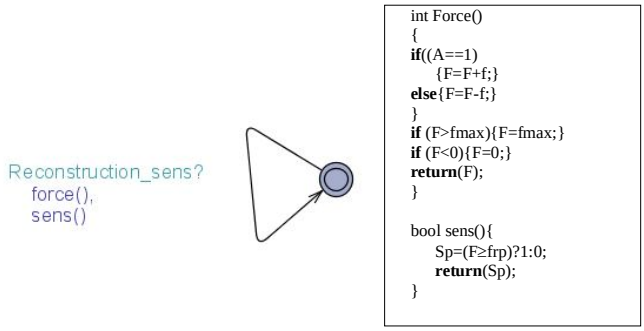


Figure A2.9. Modèle de sens d'un moteur à 1 sens de rotation

Si le moteur modélisé à une grande inertie, la valeur de *fmax* sera élevé. Si le moteur doit avoir une force importante pour se mettre en marche, la valeur de *frp* sera éloignée de 0. Pour modéliser un moteur sans inertie, il faut mettre les valeurs suivante : *f*=2, *frp*=1, *frm*=-1 et *fmax*=2 et, 3 temps de cycle dans une même position, par exemple.

4.4 Moteur à 2 sens de rotation, chariots

Pour le moteur à 2 sens de rotation, est identique au distributeur 5/3 à la différence que les paramètres ne sont pas fixes à 1, -1 et 0. En effet c'est en modifiant la valeur de seuil (*frp, frm)*, de *fmax* et de *f*, qu'il est possible de faire apparaître plus ou moins d'inertie, de retard au démarrage.

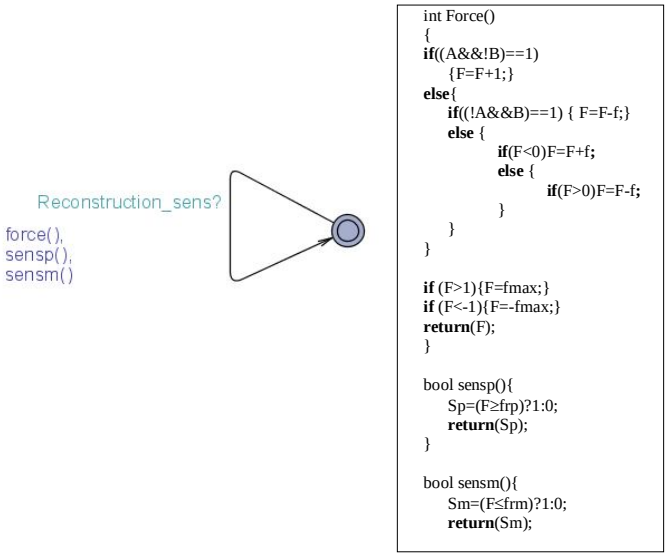


Figure A2.10. Modèle de sens d'un moteur à 2 sens de rotation

5 Modèle de positions

5.1 2 capteurs

Un élément à 2 capteurs peut prendre 3 positions différentes : sur chacun des capteurs, ou entre les deux. Le laps de temps, pour quitter le capteur et pour arriver sur le capteur suivant, est modélisée par la variable iter2. (Cette valeur est différente dans les 2 cas en réalité mais ça ne modifie en rien la validation des contraintes)

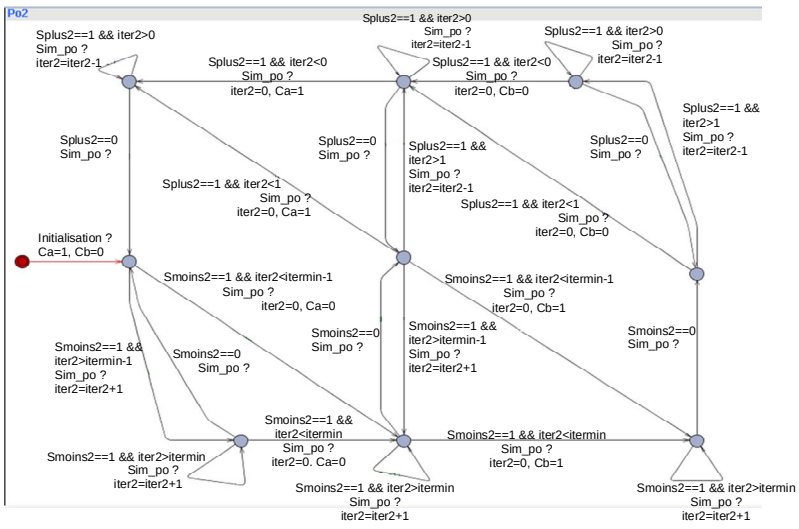


Figure A2.11. Modèle de positions pour 2 capteurs

5.2 3 capteurs

Pour un élément à 3 capteurs, idem que le modèle 2 capteurs sauf qu'il y a 5 positions.

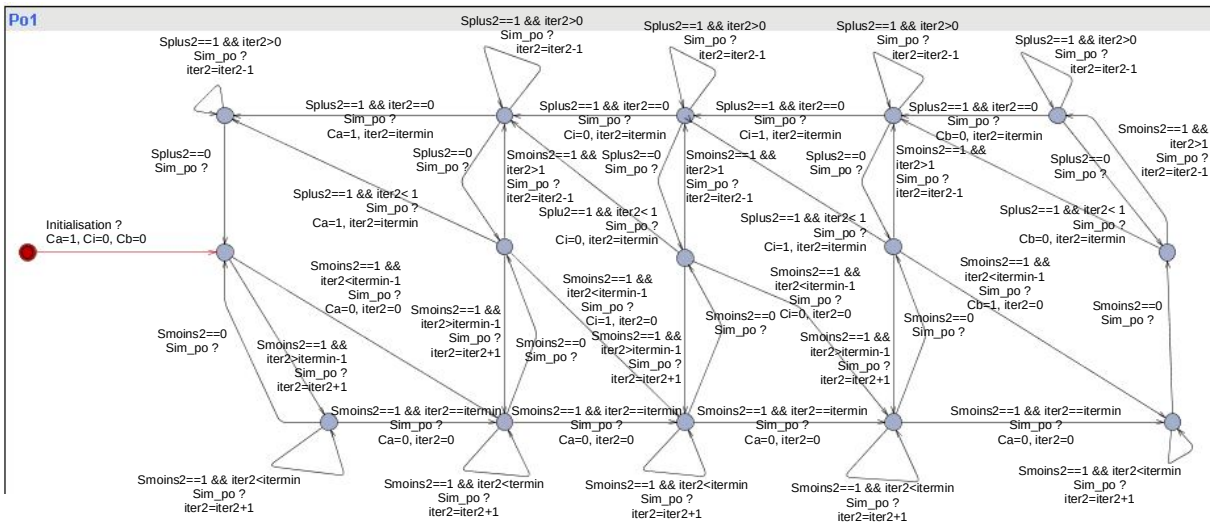


Figure A2.12. Modèle de positions pour 3 capteurs

5.3 4 capteurs

Pour un élément à 4 capteurs, le modèle est identique à celui de 2 ou 3 capteurs sauf qu'il y a 7 positions.

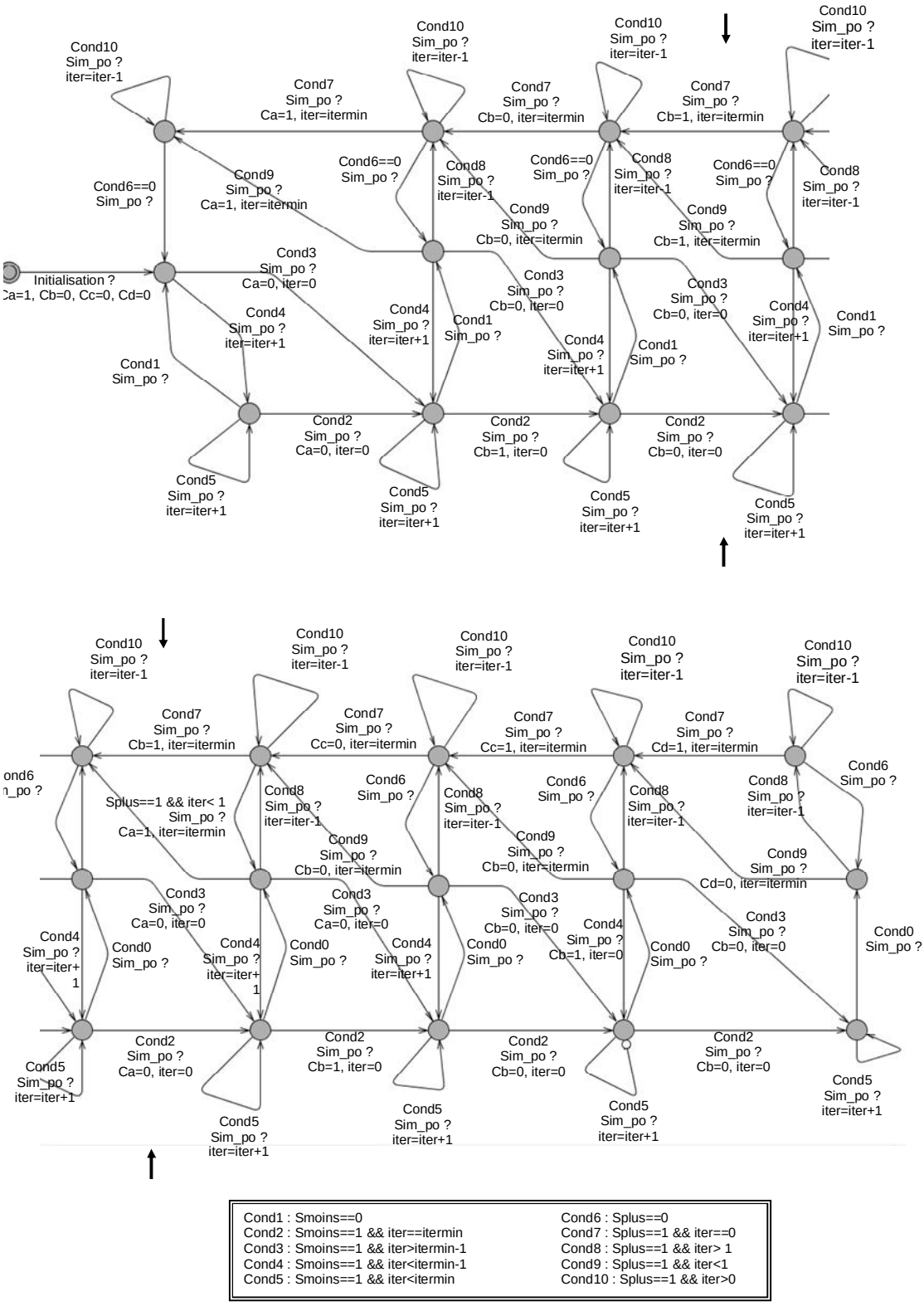


Figure A2.13. Modèle de positions pour 4 capteurs

Pour la validation de certaines contraintes, il est nécessaire de simuler une zone critique pour cela, il faut ajouter au modèle de positions avec une zone critique délimité par des capteurs virtuels.

5.4 Tapis

Le modèle de position du tapis permet de générer la position de la pièce sur le tapis ainsi que les capteurs. La figure A2.14 représente un tapis avec un capteur de fin de course *k* pour détecter la sortie de la pièce du tapis.

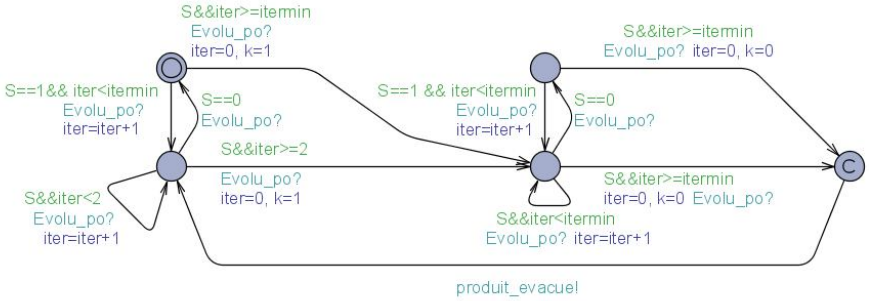


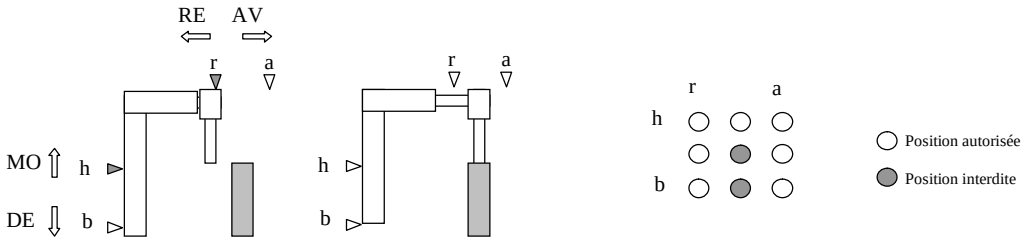
Figure A2.14. Modèle de positions pour un produit sur un tapis

6 Autres exemples d'obtention de contraintes

D'autres exemples ont été traités mais pas présentés dans le chapitre 4. Un exemple du mouvement en U d'un préhenseur et un exemple de partage de zone commune entre deux chariots reprenne

6.1 Mouvement en U d'un préhenseur

Sur cet exemple du préhenseur, le système est constitué de deux vérins : Vérin Horizontal {*a*, *r*, *AV*, *RE*} et d'un Vérin Vertical {*b*, *h*, *MO*, *DE*} pilotés par des distributeurs 5/2 bistables. Pour éviter la collision avec un obstacle le préhenseur ne peut se déplacer horizontalement qu'en position haute.



a) Position autorisée b) Position interdite c) Différentes positions atteignables

Figure A2.15. Préhenseur avec un mouvement en U

Les variables utilisées au niveau du système et au niveau de l'API sont les suivantes :

	Variables système	Variables API
Vérin vertical	*MO*, *DE,* *h*, *b*	*MO_auto*, *DE_auto*, *h_auto*, *b_auto*
Vérin horizontal	*AV*, *RE*, *a*, *r*	*AV_auto*, *RE_auto*, *a_auto*, *r_auto*

Figure A2.16. Variables pour le préhenseur

La vérification des contraintes se fait sur les mêmes modèles que précédemment car les ElPO utilisés sont les mêmes. La prise en compte de la structure du système se fait au niveau de la définition des états interdits, c'est-à-dire dans l'équation logique du modèle de vérification : $\neg a \wedge \neg r \wedge \neg h$ et au niveau de la définition des contraintes.

Pour obliger le système à se déplacer horizontalement seulement en position haute, les quatre contraintes suivantes sont définies :

- Interdire la commande AV si le préhenseur n'est pas en position h :

$$AV\_auto \wedge \neg h\_auto = 0 \qquad (A2.1)$$

- Interdire la commande RE si le préhenseur n'est pas en position h :

$$RE\_auto \wedge \neg h\_auto = 0 \qquad (A2.2)$$

- Interdire l'envoi des deux commandes en même temps :

$$AV\_auto \wedge DE\_auto = 0 \qquad (A2.3)$$

- Interdire l'envoi des deux commandes en même temps :

$$RE\_auto \wedge DE\_auto = 0 \qquad (A2.4)$$

Une cinquième contrainte est ajoutée pour interdire de descendre si le vérin vertical n'est pas en *a* ou en *r* :

$$DE \wedge \neg a \wedge \neg r = 0 \qquad (A2.5)$$

Pour prendre en compte le sens de déplacement du système, il est nécessaire de définir les trois contraintes suivantes :

- Interdire de désactiver AV si le préhenseur n'est pas en a :

$$\downarrow AV\_auto \wedge \neg a\_auto = 0 \quad \text{(A2.6)}$$

- Interdire de désactiver RE si le préhenseur n'est pas en r :

$$\downarrow RE\_auto \wedge \neg r\_auto = 0 \quad \text{(A2.7)}$$

- Interdire de désactiver DE si le préhenseur n'est pas en b :

$$\downarrow DE\_auto \wedge \neg b\_auto = 0 \quad \text{(A2.8)}$$

Ou comme nous l'avons vu dans le chapitre 4, les trois contraintes suivantes :

- Interdire de désactiver AV si le préhenseur n'a pas quitté r :

$$\downarrow AV\_auto \wedge r\_auto = 0 \quad \text{(A2.9)}$$

- Interdire de désactiver RE si le préhenseur n'a pas quitté a :

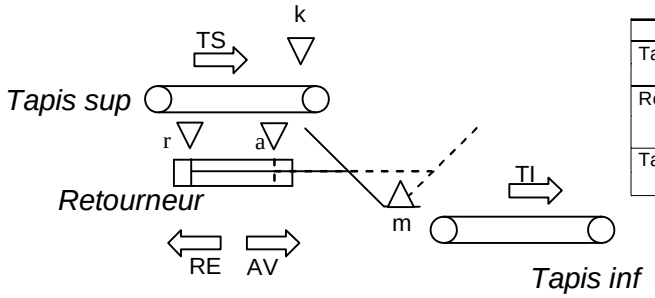
$$\downarrow RE\_auto \wedge a\_auto = 0 \quad \text{(A2.10)}$$

- Interdire de désactiver DE si le préhenseur n'a pas quitté h :

$$\downarrow DE\_auto \wedge h\_auto = 0 \quad \text{(A2.11)}$$

6.2 Transfert de pièce par retourneur

Soit un système (figure A2.17.) qui est constitué d'un tapis supérieur {*k*, *TS*} alimentant un tapis inférieur {*TI*} en pièces pour être évacuées. Le transfert s'effectue par un retourneur {*a*, *r*, *t*, *AV*, *RE*}. Une fois la pièce sur le tapis inférieur, elle est évacuée. Il est supposé que le Tapis inférieur marche en permanence, ce qui signifie que les pièces ne peuvent s'accumuler devant le retourneur. Ici, il n'y a pas d'interactions entre ElPO ce qui ne nécessite donc pas de contraintes.



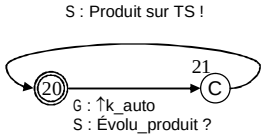
a) Système

	Variables système	Variables API
Tapis sup	TS k	TS_auto k_auto
Retourneur	*AV, RE,*	*AV_auto, RE_auto,*
	a, r,	*a_auto, r_auto,*
Tapis inf	*TI*	*TI_auto*

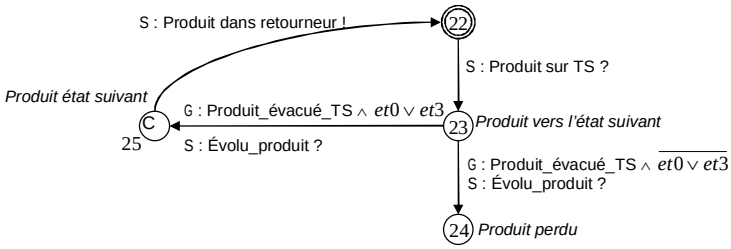
b) Variables

Figure A2.17. Transfert de pièces

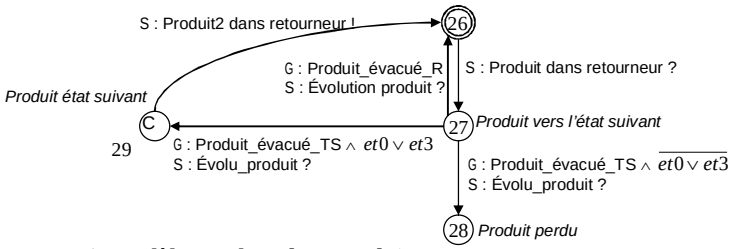
Il y a un passage critique du produit : entre le tapis supérieur et le retourneur Il y a trois états produits à définir : pièce sur TS, pièce dans le retourneur, deux pièces dans le retourneur (figure A2.17.).



a) Modèle produit *produit sur TS*



b) Modèle produit *tapissup_retourneur*



c) Modèle produit *deuxproduits_retourneur*

Figure A2.18. Modèles de produit

Les contraintes mises en place pour éviter de perdre le produit entre le passage du tapis supérieur au retourneur sont :

- Interdire l'ordre TS, si le retourneur n'est pas en position r lorsqu'il y a une pièce sur le tapis supérieur

$$TS\_auto \wedge k\_auto \wedge \neg r\_auto = 0 \qquad \text{(A2.}12\text{)}$$

- Interdire l'activation en même temps de TS et AV lorsqu'il y a une pièce sur le tapis supérieur

$$TS\_auto \wedge AV\_auto \wedge k\_auto = 0 \qquad \text{(A2.}13\text{)}$$

- Interdire la désactivation de AV si le retourneur n'a pas quitté r

$$\downarrow AV\_auto \wedge r\_auto = 0 \qquad \text{(A2.}\textit{14}\text{)}$$

La contrainte à ajouter aux précédentes pour éviter de mettre plusieurs pièces dans le retourneur est :

- Interdire d'activer TS, s'il y a une pièce dans le retourneur lorsqu'il y a une pièce sur le tapis supérieur

$$\uparrow TS\_auto \wedge k\_auto \wedge m\_auto = 0 \qquad \text{(A2.15)}$$

La vérification de la suffisance de ces contraintes se fait sur les ElPO suivant :

- Du retourneur :

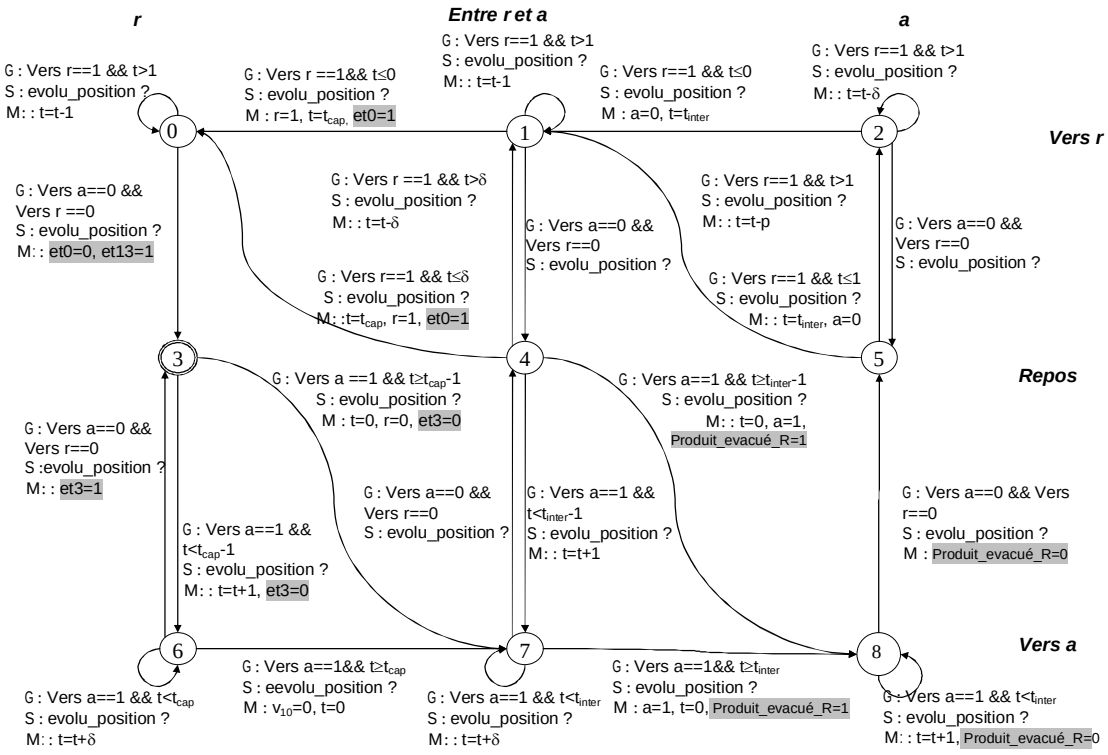


Figure A2.19. Modèle de positions du retourneur

- Du tapis : ce modèle représente la pièce qui arrive sous le capteur puis qui le quitte pour être évacuée du tapis

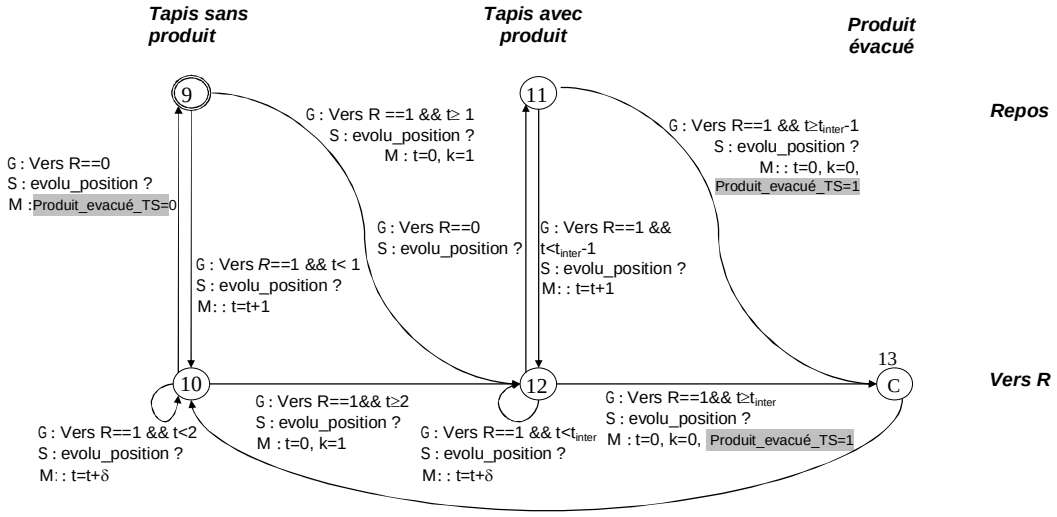


Figure A2.20. Modèle de positions du tapis

- De la génération du capteur dans le retourneur : ce modèle permet de mettre à jour le capteur placé dans le retourneur. La mise à 1est générée lorsque la pièce est évacuée du tapis supérieur et la mise à 0, lorsque la pièce est évacuée du retourneur.

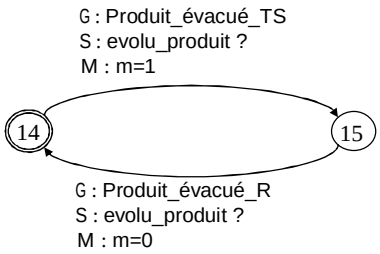


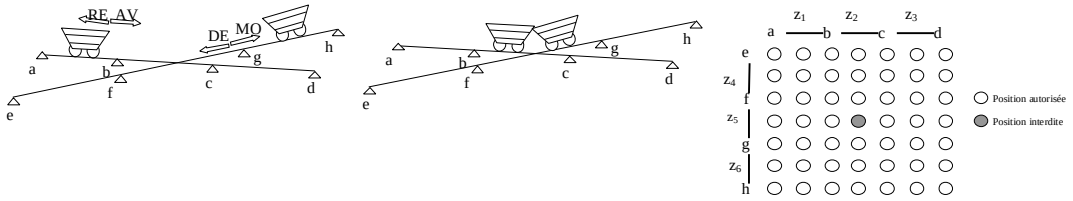
Figure A2.21. Modèles de génération du capteur m

La propriété P2 a été validée sur chaque modèle de produit de façon modulaire. Les contraintes de (1) à (3) ont été validées à partir des modèles de commande (tapis : figure A2.5, retourneur : figure A2.6), sens (tapis : figure A2.9, retourneur : figure A2.7) et positions (tapis : figure A2.15, retourneur : figure A2.11) sur le modèle produit *tapissup_retouneur* (figure A2.18.b). La contrainte (4) associée aux précédentes, a été validée sur les mêmes modèles et le modèle produit *deuxproduits_retourneur* (figure A2.18.c). Sur le modèle produit *deuxproduits_retourneur*, la transition de l'état 27 à l'état 26 a été ajoutée pour représenter le départ de la pièce du retourneur.

Cet exemple a présenté la prise en compte du produit dans le système et l'application de l'approche de vérification des contraintes.

6.3 Deux chariots partageant une zone commune

Le système suivant est composé de deux chariots (*Ch1* et *Ch2*) qui se déplacent de *a* à *d* et de *e* à *h* sur des voies qui partagent une zone commune. La zone commune est délimitée pour le chariot *Ch1* par les capteurs *b* et *c* et pour le chariot *Ch2* par les capteurs *f* et *g*. Les chariots sont pilotés par les commandes pour le chariot *Ch1* {*RE, AV*} et pour le chariot *Ch2* {*MO, DE*}



a) Position autorisée b) Position interdite c) Différentes positions atteignables

Figure A2.22. Chariots avec zone commune

Nous allons considérer deux cas : Soit les chariots n'ont pas d'inertie, c'est-à-dire que le contact avec le capteur est assez long pour permettre l'arrêt du chariot sur le capteur. Soit les chariots ont de l'inertie, dans ce cas, le chariot va quitter obligatoirement le capteur bien que l'ordre soit désactivé correctement. La définition des contraintes dans l'API et la vérification se font sur les variables suivantes :

	Variables système	Variables API
Chariot Ch1	*AV, RE,*	*AV_auto, RE_auto,*
	a, b, c, d *z2*	*a_auto, b_auto,* *c_auto, d_auto* *z2_auto*
Chariot Ch2	*MO, DE,*	*MO_auto, DE_auto,*
	e, f, g, h *z5*	*e_auto, f_auto,* *g_auto, h_auto* *z5_auto*

Figure A2.23. Variables pour le préhenseur avec capteur intermédiaire

Si l'on suppose que les chariots n'ont pas d'inertie, les zones *z1* et *z2* ne doivent jamais être vraies en même temps. Pour pouvoir définir les contraintes, les informations sur les zones intermédiaires doivent être reconstruites. Pour cela, la zone *z1* est définie par l'intervalle]*a, b*[, la zone *z2* par l'intervalle]*b, c*[, la zone *z3* par l'intervalle]*c, d*[, la zone *z4* par l'intervalle]*e, f*[, la zone *z5* par l'intervalle]*f, g*[, la zone *z6* entre par l'intervalle]*g, h*[. La reconstruction des zones se fait à partir de l'information capteur et de la commande en supposant que celle ci est équivalente à la commande :

- z1=1 si ↓a∨(↓b∧RE) et z1=0 si ↑a∨↑b ;
- z2=1 si (↓b∧AV)∨(↓c∧RE) et z2=0 si ↑b∨↑c ;…

Il est maintenant possible de définir simplement les contraintes. Deux configurations du système sont à prendre en compte. Soit le chariot *C1* veut entrer dans la zone, et il y a déjà le chariot *C2* dans la zone commune (et inversement). Soit les deux chariots veulent rentrer en même temps. L'ensemble de contraintes correspondant à ces deux configurations est le suivant :

- Interdire l'envoi des commandes AV si le chariot C2 se trouve dans la zone z5, et que le chariot C1 est en c :

$$AV \wedge z5 \wedge c = 0 \quad \text{(A2.16)}$$

- Interdire l'envoi de la commande RE si le chariot C2 se trouve dans la zone z5, et que le chariot C1 est en b :

$$RE \wedge z5 \wedge b = 0 \quad \text{(A2.17)}$$

- Interdire l'envoi des commande DE si le chariot C1 se trouve dans la zone z2, et que le chariot C2 est en f :

$$DE \wedge z2 \wedge f = 0 \quad \text{(A2.18)}$$

- Interdire l'envoi des commandes MO si le chariot C1 se trouve dans la zone z2, et que le chariot C2 est en g :

$$MO \wedge z2 \wedge g = 0 \quad \text{(A2.19)}$$

- Interdire l'envoi des commandes AV et DE si les chariots se trouve en c et g :

$$AV \wedge DE \wedge c \wedge g = 0 \quad \text{(A2.20)}$$

- Interdire l'envoi des commandes AV et MO si les chariots se trouve en c et f

$$AV \wedge MO \wedge c \wedge f = 0 \quad \text{(A2.21)}$$

- Interdire l'envoi des commandes RE et DE si les chariots se trouve en b et g

$$RE \wedge DE \wedge b \wedge g = 0 \quad \text{(A2.22)}$$

- Interdire l'envoi des commandes RE et MO si les chariots se trouve en b et f

$$RE \wedge MO \wedge b \wedge f = 0 \quad \text{(A2.23)}$$

La vérification des contraintes sous UPPAAL se fait sur le modèle de commande (figure A2.6), le modèle de sens (figure A2.10) et un modèle de positions à 4 capteurs (figure A2.13). L'équation logique z2∧z5 = 0 définie dans le modèle de vérification représente la

position à ne pas atteindre. L'ensemble de contraintes ainsi défini n'est pas suffisant. En effet, deux comportements sur les capteurs délimitant la zone n'ont pas été pris en compte :

- Si le concepteur demande la désactivation juste avant de quitter le capteur, ayant un temps de cycle de décalage entre les valeurs lues et les valeurs réelles, les deux chariots peuvent se retrouver en même temps dans la zone
- Si le concepteur demande l'activation puis la désactivation successivement sur le capteur, les chariots vont se déplacer petit à petit vers la zone avec un cycle de décalage et vont finir par se rentrer dedans.

Pour remédier à cela, quatre contraintes sont ajoutées pour obliger la désactivation de la commande pour rentrer dans la zone seulement lorsque le capteur vient d'être activer pour les capteurs délimitant la zone :

- Interdire de désactiver MO si f ne vient pas d'être activé lorsqu'il est sur f

$$\downarrow MO \wedge f \wedge \neg(\uparrow f) = 0 \quad \text{(A2.24)}$$

- Interdire de désactiver DE si g ne vient pas d'être activé lorsqu'il est sur g

$$\downarrow DE \wedge g \wedge \neg(\uparrow g) = 0 \quad \text{(A2.25)}$$

- Interdire de désactiver AV si b ne vient pas d'être activé lorsqu'il est sur b

$$\downarrow AV \wedge b \wedge \neg \uparrow b = 0 \quad \text{(A2.26)}$$

- Interdire de désactiver RE si c ne vient pas d'être activé lorsqu'il est sur c

$$\downarrow RE \wedge c \wedge \neg \uparrow c = 0 \quad \text{(A2.27)}$$

Les douze contraintes ainsi définies vérifient la propriété sous UPPAAL en testant si la zone z2 et la zone z5, peuvent être atteintes en même temps. Dans le cas de systèmes réels avec des chariots, il est nécessaire de prendre en compte les effets d'inertie. Ces effets n'ont pas été traité au cours de cette thèse et font l'objet de perspectives.

Annexe 3 :
Bibliothèque de contraintes

Le chapitre 4 a permis de valider un ensemble de contraintes sous le logiciel Uppaal. Nous proposons dans cette annexe de regrouper les ensembles de contraintes pouvant être réutilisé pour de nouveaux systèmes en fonction de :

- Élément de PO
  - Vérins bistables, chariots
  - Vérins monostables, tapis
- Interaction entre éléments de PO
  - Zone commune entre 2 vérins
  - Mouvement du préhenseur en U
  - Mouvement en S
  - Zone commune entre 2 chariots
- Interaction produit
  - Tapis retourneur / Retourneur tapis
  - Retourneur tapis-ascenseur

1 Éléments de PO

1.1 Vérins bistables, chariots

- $AV \wedge RE = 0$ (1) : Interdire l'envoi des deux commandes en même temps
- $\uparrow AV \wedge a = 0$ (2) : Interdire la commande *AV*, si le vérin horizontal est déjà en *a*
- $\uparrow RE \wedge r = 0$ (3) : Interdire la commande *RE*, si le vérin horizontal est déjà en *r*
- $\uparrow r \wedge RE = 0$ (4) : Désactiver la commande *RE*, si le vérin horizontal est arrivé en *r*
- $\uparrow a \wedge AV = 0$ (5) : Désactiver la commande *AV*, si le vérin horizontal est arrivé en *a*

1.2 Vérins monostables, tapis

Aucune contrainte

2 Interaction entre éléments de PO

2.1 Zone commune entre 2 vérins

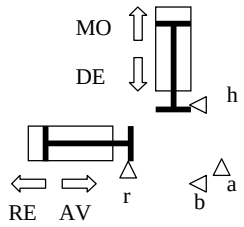


Figure A3.1. Zone commune entre 2 vérins

Dans la configuration de la figure A3.1, il faut définir les contraintes suivantes

- $AV \wedge \neg h = 0$ (1) : Interdire la commande *AV*, si le vérin vertical n'est pas en *h*
- $DE \wedge \neg r = 0$ (2) : Interdire la commande *DE*, si le vérin vertical n'est pas en *r*
- $AV \wedge DE = 0$ (3) : Interdire *AV* et *DE* en même temps

En fonction de la technologie des pré-actionneurs des vérins, si ce sont des distributeurs 5/2, il faut ajouter :

- $\downarrow AV \wedge \neg a = 0$ (4) : Interdire de désactiver la commande *AV*, si le vérin n'est pas en *a*
- $\downarrow DE \wedge \neg b = 0$ (5) : Interdire de désactiver la commande *DE*, si le vérin n'est pas en *b*

2.2 Mouvement du préhenseur en U

Dans la configuration de la figure A3.2, il faut définir les contraintes suivantes

- $AV \wedge \neg h = 0$ (1): Interdire la commande *AV* si le préhenseur n'est pas en position *h*
- $RE \wedge \neg h = 0$ (2): Interdire la commande *RE* si le préhenseur n'est pas en position *h*
- $AV \wedge DE = 0$ (3) : Interdire l'envoi des deux commandes en même temps
- $RE \wedge DE = 0$ (4) : Interdire l'envoi des deux commandes en même temps
- $DE \wedge \neg a \wedge \neg r = 0$ (5). Pour prendre en compte le sens de déplacement du système,

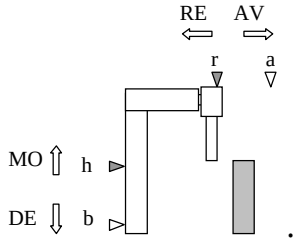


Figure A3.2. Préhenseur avec un mouvement en U

En fonction de la technologie des pré-actionneurs des vérins, si ce sont des distributeurs 5/2, il faut ajouter :

- $\downarrow AV \wedge \neg a = 0$ (6) : Interdire de désactiver *AV* si le préhenseur n'est pas en *a*
- $\downarrow RE \wedge \neg r = 0$ (7) : Interdire de désactiver *RE* si le préhenseur n'est pas en *r*
- $\downarrow DE \wedge \neg b = 0$ (8) : Interdire de désactiver *DE* si le préhenseur n'est pas en *b*

2.3 Mouvement en S

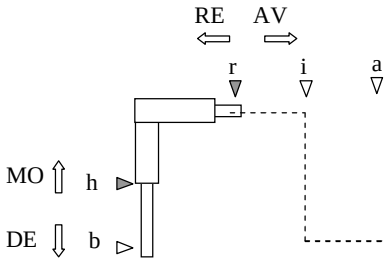


Figure A3.3. Préhenseur avec un capteur intermédiaire

Dans la configuration de la figure A3.3, il faut définir les contraintes suivantes :

- $MO \wedge \neg i = 0$ (1) : Interdire la commande *MO* si le préhenseur n'est pas en position *i*
- $DE \wedge \neg i = 0$ (2) : Interdire la commande *DE* si le préhenseur n'est pas en position *i*
- $AV \wedge z1 \wedge \neg h = 0$ (3) : Interdire la commande *AV* si le préhenseur n'est pas en position *h* dans la zone *z1*
- $RE \wedge z1 \wedge \neg h = 0$ (4) : Interdire la commande *RE* si le préhenseur n'est pas en position *h* dans la zone *z1*
- $AV \wedge z2 \wedge \neg b = 0$ (5) : Interdire la commande *AV* si le préhenseur n'est pas en position *h* dans la zone *z1*
- $RE \wedge z2 \wedge \neg b = 0$ (6) : Interdire la commande *RE* si le préhenseur n'est pas en position *h* dans la zone *z1*
- $RE \wedge i \wedge \neg h = 0$ (7) : Interdire la commande *RE* lorsqu'il est en position *i* s'il n'est pas en position *h*
- $AV \wedge i \wedge \neg b = 0$ (8) : Interdire la commande *AV* lorsqu'il est en position *i* s'il n'est pas en position *b*
- $AV \wedge MO = 0$ (9) : Interdire la commande *MO* et *AV* en même temps
- $RE \wedge DE = 0$ (10) : Interdire la commande *DE* et *RE* en même temps

En fonction de la technologie des pré-actionneurs des vérins, si ce sont des distributeurs 5/2, il faut ajouter :

- $\downarrow DE \wedge \neg b = 0$ (11) : Interdire de désactiver *DE* si le préhenseur n'est pas en *b*

- $\downarrow MO \wedge \neg h = 0$ (12) : Interdire de désactiver *MO* si le préhenseur n'est pas en *h*
- $\downarrow AV \wedge (\neg i \vee \neg a) = 0$ (13) : Interdire de désactiver *AV* si le préhenseur n'est pas en *a* ou en *i*
- $\downarrow RE \wedge (\neg i \vee \neg r) = 0$ (14) : Interdire de désactiver *RE* si le préhenseur n'est pas en *r* ou en *i*

2.4 Zone commune 2 chariots

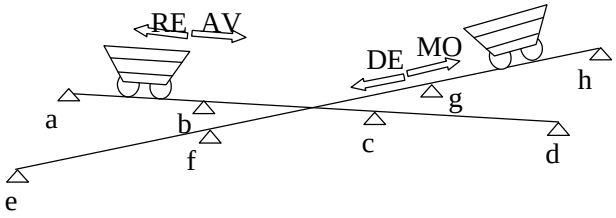


Figure A3.4. Chariots avec zone commune

Dans la configuration de la figure A3.4, les contraintes suivantes doivent être définies :

- $AV \wedge z5 \wedge c = 0$ (1) : Interdire l'envoi des commandes *AV* si le chariot *C2* se trouve dans la zone *z5*, et que le chariot *C1* est en *c*
- $RE \wedge z5 \wedge b = 0$ (2) : Interdire l'envoi de la commande *RE* si le chariot *C2* se trouve dans la zone *z5*, et que le chariot *C1* est en *b*
- $DE \wedge z2 \wedge f = 0$ (3) : Interdire l'envoi des commande *DE* si le chariot *C1* se trouve dans la zone *z2*, et que le chariot *C2* est en *f*
- $MO \wedge z2 \wedge g = 0$ (4) : Interdire l'envoi des commandes *MO* si le chariot *C1* se trouve dans la zone *z2*, et que le chariot *C2* est en *g*
- $AV \wedge DE \wedge c \wedge g = 0$ (5) : Interdire l'envoi des commandes *AV* et *DE* si les chariots se trouve en *c* et *g*
- $AV \wedge MO \wedge c \wedge f = 0$ (6) : Interdire l'envoi des commandes *AV* et *MO* si les chariots se trouve en *c* et *f*
- $RE \wedge DE \wedge b \wedge g = 0$ (7) : Interdire l'envoi des commandes *RE* et *DE* si les chariots se trouve en *b* et *g*
- $RE \wedge MO \wedge b \wedge f = 0$ (8) : Interdire l'envoi des commandes *RE* et *MO* si les chariots se trouve en *b* et *f*

- $\downarrow RE \wedge (\neg a \vee \neg b \vee \neg c) = 0$ (9) : Interdire de désactiver *RE* si le chariot C1 n'est pas sur un capteur *a*, ou *b*, ou *c*
- $\downarrow AV \wedge (\neg b \vee \neg c \vee \neg d) = 0$ (10) : Interdire de désactiver *AV* si le chariot C1 n'est pas sur un capteur *b*, ou *c*, ou *d*
- $\downarrow DE \wedge (\neg f \vee \neg g \vee \neg h) = 0$ (11) : Interdire de désactiver *DE* si le chariot C2 n'est pas sur un capteur *f*, ou *g*, ou *h*
- $\downarrow MO \wedge (\neg e \vee \neg f \vee \neg g) = 0$ (12) : Interdire de désactiver *MO* si le chariot C2 n'est pas sur un capteur *e*, ou *f*, ou *g*

3 Interaction produits

La configuration reprise ici, est celle de l'exemple du retourneur

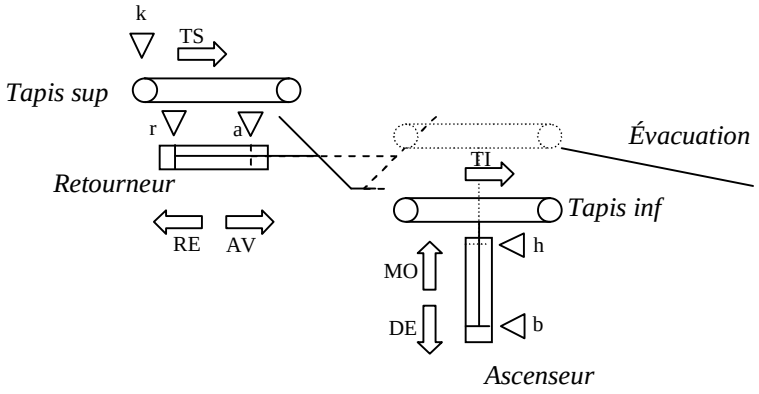


Figure A3.5. Transfert de pièces

3.1 Tapis retourneur

- $\uparrow TS \wedge pièce\_TS \wedge \neg r = 0$ (1) : Interdire d'activer *TS*, si le retourneur n'est pas en position *r* lorsqu'il y a une pièce sur le tapis supérieur
- $\uparrow TS \wedge pièce\_TS \wedge pièce\_retourneur = 0$ (2) : Interdire d'activer *TS*, s'il y a une pièce dans le retourneur lorsqu'il y a une pièce sur le tapis supérieur

- $TS \wedge AV \wedge pièce\_TS = 0$ (3) : Interdire l'activation en même temps de *TS* et *AV* lorsqu'il y a une pièce sur le tapis supérieur
- $\downarrow AV \wedge \neg a = 0$ (4) : Interdire la désactivation de *AV* si le retourneur n'est pas en *a*

3.2 Retourneur tapis

- $AV \wedge TI \wedge pièce\_retourneur = 0$ (1) : Interdire *AV* et *TI* en même temps, s'il y a une pièce dans le retourneur,
- $AV \wedge MO \wedge pièce\_retourneur = 0$ (2) : Interdire *AV* et *MO* en même temps lorsqu'il y a une pièce dans le retourneur. Cette contrainte est redondante a celle mise en place pour éviter la collision entre les deux vérins,
- $AV \wedge pièce\_retourneur \wedge \neg b = 0$ (3) : Interdire *AV* lorsqu'il y a une pièce dans le retourneur si l'ascenseur n'est pas en *b*.
- $AV \wedge pièce\_TI \wedge pièce\_retourneur = 0$ (4) : Interdire l'activation de *AV* lorsqu'il y a une pièce sur le tapis inférieur et une dans le retourneur,
- $\downarrow MO \wedge \overline{h} = 0$ (5) : Interdire la désactivation de *MO* si l'ascenseur n'est pas en *h*

3.3 Retourneur tapis ascenseur

- $AV \wedge TI \wedge pièce\_retourneur = 0$ (1) : Interdire *AV* et *TI* en même temps, s'il y a une pièce dans le retourneur,
- $AV \wedge MO \wedge pièce\_retourneur = 0$ (2) : Interdire *AV* et *MO* en même temps lorsqu'il y a une pièce dans le retourneur. Cette contrainte est redondante a celle mise en place pour éviter la collision entre les deux vérins,
- $AV \wedge pièce\_retourneur \wedge \neg b = 0$ (3) : Interdire *AV* lorsqu'il y a une pièce dans le retourneur si l'ascenseur n'est pas en *b*.
- $AV \wedge pièce\_TI \wedge pièce\_retourneur = 0$ (4) : Interdire l'activation de *AV* lorsqu'il y a une pièce sur le tapis inférieur et une dans le retourneur,
- $\downarrow MO \wedge \neg h = 0$ (5) : Interdire la désactivation de *MO* si l'ascenseur n'est pas en *h*

Résumé

Mots-clés : Validation de la commande, Système à Événements Discrets, Synthèse de la commande, Modélisation, Système manufacturier, Model-checking, Système Homme-Machine.

Le Conseil Régional de Champagne-Ardenne porte un intérêt particulier aux formations supérieures professionnalisantes en sciences et techniques. Dans ce cadre, il a été proposé de permettre l'utilisation de Systèmes Automatisés de Production (SAP) par des apprenants. Les travaux présentés dans cette thèse portent sur la conception de la commande de systèmes manufacturiers par des automaticiens au niveau de compétence variable, pouvant aller du novice jusqu'à l'expert. L'utilisation de Parties Opératives (PO) réelles, en particulier de façon distante à travers le web, par des automaticiens, qui sont tous susceptibles de commettre des erreurs de conception, soulève des problèmes originaux à la fois théoriques de validation et de vérification de la commande mais aussi de prise en compte du concepteur. Ces problèmes, que l'on retrouve dans l'industrie, nécessitent une approche de l'automatisation prenant en compte la composante humaine (expert - formateur, concepteur - apprenant) du Système Homme-Machine. Les contributions développées dans cette thèse portent sur l'adaptation du système aux savoirs du concepteur et la sécurisation de l'installation. Concernant le premier point, il est mis en évidence l'importance pédagogique de travailler sur toute l'installation plutôt que sur une partie. Une approche méthodologique est proposée, et consiste à décomposer fonctionnellement la PO en vue de pouvoir adapter le cahier des charges au concepteur, tout en conservant la vision globale du système manufacturier. Concernant le second point, deux approches sont envisagées pour assurer la sécurité de la Partie Opérative. La première, hors ligne, repose sur des travaux antérieurs de l'équipe « SED et Supervision du CReSTIC » dans le domaine de la synthèse par Ramadge et Wonham pour l'obtention d'une commande sûre, robuste et sans blocage. Celle-ci a été adaptée pour assurer d'une part que les modèles de PO et de spécifications sont justes, et d'autre part que la commande proposée respecte le comportement maximum admissible par le système, compte tenu des spécifications du cahier des charges. La seconde approche, en ligne, utilise un filtre, qui consiste à n'envoyer vers la PO que des commandes validées. La principale difficulté de ce type d'approche réside dans les spécifications du filtre, et dans son implémentation. Une structure de filtre à 2 niveaux, ayant des capacités explicatives pour le concepteur, est proposée. Le premier appelé « filtre de validation système » permet de sécuriser la PO et repose sur la spécification de contraintes logiques. Le second appelé « filtre de validation fonctionnel » permet de vérifier le respect du cahier des charges et n'a pas été développé dans cette thèse. Une erreur au niveau de la définition des spécifications de sécurité entraîne une validation de la commande erronée et met ainsi le système en danger. L'approche proposée pour s'assurer que les contraintes sont correctement définies, consiste à vérifier formellement, avant l'implémentation dans l'Automate Programmable Industriel (API), au moyen du model-checker UPPAAL, qu'elles sont nécessaires et suffisantes pour éviter toutes situations de détérioration de l'installation (éléments de PO et produit), et cela en considérant la commande la plus permissive possible. Le filtre de validation système revient à garantir un filtrage « robuste » sécuritaire des commandes. Deux applications du filtre « robuste » de commande sont présentées et permettent de montrer d'une part l'intérêt des concepts, méthodes et outils proposés et d'autre part leur applicabilité. La première concerne un système réel de conditionnement de médicaments. L'approche de sécurisation de la PO et d'adaptation au concepteur est mise en œuvre. Celle-ci a été testée et son intérêt montré au moyen d'expérimentations menées avec des automaticiens novices et plus expérimentés. La seconde application concerne une simulation de magasin automatisé. Le filtre est utilisé dans ce cas, non pas pour éviter les détériorations, mais pour apporter des explications. Ces exemples mettent aussi en évidence les limites et les perspectives de ces travaux.

Abstract

Keywords: Control validation, discrete event system, control synthesis, modelling, manufacturing system, model-checking, human-machine system

The regional council of Champagne-Ardenne has a particular interest for the professional higher education in science and technology. In this context, it was proposed to allow the use of automated production systems (APS) by learners. The work presented in this thesis, focuses on the control design by automatic control engineers who can have different levels of competence, ranging from novice to expert. The use of real plant, especially remotely through the web, can involve control design errors. This raises two original problems; the validation and verification of the control, and the requirement to take into the designer. These problems, which are also found in industry, require an approach taking into account the human part (expert - trainer, designer - learner) from the Human-Machine System. The contributions developed in this thesis focuse on adapting the system to designer knowledge and on the system safety. On the first point, it highlights the educational importance to work on the global system rather than a part. A methodological approach is proposed. It is based on the functional decomposition of the plant to adapt the specifications to the designer. This approach helps maintain the global vision of the manufacturing system. On the second point, two approaches are envisaged to ensure the plant safety. The first, offline, is based on previous work performed by the team "Discrete Event System and Supervision from CReSTIC", in the field of Wonham and Ramadge synthesis. Supervisory control allows to obtain a sure, robust and without deadlock controller. This approach has been adapted to ensure that the plant models and specifications are fair, and that the proposed controller respects the maximum permissible behaviour of the system. The second approach, online, uses a filter. This approach allows sending to the plant only validated Programmable Logic Controller (PLC) outputs. The main difficulties of designing the filter lie firstly in its specifications and implementation, and secondly in its explanatory capabilities for the designer. A filter structure with two levels is proposed. The first one called "system validation filter", assuring the plant safety, is based on logical constraints. The second one called "functional validation filter" enables to verify if specifications are conformed. This filter has not been developed in this thesis. An error in the definition of security specifications leads to a wrong validation of the control and can put the system in a dangerous mode. The proposed approach to ensure that the constraints are properly defined is to formally verify, before the implementation in the PLC, using the model-checker UPPAAL. The verification procedure determines the necessary and sufficient constraints set to avoid system deterioration situations (plant elements and parts), by considering the most possible permissive control. The system validation filter acts as a robust filter of PLC outputs. Two applications of control robust filter are presented and allow to show, on the one hand, a benefit of the concepts, methods and tools and on the other hand, the applicability. The first application deals with a real system of drugs packaging. The approach to secure the plant and to adapt the system to the designer is implemented. It has been tested and its interest has been shown through experiments conducted with novice and more experienced control engineers. The second application is based on a simulated automatic warehouse. The filter is used in this case, not to avoid plant elements and parts collisions, but to provide explanations. These examples also highlight the limits and the perspectives for this work.