Synthèse structurelle d'un contrôleur basée sur le Grafcet

Bassam Kattan

UNIVERSITÉ JOSEPH FOURIER – GRENOBLE 1

ECOLE DOCTORALE

Électronique, Électrotechnique, Automatique, Télécommunication et Signal

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

***Spécialité* : Automatique – Productique**

Présentée et soutenue publiquement par

Bassam KATTAN

le 3 septembre 2004

Synthèse structurelle d'un contrôleur basée sur le Grafcet

Sous la direction de M. Hassane ALLA (*Laboratoire d'Automatique de Grenoble*)

COMPOSITION DU JURY :

M. Christian COMMAULT	Professeur, ENSIEG, INP de Grenoble (Président).
M. Jean-Louis FERRIER	Professeur, Université d'Angers (Rapporteur).
M. François PRUNET	Professeur, Université de Montpellier 2 (Rapporteur).
M. Pascal YIM	Professeur, Ecole Centrale de Lille (Examinateur).
M. Eric RUTTEN	Chargé de recherche à l'INRIA Rhône-Alpes (Examinateur).
M. Hassane ALLA	Professeur, Université Joseph Fourier, (Directeur de thèse).

Thèse préparée au sein du LABORATOIRE D'AUTOMATIQUE DE GRENOBLE

Remerciements

Le travail de thèse présenté dans ce mémoire a été réalisé au Laboratoire d'Automatique de Grenoble dans l'équipe de Conception des Systèmes Sûrs (CSS).

Je remercie tout d'abord Monsieur **C. Commault** professeur à l'INPG, pour l'honneur qu'il me fait en présidant le jury.

Je remercie sincèrement tous les membres du jury d'avoir accepté de porter un avis sur ce mémoire.

Je remercie vivement mon directeur de thèse le professeur **Hassane ALLA** pour la confiance qu'il a su m'accorder en acceptant d'encadrer mes travaux de recherche et qui n'a pas cessé de me faire profiter de sa haute compétence, son aide efficace et ses conseils pertinents. Je lui suis très reconnaissant de sa patience surtout pendant la rédaction de ce mémoire.

J'adresse mes remerciements les plus chaleureux aux membres de l'équipe de CSS ainsi à Mme Zineb Simeu-Abazi et Mme Maria DiMascolo pour leur soutien amical qui m'ont fait bénéficier d'une ambiance de travail très agréable.

C'est le moment aussi pour dire un grand merci à tous ceux qui m'ont aidé pour que je puisse réaliser mes études supérieures par leurs encouragements. Je cite surtout mon père, ma mère, ma sœur, mes frères, et tous mes amis. j'exprimer mes plus sincères gratitudes à toutes ces personnes et leur dire que vous étiez toujours présents ici avec votre humour, votre charme et votre chaleur.

Enfin, à ma femme que la réalisation de ce mémoire doit à sa présence, à sa confiance et à sa patience sans limite.

Encore une fois, merci à toutes et à tous.

Table des Matières

Table des Matières

Introduction Générale

INTRODUCTION GENERALE

L’évolution technologique a conduit au développement de systèmes de production complexes, dont l’impact socio-économique est devenu très fort, dans la mesure où ils occupent des places de plus en plus stratégiques au sein des organisations. De tels systèmes intègrent de nombreux composants matériels et logiciels et interagissent avec des environnements complexes. Ainsi la synthèse de la commande de ces systèmes devient de plus en plus difficile, et elle ne peut être basée sur des essais et corrections successives, coûteux et approximatifs. L'expérience et la perspicacité de l'ingénieur qui conçoit ces systèmes deviennent insuffisantes pour imposer le respect du cahier des charges. Par conséquent, il est nécessaire de disposer d’outils aidant à la conception de systèmes de commande et permettant de vérifier si le système commandé répond bien au cahier des charges.

L’objectif de cette thèse est de proposer une méthode de synthèse d'un superviseur/contrôleur qui impose au système le respect des spécifications logiques de fonctionnement.

Pour atteindre cet objectif il faut d’abord modéliser le système (l’objet physique) que l’on désire contrôler, et ensuite exprimer les objectifs de commande (spécifications) et finalement utiliser des techniques mathématiques qui permettent d’élaborer les contrôleurs correspondants.

La théorie classique des systèmes continus (y compris en temps discret) et de l’automatique s’intéresse à des systèmes obéissant essentiellement aux lois de la physique, et descriptibles par des équations différentielles ou aux dérivées partielles (ou leur discrétisation approchée en temps). Le vocable systèmes à événements discrets (SED) recouvre des systèmes également dynamiques, mais dont la dynamique échappe totalement à ce genre de description. En réalité, au lieu de s’intéresser au déroulement continu des phénomènes, on ne se soucie que des ‘’débuts’’ et des ‘’fins’’ de ces phénomènes (les événements discrets) et de leur enchaînement dynamique, logique ou temporel. Les modèles SED sont utilisés dans le domaine de la production manufacturière, la robotique, les trafics des véhicules, la logistique, les réseaux de communications, etc.

L’étude des systèmes à événements discrets peut être menée avec différents outils tels que la simulation sur ordinateur, réseaux de files d’attente, les langages de programmation parallèle/temps réel, des modèles dynamiques algébriques, comme l'algèbre ‘’Max Plus’’, et

finalement les réseaux de Petri, les automates et les langages qui reposent sur l'ordre exact d'occurrence des événements.

Le mot *supervision* peut comprendre plusieurs aspects comme la sûreté de fonctionnement, la surveillance, la gestion de modes, et le pilotage. Dans notre recherche, le mot supervision concerne non pas l'analyse des défaillances du système mais son fonctionnement normal. La supervision concerne alors l'automatisation. Elle recouvre la notion de synthèse du superviseur, c'est à dire l'élément qui va contraindre le système à respecter un ensemble de comportements définis dans un cahier des charges. C'est la vision de la théorie de la supervision qui a été initiée par les travaux de Ramadge et Wonham dans le début des années 80 [RAM 83], elle repose sur la théorie des langages formels et la théorie des automates.

Dans la théorie de la supervision, le système complet est généralement divisé en deux parties : le procédé et le superviseur. Le procédé, un SED, est la partie du système qui doit être supervisée. Le comportement du procédé sans le superviseur (boucle ouverte) est souvent jugé non satisfaisant à l'égard de ce qui est souhaité. Il est possible de modifier dans une certaine limite, le langage généré par le système, par l'ajout d'une structure particulière appelée superviseur. Le rôle du superviseur est d'assurer que les aspects non satisfaisants du comportement du procédé ne se produiront pas. Le comportement souhaité est précisé, en imposant que le langage généré par le procédé couplé au superviseur appartienne à un certain langage de spécification. En détectant chacune des évolutions du procédé, donc en connaissant l'état courant du procédé, le superviseur choisit l'action ''corrective'' appropriée. Cette action se traduit par des lois de contrôle transmises au procédé. Ces lois de contrôle contiennent les événements dont l'occurrence est autorisée depuis l'état courant du procédé. Le superviseur intervient sur l'évolution du procédé uniquement en interdisant l'occurrence de certains événements, c'est-à-dire en ne les faisant pas apparaître dans une (ou plusieurs) loi de contrôle. Une autre distinction essentielle entre le procédé et le superviseur est que le procédé est donné, il correspond à un système existant qui ne peut être changé, tandis que le superviseur est un système à construire, donc plus souple.

L'approche de R&W est basée sur la modélisation des systèmes par des automates à états finis et des langages formels. Cependant, le grand nombre d'états à considérer pour représenter le comportement du système ainsi que le manque de structure dans les modèles limite la possibilité de développer des algorithmes de calcul efficaces pour l'analyse et la synthèse des systèmes réels. En outre l'utilisation de cette approche a montré une difficulté dans l'implémentation du système supervisé.

Pour pallier ces difficultés, plusieurs méthodes de synthèse de commande ont été définies parmi lesquelles : la commande supervisée. Ce nouveau concept permet de séparer clairement la commande et la supervision. Dans cette approche le procédé couplé au système de commande, que l'on appelle procédé étendu apparaît comme un système générant spontanément des événements. Le rôle du superviseur consiste alors uniquement à interdire au procédé étendu de produire certains événements appelés contrôlables. François Charbonnier [CHA 96] a montré dans son travail que si le langage des spécifications de supervision est contrôlable par rapport au langage du procédé étendu, le Grafcet de spécification lui-même constitue le Grafcet de superviseur. Cependant, le problème n'est pas résolu dans le cas où le langage des spécifications est non contrôlable.

Le concept de commande supervisée est indépendant de l'outil de modélisation utilisé. Nous allons utiliser le Grafcet tout au long de notre travail. Cet outil est conçu au départ comme outil de spécification du cahier des charges. Il est devenu également un outil pour la synthèse de la commande puis un langage de programmation des automates programmables. L'utilisation du Grafcet dans la synthèse de commande permet de réaliser (exécution physique) le superviseur abstrait obtenu par la synthèse.

Etant donné un ensemble de spécifications de fonctionnement, nous définissons puis modélisons sous forme de Grafcet le procédé étendu et les spécifications de supervision.

La contribution de notre travail dans ce contexte a porté sur plusieurs aspects. Dans un premier temps, nous développons une approche pour synthétiser un superviseur dans le cas où le langage des spécifications de supervision est non contrôlable par rapport au langage du procédé étendu sous supervision. Nous proposons également une méthode systématique de passage de l'automate superviseur vers le Grafcet superviseur. Cependant dans cette approche, un inconvénient majeur subsiste : Le Grafcet final de superviseur obtenu par cette approche contient un nombre important d'étapes qui est identique à l'automate synthétisé. On imagine alors la complexité des superviseurs que l'on obtient dans le cas d'applications complexes.

Dans un deuxième temps, notre contribution a consisté à proposer la synthèse du Grafcet de superviseur directement à partir d'un Grafcet de spécification. Nous avons obtenu ainsi des mécanismes de construction structurelle beaucoup plus aisés à manipuler et qui permettent d'éviter l'explosion combinatoire du nombre d'états dans le modèle final. Ce résultat permet la synthèse d'un contrôleur basée sur l'exploitation des invariants de marquage des réseaux de Petri. L'idée est d'ajouter des places de contrôle pour respecter les

spécifications. Nous verrons que l'utilisation du Grafcet nous apportera une propriété intéressante ouvrant la voie à la synthèse d'un contrôleur optimal.

Notre objectif dans ce travail est d'étudier les SED en vue de leur commande. Pour cela il nous a paru nécessaire de commencer par une présentation générale des SED. C'est l'objet du premier chapitre. Nous faisons une présentation des aspects logiques des SED. Tout d'abord nous évoquons quelques notions relatives aux langages formels. Puis nous présentons les outils les plus usuels dans le champ de commande des SED tels que les automates, les Réseaux de Petri et le Grafcet.

Le chapitre deux présente dans un premier temps les principales méthodes existantes dans la littérature sur la commande des systèmes à événements discrets. La théorie de la supervision des SED proposée par Ramadge & Wonham est d'abord présentée en détail. Elle est à l'origine de la commande par supervision. Puis deux méthodes de synthèse de la supervision sont décrites. La première combine l'approche de Ramadge & Wonham et la théorie des régions, et la deuxième exploite les invariants de marquage de réseaux de Petri pour synthétiser le système de commande.

Nous traitons dans le troisième chapitre les problèmes liés à l'utilisation de la théorie de Ramadge & Wonham pour la conception de commande de SED ainsi que les différents travaux ayant traité ces problèmes. Puis l'approche de commande supervisée des SED, proposée par François Charbonnier est introduite. Nous avons complété cette approche dans le cas où le langage des spécifications est non contrôlable.

Le quatrième chapitre constitue notre contribution principale. C'est une approche originale de synthèse d'un contrôleur basée sur le Grafcet. Cette approche permet de déterminer des places de contrôle de manière structurelle. En particulier, nous allons montrer comment construire de manière systématique les contraintes linéaires à partir des états interdits. Nous présentons ainsi quelques propriétés qui garantissent l'optimalité du contrôleur obtenu. Enfin, nous terminerons notre mémoire par une conclusion et les perspectives de ce travail.

Chapitre 1

Généralités sur les systèmes à événements discrets

CHAPITRE 1

Généralités sur les systèmes à événements discrets

Dans ce premier chapitre nous faisons un survol des aspects logiques des systèmes à événements discrets. Nous évoquons tout d'abord quelque notions relatives aux langage formels. Puis nous nous intéresserons aux outils les plus usuels dans le champ de commande des SED tels que les automates, les Réseaux de Petri et le Grafcet. Dans la suite nous allons présenter des définitions formelles pour ces trois outils.

Chapitre 1

Généralités sur les systèmes à événements discrets

1.1 Introduction

Alors que la théorie classique des systèmes continus (y compris en temps discret) et de l'Automatique s'intéresse à des systèmes obéissant essentiellement aux lois de la Physique, et descriptible par des équations différentielles ou aux dérivées partielles (ou leur discrétisation approchée en temps), le vocable systèmes à événements discrets (SED) recouvre des systèmes également dynamiques, mais dont la dynamique échappe totalement à ce genre de description. En réalité, au lieu de s'intéresser au déroulement continu des phénomènes, on ne se soucie que des ''débuts'' et des '' fins'' de ces phénomènes (les événements discrets) et leur enchaînement dynamique, logique ou temporel. Les modèles SED sont utilisés dans le domaine de la production manufacturière, la robotique, les trafics de véhicules, la logistique, les réseaux de communications, etc.

La plupart des systèmes physiques cités ci-dessus présente des caractéristiques communes telles que le parallélisme, la synchronisation et la concurrence.

Les modèles SED peuvent être utilisés à différents niveaux :

- *Spécification.* Avant de concevoir un système, il faut déterminer ce qu'on veut lui faire faire, que doit être sa réponse dans certain nombre de situations-type, etc. ?

- *Conception, architecture*. Une fois spécifié le comportement fonctionnel du système, il faut le concevoir, notamment du point de vue de son architecture : composants, agencement et articulations, mécanismes de synchronisation et d'exécution.

- *Validation logique.* Il faut ensuite vérifier que le système ainsi conçu répond bien aux spécifications désirées, et qu'il n'engendre pas d'autres comportements indésirables.

- *Evaluation de performance. A* cette étape, la notion de temps intervient. On cherche alors à répondre à des questions du type : combien d'événements d'un type donné se produisent en une heure, à quelle date de produira le *n*-ème événement, etc. ?

- *Ordonnancement.* L'ordonnancement a pour but d'établir des politiques de priorité, de routage, etc. destinées à résoudre les problèmes posés par les phénomènes de concurrence.

Pour conclure, la théorie des systèmes à événement discrets peut être divisée actuellement en deux grandes approches [CAS 99]:

- L'approche *logique* qui ne s'intéresse qu'à l'occurrence des événements ou l'impossibilité de cette occurrence et à la succession de ces événements, mais pas à la date précise de ces occurrences, autrement dit pas aux aspects de performance ; Ramadge et Wonham [RAM 89a][WON 94] ont utilisé cette approche pour aborder la problématique de la commande qui fait l'objet de nos recherches, et dont il est donc essentiellement question dans la suite de ce mémoire.

- L'approche *quantitative* qui s'adresse à l'aspect évaluation de performance voire à l'optimisation de ces performances ; dans ce contexte général, on peut distinguer deux autres approches : la première est l'approche d'analyse de perturbation initiée par Y.C. Ho [Ho 91] qui cherche à résoudre toutes sortes de problèmes d'optimisation non classiques, en raison de l'aspect *"événements discrets"*, et ce généralement dans un contexte stochastique. La deuxième est l'approche *"Max Plus"* qui se caractérise par l'utilisation d'une *algèbre adaptée*, ou plus généralement, l'algèbre des *dioïdes* (ou *semi-anneaux)* [COH 97].

L'étude des systèmes à événement discrets peut être menée avec différents outils tels que la simulation sur ordinateur, réseaux de files d'attente, les langages de programmation parallèle/temps réel et finalement des modèles dynamiques algébriques, comme l'algèbre "Max Plus" qui aide à étudier certaines classes des SED (graphes d´événements temporisés) il prend en compte l'aspect quantitatif (évaluation de performance), et s'appuie sur des modèles mathématiques tout à fait analogues aux modèles utilisés en automatique classique. Cette classe de systèmes n'a pas traitée dans ce mémoire.

Certains outils tels que les automates et les langages, le Grafcet, et les réseaux de Petri ont prouvé leur efficacité pour la commande des SED. Nous avons retenu ces outils dans le cadre de notre recherche, c'est pour cela que nous les présentons en détail dans la suite de ce chapitre. Nous justifierons par la suite l'utilisation conjointe de ces outils.

1.2 Modèle logique de systèmes à événements discrets

Définition 1-1

Un *Système à Evénements Discrets* est un système dans lequel l'espace des états est discret. Un tel système est à opposer à un système continu pour lequel l'état est représenté par des grandeurs qui prennent des valeurs dans un domaine continu. Les SED sont des systèmes qui sont fondamentalement asynchrones, c'est-à-dire que ceux-ci ne sont pas cadencés par une horloge. La dynamique de ces systèmes est assurée par l'occurrence des événements, ils se produisent de manière instantanée. En l'absence d'événement, l'état du système demeure inchangé. L'évolution du temps entre deux occurrences ne provoque aucun effet détectable sur le système.

L'évolution d'un SED peut être décrite par un ensemble de couples : (*e*,*t*) où "*e*" représente un événement et "*t*" représente l'instant d'occurrence de cet événement. Un ensemble ordonné de couples constitue ce que l'on appelle une *séquence*. Une telle description se place à un niveau temporel dans le sens où l'instant d'occurrence des événements est une information considérée comme pertinente.

En revanche, si l'on considère un *modèle logique* pour décrire le SED, seul l'ordre d'occurrence des événements importe.

En général, un SED peut avoir un comportement non déterministe. Nous entendons ici le "non-déterminisme" par le fait que nous ne pouvons pas prévoir, a priori, quelle sera l'évolution du système. En d'autres termes, pour un état donné du système, plusieurs événements différents sont supposés susceptibles de se produire. Donc une séquence unique ne suffit alors plus pour décrire le comportement du système. Ainsi, l'évolution d'un SED sera en général décrite par un ensemble de séquences d'événements. Cet ensemble de séquences constitue un *langage* sur l'ensemble des événements possibles dans le système.

Dans la suite, nous nous intéresserons uniquement à des modèles logiques.

1.3 Langage et automate

1.3.1 Evénement, chaîne et langage

Pour formaliser des systèmes à événements discrets sous forme de langage, on représente les événements par des symboles. L'ensemble de tous ces symboles (σ_1 , …, σ_n) est fini et constitue un alphabet noté Σ.

Toutes les séquences finies d'événements, ou trace, peuvent alors être représentées par une séquence de symboles $s = \sigma_1.\sigma_2$ … appelée chaîne (ou mot) sur l'alphabet Σ.

On appelle alphabet (ou vocabulaire), un ensemble fini de symboles noté Σ. Dans le cas d'un SED, l'alphabet pourra représenter l'ensemble des événements possibles dans le système. Cet ensemble est composé de tous les événements qui font évoluer le SED.

Une *chaîne* (ou *mot* ou *séquence*) définie sur un alphabet Σ est une suite finie d'éléments de Σ notée *s*. La longueur d'une chaîne *s*, notée |*s*|, représente le nombre de symboles de *s* (par exemple, |abba| = 4).

La relation entre un système et sa traduction sous forme de langage peut être comme suit :

- Evénement → symbole $\sigma \in \Sigma$
- Trace → séquence *s*
- Comportement du système → langage *L*

Soit Σ^* l'ensemble de toutes les chaînes qui peuvent être formés sur les éléments de Σ, y comprises la chaîne vide ε. La concaténation de deux chaînes *v* et *x* est notée par *vx*. Nous disons que la chaîne *v* est un préfixe de la chaîne *vx*.

1.3.2 Langage

Définition 1-2

On appelle *langage* défini sur un alphabet Σ, tout sous-ensemble de Σ*.

Nous pouvons à présent définir la *préfixe-clôture (la fermeture préfixielle)* d'un langage *L*, comme le langage contenant tous les préfixes des chaînes de *L*. nous noterons par $\overline{L}$ la préfixe-clôture du langage *L*.

$$\overline{L} = \{ s_1 \in \Sigma^* \mid \exists\, s_2 \in \Sigma^* , s_1 s_2 \in L \}$$

1.3.3 Langage préfixe-clos

Définition 1-3

Un langage *L* est *préfixe-clos(fermé)* s'il est égal à sa préfixe clôture.

C'est-à-dire, *L* est préfixe-clos si $L = \overline{L}$. Un langage préfixe-clôs contient toute l'évolution passée du SED.

Depuis un état *q*, l'occurrence de l'événement σ est associé à un changement d'état, l'état courant devient $q' = \delta(q,\sigma)$. La relation de transition δ peut être étendue pour s'appliquer à une chaîne *s* : $q' = \delta(q, s)$ où *q'* est l'état atteint quand la chaîne *s* est exécutée depuis l'état *q*. Cette dernière équation peut être explicitée : d'une manière générale, si *s* est une chaîne et σ un événement $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$.

Si les concepts de langages sont familiers aux informaticiens, ils le sont nettement moins dans la communauté des automaticiens. De plus, il est souvent malaisé de modéliser des SED sous forme de langages, qui s'avèrent trop abstraits pour être utilisés directement. La mise en oeuvre exige de se limiter à certaines classes de langages, pratiquement les langages réguliers.

Il apparaît donc nécessaire d'utiliser pour représenter les SED des modèles intermédiaires tels que les automates, les réseaux de Petri, et Grafcet. Le modèle des SED le plus proche des langages est incontestablement le modèle automate à espace d'états fini

1.3.4 Automate fini :

Un automate est une machine à états qui permet de décrire le fonctionnement entrées/sorties d'un système à événements discrets. Un automate est représenté par une succession d'états et de transitions, ces dernières sont associées à des événements.

De façon générale, un automate est une machine qui a des entrées et des sorties discrètes et qui réagit à une modification de ses entrées en changeant ses sorties. Formellement un automate fini *A* peut être défini de la façon suivante :

Définition 1-4

Un automate fini *A* est un 5-tuplet $A = (Q, \Sigma, \delta, q_0, Q_m)$ où :

- *Q* est l'ensemble fini des états.

- Σ est un ensemble fini de symboles d'entrée (ensemble des événements), appelé alphabet d'entrée.

- δ est la fonction de transition d'états de Q x Σ vers Q qui associe un état d'arrivée à un état de départ et à un symbole d'entrée.

- $q_0 \in Q$ est l'état initial.

- $Q_m \subseteq Q$ est l'ensemble des états marqués.

Un automate peut être représenté par son *graphe de transition d'états*. Le graphe de transition d'états d'un automate fini *A* est donné dans la Fig. 1.1.

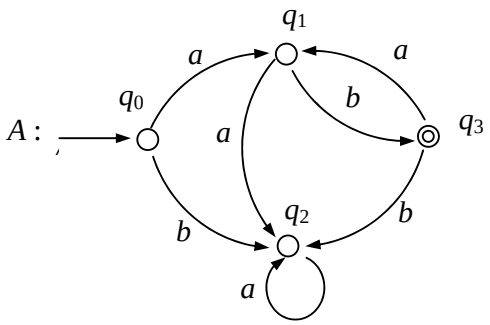


Figure 1.1. ***Graphe de transition d'états d'un automate fini.***

Dans la Fig. 1.1, les états de *A* sont représentés par des cercles. Nous avons : $Q = \{q_0, q_1, q_2, q_3\}$. *L'* état initial q_0, est repéré par une flèche entrante. Les états finals sont représentés par des doubles cercles, ainsi : $Q_m = \{q_3\}$. La fonction δ de transition d'états est représentée par des arcs associés à des symboles de l'alphabet Σ. Dans notre exemple, l'alphabet Σ correspond à l'ensemble {a, b}. Il existe une transition d'états associée au symbole "a" entre q_0 et q_1. Cela signifie : $\delta(q_0, a) = q_1$.

- Un état $q \in Q$ est dit accessible s'il existe une chaîne $s \in \Sigma^*$ telle que $q = \delta(q_0, s)$, c'est-à-dire que l'automate peut y accéder depuis l'état initial. Par extension, l'automate *A* est accessible si tout état $q \in Q$ est accessible.

- Un état $q \in Q$ est dit co-accessible s'il existe une chaîne $s \in \Sigma^*$ telle que $\delta(q, s) \in Q_m$, c'est-à-dire qu'à partir de cet état l'automate peut atteindre un état marqué. Par extension, l'automate *A* est co-accessible si tout état $q \in Q$ est co-accessible.

Selon que l'on implique les états marqués ou non marqués, l'automate *A* définit deux classes importantes de langages : $L(A)$ ou $L_m(A)$.

1.3.5 Langage généré d'un automate fini

Définition 1-5

Le langage généré par un automate fini A est donné par :

$$L(A) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q\}$$

Ce langage représente l'ensemble de toutes les chaînes qui permettent de rejoindre un état quelconque de l'automate à partir de son état initial. Donc le langage généré par un automate est toujours préfixe-clos $\overline{L}(A) = L(A)$.

1.3.6 Langage marqué d'un automate fini

Définition 1-6

Le langage marqué d'un automate fini A est donné par :

$$L_m(A) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$$

Ce langage représente l'ensemble de toutes les chaînes qui permettent de rejoindre un état marqué à partir de l'état initial. De telles chaînes sont appelées *tâches*.

Cette définition implique que $L_m(A) \subseteq L(A)$, puisque $Q_m \subseteq Q$.

1.3.7 Opérations sur les automates et les langages

On considère deux langages L_1 et L_2 construits sur un même alphabet Σ.

L'intersection de L_1 et L_2 est définie sur Σ par :

$$L_1 \cap L_2 = \{ s \mid s \in L_1 \text{ et } s \in L_2 \}$$

$L_1 \cap L_2$ désigne le langage contenant toutes les chaînes qui appartiennent à la fois à L_1 et à L_2.

L'union de L_1 et L_2 notée, $L_1 \cup L_2$ est le langage contenant toutes les chaînes qui sont soit contenues dans L_1, soit contenues dans L_2. formellement le langage, $L_1 \cup L_2$ est défini sur Σ par :

$$L_1 \cup L_2 = \{ s \mid s \in L_1 \text{ ou } s \in L_2 \}$$

La concaténation de L_1 et L_2 est défini sur Σ par :

$$L_1 L_2 = \{ s \mid s = uv, u \in L_1 \text{ et } v \in L_2 \}$$

Le modèle automate, ainsi que sa relation avec les langages ont été exposés. Il existe plusieurs opérations réalisables sur les automates. Le plus important, notamment pour la synthèse du superviseur est probablement la composition d'automates.

1.3.8 Composition d'automates

Il existe deux sortes de compositions :

• *la composition synchrone,* qui est effectuée quand les alphabets des langages associés aux automates considérés ont au moins un événement en commun,

• *La composition asynchrone*, qui est réalisée quand les alphabets des langages associés aux automates considérés n'ont aucun événement en commun.

Soient deux automates A_1 A_2 définis respectivement par :

$A_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$ et par $A_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$

La composition synchrone de A_1 et de A_2 est notée $A_1 \mid\mid_s A_2 = (Q , \Sigma, \delta, q_0, Q_m)$ où :

- $Q = Q_1 \times Q_2$ est l'ensemble des états.
- $\Sigma = \Sigma_1 \cup \Sigma_2$ représente l'alphabet.
- $q_{0,1} \times q_{0,2}$ est l'état initial.
- $Q_m = Q_{m,1} \times Q_{m,2}$. est l'ensemble des états marqués
- Pour tout $q = (q_1, q_2) \in Q$, pour $\sigma \in \Sigma$, la relation de transition est telle que :

$$\delta_{1||_s 2}(q, \sigma) = \begin{cases} \delta_1(q_1, \sigma) \times \delta_2(q_2, \sigma) & \text{si } \delta_1(q_1, \sigma)! \wedge \delta(q_2, \sigma)! \\ \delta_1(q_1, \sigma) \times \{q_2\} & \text{si } \delta_1(q_1, \sigma)! \wedge \sigma \notin \Sigma_2 \\ \{q_1\} \times \delta_2(q_2, \sigma) & \text{si } \delta_2(q_2, \sigma)! \wedge \sigma \notin \Sigma_1 \\ \varnothing & \text{sinon} \end{cases}$$

Si un événement σ, appartient à $\Sigma_1 \cap \Sigma_2$, il doit se produire de manière synchrone dans les deux automates, par contre, s'il n'appartient qu'à un ensemble, il se produit de manière asynchrone.

La composition asynchrone peut être vue comme un cas particulier de la composition synchrone. La composition asynchrone de A_1 et de A_2 est notée $A_1 \| A_2 = (Q , \Sigma, \delta, q_0, Q_m)$ où et les notations sont identiques aux précédentes, La relation de transition, pour tout $q = (q_1, q_2) \in Q$, et pour tout $\sigma \in \Sigma$, est définie par :

$$\delta_{1||2}(q, \sigma) = \begin{cases} \delta_1(q_1, \sigma) \times \{q_2\} & \text{si } \delta_1(q_1, \sigma)! \wedge \sigma \notin \Sigma_2 \\ \{q_1\} \times \delta_2(q_2, \sigma) & \text{si } \delta_2(q_2, \sigma)! \wedge \sigma \notin \Sigma_1 \\ \varnothing & \text{sinon} \end{cases}$$

La modélisation des SED réels est généralement difficile à mettre en oeuvre du fait de l'explosion combinatoire du nombre d'états (même dans les petits systèmes) et même si les automates apparaissent comme un formalisme approprié permettant de modéliser efficacement une large classe de systèmes à événements discrets, certaines applications manufacturières s'expriment plus facilement sous forme de réseaux de Petri. Les réseaux de Petri (RdP) ont, par rapport aux automates, l'avantage d'être un modèle beaucoup plus général, bénéficiant de structures beaucoup plus riches, s'adaptant parfaitement à la description de certains types de SED.

Dans le prochain paragraphe nous allons faire quelques rappels sur les réseaux de Petri. Pour plus de détails sur les réseaux de Petri, on pourra se reporter à [DAV 92].

1.4 Réseaux de Petri

1.4.1 Notions de Base

Un réseau de Petri (RdP) comporte deux types de nœuds : les places et les transitions. Une place est représentée par un cercle et une transition par un trait. Les places et transition sont reliées par des arcs. Le nombre de places est fini, et non nul. Le nombre de transitions est également fini et non nul. Un arc est orienté. Il relie soit une place à une transition soit une transition à une place.

Autrement dit, un RdP est un graphe biparti, c'est-à-dire qu'il y a alternance des places et des transitions sur un chemin formé d'arcs consécutifs. Et tout arc doit obligatoirement avoir un nœud à chacune de ses extrémités.

Définition 1-7

Un RdP ordinaire non marqué est une quadruplet $Q = < P, T,$ Pré, Post > tel que:

$P = \{P_1, P_2, \ldots, P_n\}$ est un ensemble fini et non vide de places;
$T = \{T_1, T_2, \ldots, T_m\}$ est un ensemble fini et non vide de transitions,

$P \cap T = \varnothing$; c'est-à-dire que les ensembles P et T sont disjoint;
Pré : P x $T \rightarrow \{0,1\}$ est l'application d'incidence avant;
Post : P x $T \rightarrow \{0,1\}$ est l'application d'incidence arrière.

Pré(P_i,T_j) est le poids de l'arc $P_i \rightarrow T_j$, ce poids est égal à 1 si l'arc existe et 0 sinon.

Post (P_i,T_j) est le poids de l'arc $T_j \rightarrow P_i$. Les applications "Pré" et " Post" sont relatives à la transition T_j du couple (P_i, T_j).

Lorsque le poids des éléments dans les matrices Pré , Post est soit nul, soit égal à 1, le réseau est dit sauf. Nous nous limitons à la présentation de ce cas qui correspond à celui du Grafcet.

1.4.2 Le Marquage de RdP

La distribution des jetons dans les places détermine le marquage du réseau. Ce marquage M associe un nombre entier $M(P_i)$ (positif ou nul) à chaque place P_i du réseau.

Le marquage d'un réseau de Petri est modélisé par une fonction $M_i : P \rightarrow \mathbb{N}$, où $M_i(P)$ représente le nombre de jetons contenus dans la place P au $i^{ème}$ marquage. On dira qu'un marquage M est binaire si toutes les places du réseau contiennent au plus un jeton.

- Il est donc défini par le vecteur de ces marquages soit $M = (M(P_1), M(P_2), ..., M(P_i),..., M(P_n))$. Dans ces réseaux, les jetons n'ont pas d'identité et ne peuvent être distingués entre eux.

Dans l'exemple de la Figure 1.2, nous avons :

- l'ensemble des places $P = \{P_1, P_2, P_3, P_4, P_5\}$.
- l'ensemble des transitions $T = \{T_1, T_2, T_3, T_4\}$.
- le marquage $M = (0, 0, 0, 1, 1)$.

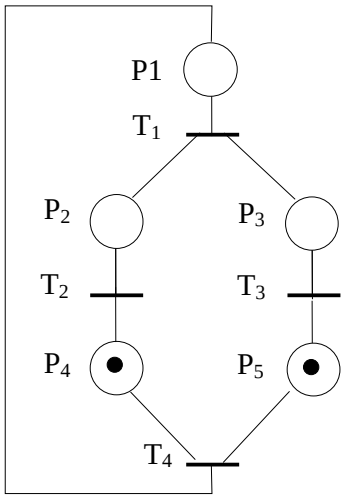


Figure 1.2 Réseaux de Petri places-transitions

Un certain marquage est considéré à priori comme le marquage initial. A partir de ce marquage, il est possible de faire évoluer le réseau en suivant certaines règles. Donc L'état du RdP défini par son marquage peut évoluer par le franchissement de transitions.

1.4.3 Franchissement d'une transition

Le franchissement d'une transition n'est autorisé que lorsque chacune des places en amont de cette transition contient au moins un jeton. Le franchissement d'une transition fait évoluer le marquage du réseau en retirant un jeton dans chacune des places en amont de la transition et en ajoutant un jeton dans chacune des places aval de la transition. Il y a un seul franchissement à la fois, et la durée de ce franchissement est nulle.

Dans l'exemple de la figure 1.2 le marquage M devient, en franchissant T_4, M' = (1 0 0 0 0).

Aux réseaux de Petri, il fallait associer une façon normalisée de définir comment le système de commande interagissait avec l ‘environnement au moyen des entrées et des sorties (capteurs et d’actionneurs). Il fallait donc un norme pour l’interprétation des réseaux de Petri. C’est ce qui a donné les réceptivités associées aux transitions et les action associées aux étapes dans le Grafcet. C'est dans cet esprit que le Grafcet a été conçu. Il est aujourd'hui largement utilisé dans l'industrie, normalisé.

1.5 Le Grafcet

Dans le cadre de l'ingénierie des systèmes automatisés de production, concevoir une commande de système à événements discrets est une activité complexe. Elle nécessite

souvent l'utilisation de méthodes qui mettent en œuvre un ensemble de langages de spécification. Par ailleurs, on observe que les représentations graphiques sont de plus en plus privilégiées par rapport aux représentations littérales et algébriques du fait de leur aptitude à servir de support de communication entre tous les intervenants. C'est dans cet esprit que le Grafcet a été conçu. Il est aujourd'hui largement enseigné, utilisé dans l'industrie, et il est servi par de nombreuses publications.

Le Grafcet (Graphe Fonctionnel de Commande Etape/Transition) est un graphe qui permet de décrire facilement les fonctionnalités d'un automatisme séquentiel [BLA 79]. Il a été normalisé, dans un premier temps, par l'AFNOR (Association Française de NORmalisation) en 1977 et ensuite par la CEI (Commission Electronique Internationale) en 1989.

1.5.1 Eléments de base.

Le Grafcet est un graphe constitué de séquences d'étapes et de transitions reliées par des liaisons orientées (figure 1.3).

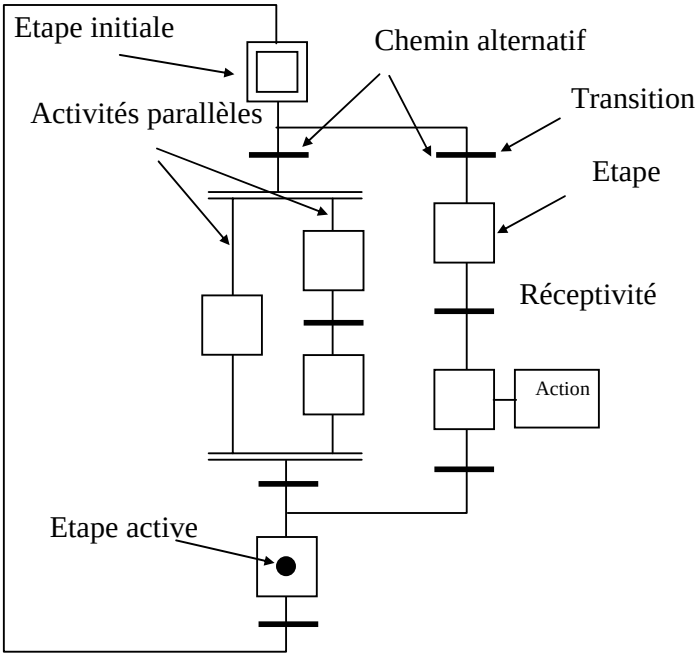


Figure 1.4 Concepts de base du Grafcet

Étape : L'étape représente un état dans lequel l'automatisme est invariant vis à vis de ses entrées/sorties. Elle peut être active ou inactive. L'état du Grafcet est défini, à un instant donné, par l'ensemble de ses étapes actives.

Transition : La transition traduit la possibilité d'évolution d'un état vers un autre. Cette évolution est la conséquence du franchissement de la transition. Une transition est validée si toutes ses étapes immédiatement amont sont actives.

Liaison orientée : Une liaison orientée relie une étape à une transition et inversement. Elle indique les configurations atteignables à partir d'un état donné.

1.5.2 Interprétation du graphe.

L'interprétation d'un graphe ainsi constitué revient à associer des actions aux étapes et des réceptivités aux transitions, traduisant l'aspect combinatoire de l'automatisme.

Les actions : L'action spécifie ce qui doit être fait lors de l'activation de l'étape. Une action peut être interne (compteur, armement de temporisation) ou externe (sortie de l'automate).

Les réceptivités : La réceptivité est une expression booléenne qui peut prendre les valeurs vraies ou fausses en fonction de l'état ou des changements d'état des variables qui la composent. La réceptivité conditionne le franchissement de la transition. Une variable peut être interne (état d'étape, temporisation) ou externe (entrée de l'automate). Elle est active soit sur un niveau soit sur un front.

1.5.3 Règles d'évolution.

L'aspect dynamique est défini par les cinq règles d'évolution suivantes :

- ***Situation initiale*** *:* la situation initiale correspond aux étapes actives au début du fonctionnement. C'est donc le comportement au repos.
- ***Franchissement d'une transition :*** une transition est dite validée lorsque toutes les étapes amont de cette transition sont actives. Le franchissement d'une transition est effectif lorsque la transition est validée et lorsque la réceptivité associée est vraie.
- ***Activation des étapes*** : le franchissement d'une transition entraîne immédiatement l'activation des étapes aval de cette transition et la désactivation de ses étapes amont.
- ***Evolutions simultanées*** : plusieurs transitions simultanément franchissables sont effectivement franchies simultanément.
- ***Activation et désactivation simultanée d'une étape*** : si, au cours du fonctionnement, une étape est simultanément activée et désactivée, alors elle reste active.

1.5.4 Extensions du Grafcet.

L'utilisation du Grafcet a mis en évidence certains besoins qui ont conduit à l'introduction de nouveaux concepts, normalisés tels que la *macro-étape*, ou les notions de *hiérarchie* et de *forçage.*

***Hiérarchie et forçage* :** La spécification de la commande d'un automatisme, notamment dans le cas des modes de marche et d'arrêt, conduit souvent à des graphes volumineux et peu lisibles. Ces problèmes ont conduit à la proposition des concepts de hiérarchie et de forçage, qui permettent d'organiser la commande. Les niveaux hiérarchiquement les plus élevés réalisent les fonctions décisionnelles alors que les niveaux les plus bas se chargent des fonctions exécutives.

Un Grafcet de niveau "n" agit sur un Grafcet de niveau "n-1" par l'intermédiaire d'un type particulier d'étape spécifiant le forçage. Le forçage est caractérisé par les propriétés suivantes:

- Il peut être temporisé, fugitif, impulsionnel.
- Il bloque, pendant la durée de son émission, le Grafcet de fonctionnement normal.
- Il contraint, à la fin de son émission, le graphe de fonctionnement normal à reprendre son évolution depuis le dernier état forcé.

Le Grafcet est largement diffusé dans le monde industriel. Il est d'ailleurs implanté sur de nombreux automates programmables et constitue un outil aisé de spécification de la commande.

1.5.5 Validation du Grafcet

Dans de nombreuses méthodes de spécification ou de conception utilisant le Grafcet, il est recommandé d'utiliser quelques technique pour la validation des modèles. La technique la plus importante est la validation par génération du graphe des situations accessibles

Les approches proposées dans [BLA 79] [ROU 94] pour la validation de Grafcet s'articulent autour du graphe des situations accessibles. Le but de ces approches :

- Tout d'abord calculer tout les comportements possibles du Grafcet, ce que traduit bien le graphe des situations accessibles.
- Ensuite valider le Grafcet par analyse des comportements obtenus.

La phase de calcul des comportements du Grafcet consiste à dresser la liste exhaustive des situations accessibles et des possibilités d'évolutions entre ces situations.

La phase de validation repose sur la mise en place d'un langage formel pour l'expression des propriétés à vérifier, et l'utilisation d'un analyseur de systèmes de transitions, comme proposé dans [LEP 94].

Ces approches sont donc basées sur l'exploitation du graphe des situations accessibles, ce qui permet de définir la notion de situation atteinte et de situation accessible, et de mettre en évidence que le graphe des situations accessibles contient toutes les propriétés du Grafcet. Si on fait abstraction de la complexité combinatoire inévitable à ce type d'approche, la meilleure approche pour valider un Grafcet est la génération de son graphe de situations accessibles.

1.6 Conclusion

Ce premier chapitre a présenté quelques notions essentielles sur les SED. Un SED peut être modélisé par un langage, et si ce dernier est régulier, il est possible de construire un automate dont le langage sera précisément celui qui décrit le comportement du SED. Ce SED est également caractérisé par l'accomplissement de certaines tâches, il est donc aussi décrit par le langage marqué de l'automate.

Les automates peuvent être utilisés dans la modélisation des SED. Lorsque le SED est décrit par plusieurs automates élémentaires L'opération de composition (synchrone ou asynchrone) est alors nécessaire pour modéliser le comportement global du système.

Les modèles SED dans le cas de systèmes réels conduisent souvent à des modèles de taille excessive du fait de l’explosion combinatoire du nombre d’états. Les réseaux de Petri (RdP) ont, par rapport aux automates, l’avantage d’être un modèle bénéficiant de structure plus riche, s’adaptant parfaitement à la description des SED.

En ce qui concerne les domaines d’application le Grafcet est bien adapté à la programmation d’un automate directement en interaction avec les capteurs es les actionneurs (commande locale) le Grafcet est plus proche de la mis en œuvre et on spécifie les réceptivités et les actions en même temps que l’on construit le Grafcet.

Chapitre 2

Méthodes de synthèse de commande de systèmes à événements discrets

CHAPITRE 2

Méthodes de synthèse de commande de systèmes à événements discrets

Dans ce chapitre nous allons présenter les principales méthodes existantes dans la littérature sur la commande des systèmes à événements discrets. La théorie de la supervision des SED proposée par Ramadge & Wonham sera d'abord présentée en détail, elle est l'origine de la commande par supervision. Puis deux méthodes de synthèse de la supervision seront décrites. La première combine l'approche de Ramadge & Wonham et la théorie des régions, et la deuxième exploite les invariants de marquage de réseaux de Petri pour synthétiser le système de commande.

Chapitre 2

Méthodes de synthèse de commande de systèmes à événement discrets.

Le problème de la synthèse d'un superviseur pour un SED a été étudié à la fois par la communauté automaticienne et par la communauté informaticienne [ASA 98]. L'équipe du professeur Wonham à Toronto est à l'origine de la théorie de la commande par supervision [RAM 83][[RAM 87a][RAM 87b].

2.1 Synthèse de la commande Ramadge & Wonham

Dans cette approche, le temps n'intervient pas. L'étude se place à un *niveau qualitatif*. Des modèles logiques seront utilisés pour décrire le fonctionnement d'un procédé. Un procédé est considéré comme un SED qui évolue spontanément en générant des événements. Le schéma de supervision est présenté dans la figure 2.1.

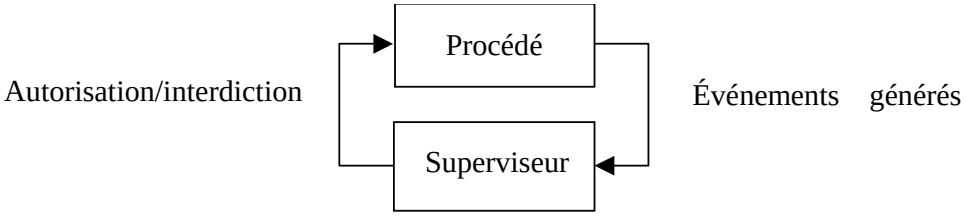


Figure 2.1. Schéma de supervision.

Dans ce schéma, un procédé est couplé à un *superviseur*. Les entrées du superviseur sont les sorties du procédé, et vice versa. Le superviseur est un SED qui évolue conformément aux événements qui sont générés dans le procédé. Le superviseur peut modifier le fonctionnement du procédé en autorisant ou en interdisant l'occurrence d'événements dans le procédé. Etant donné un procédé et un ensemble de spécifications logiques de fonctionnement, l'objectif de la théorie RW est de synthétiser un superviseur tel que le fonctionnement du procédé couplé au superviseur respecte les spécifications. De plus, on souhaite que le fonctionnement ainsi obtenu soit le plus permissif possible. La synthèse d'un tel superviseur est basée sur le concept de *contrôlabilité*.

Dans la figure 2.1, un superviseur unique est couplé au procédé. La supervision sera qualifiée de *centralisée*. En revanche, lorsque plusieurs superviseurs sont couplés à un même procédé, on parlera alors de *supervision modulaire*.

2.1.1 Concept de supervision

Dans cette théorie, un SED est modélisé par un automate et la supervision consiste à interdire le franchissement de certaines transitions et à autoriser le franchissement d'autres transitions, par l'intermédiaire de l'action du superviseur. Le SED évolue avec un maximum de liberté tout en respectant certaines contraintes fixées par les spécifications. L'objectif est de construire ce superviseur d'une manière telle que l'ensemble des événements autorisés dépende de l'évolution passée du procédé. La notion de comportement en boucle fermée s'applique à une telle approche.

Le procédé, modélisé par un automate, est couplé à un superviseur, représenté lui aussi par un automate. Le superviseur évolue conformément aux événements issus du procédé. Le superviseur intervient sur l'évolution du procédé par l'intermédiaire de lois de contrôle qui définissent l'ensemble des événements autorisés à partir de l'état dans lequel se trouve le procédé. Le couplage procédé / superviseur doit assurer le respect de spécifications données par le cahier des charges.

Le procédé est modélisé par un automate $P = (Q, \Sigma, \delta, q_0)$ où Q est un ensemble fini d'états; Σ est l'ensemble des événement; $\delta : Q \times \Sigma \to Q$ est la fonction de transition d'états; q_0 est l'état initial.

On appelle *langage de procédé* et on note $L(P)$ le langage associé au procédé P . Ce langage représente l'ensemble des séquences d'événements qui peuvent être générées par P. Ce langage est nécessairement préfixe-clos du fait que le langage accepté par l'automate du procédé est régulier. De même, tous les langages qui seront considérés dans la suite seront préfixe-clos.

2.1.2 Définition d'un superviseur

Formellement, un superviseur S peut être défini par une machine de Moore.

Définition 2-1

Le superviseur S peut être défini par le 6-uplet $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ où V est un ensemble fini d'états; Σ est l'alphabet d'entrée; $\xi : V \times \Sigma \to V$ est la fonction de transition

d'états; v_0 est l'état initial; $2^{\Sigma c}$ est l'alphabet de sortie; et $\theta : V \rightarrow 2^{\Sigma c}$ est la fonction d'affectation de sortie.

Le superviseur *S* peut être perçu comme une machine à états déterministe qui évolue conformément à une modification de son entrée (sur l'occurrence d'un événement de Σ généré par le procédé) et qui change d'état selon ξ. Pour chaque état *v*, le superviseur *S* fournit en sortie une liste d'événements interdits $\Phi = \theta$ (*v*). Rappelons que seuls les événements contrôlables peuvent être interdits par la supervision. Ainsi, chaque sortie de *S* est un élément de $2^{\Sigma c}$, où $2^{\Sigma c}$ est l'ensemble de tous les sous-ensembles de Σ_c.

Un superviseur est considéré comme une fonction :

$$S : L(P) \rightarrow \Gamma \text{ avec } \Gamma = \{\gamma \in 2^{\Sigma} \mid \Sigma_u \subseteq \gamma\}$$

$$\forall\ \omega \in L(P)\ ,\ S(\omega) = \gamma \in \Gamma$$

Où γ représente l'ensemble des événements autorisés par le superviseur depuis un état *q* de l'automate du procédé. Γ représente l'ensemble des lois de commande $\Gamma \subseteq 2^{\Sigma.}$

Donc le superviseur observe ω et il génère la commande γ, l'ensemble des événements possibles après l'observation de ω est $S(\omega) \cap \Sigma(\delta(q_0, \omega))$, si $q = \delta(q_0, \omega)$ est l'état où se trouve le procédé après la séquence ω. On note par $\Sigma_p(q)$ l'ensemble des événements qui peuvent être générés par le procédé depuis l'état *q*. Donc on peut écrire $\Sigma_p(\delta(q_0, \omega))$ pour désigner l'ensemble des événement possibles après ω dans le procédé. Autrement dit le procédé ne peut pas exécuter un événement qui appartient à l'ensemble des événements actifs actuels si cet événement n'appartient pas également à $S(\omega)$.

On appelle *langage de supervision* et on note *L*(*S*) le langage associé au superviseur *S*. On définit le langage *L*(*S*) comme l'ensemble des séquences d'événements qui ne sont pas interdites par *S*. De façon duale, *L*(*S*) représente alors l'ensemble des séquences d'événements autorisées par *S*. Formellement, *L*(*S*) peut être défini de la façon suivante :

Définition 2-2

Le langage *L*(*S*) associé à l'automate $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ est défini de façon récursive par le plus petit langage sur Σ^* satisfaisant :

- $\varepsilon \in L(S)$
- si $\omega \in L(S)$ alors $\omega\sigma \in L(S)$ pour $\sigma \in \Sigma$ si $\sigma \notin \theta(\xi(v_0, \omega))$.

Considérons une séquence d'événements ω non interdite par S, c'est-à-dire, $\omega \in L(S)$. Si la séquence d'entrée ω a conduit le superviseur dans l'état $\xi(v_0, \omega)$ et si σ est un événement de $\sum$ alors, la séquence $\omega\sigma$ appartient au langage $L(S)$ si l'événement σ n'est pas interdit par le superviseur depuis l'état $\xi(v_0, \omega)$, c'est-à-dire, si $\sigma \notin \theta(\xi(v_0, \omega))$.

Exemple d'application :

Considérons un système manufacturier composé de deux machines identiques : M_1 et M_2, et un stock entre de deux machines. Conformément à la figure 2.2, les deux machines travaillent de façon indépendante, puisent des pièces brutes en amont et rejettent des pièces usinées en aval. Nous allons utiliser cet exemple tout au long de ce chapitre.

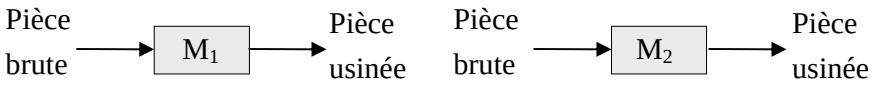


Figure 2.2. Exemple : un système manufacturier.

Modélisation du procédé : le procédé est supposé évoluer de façon spontanée en générant des événements (*générateur d'événements*). Son fonctionnement peut être décrit par un ensemble de séquences d'événements qui constitue un langage formel sur l'alphabet des événements.

Chaque machine de ce procédé peut être modélisée par un accepteur (automate sans sorties). Le graphe de transition d'états de l'accepteur des machines M_1 et M_2 est représenté dans la figure 2.3.

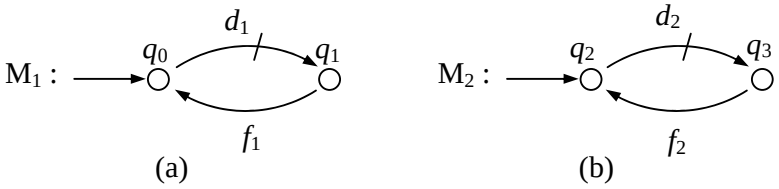


Figure 2.3. Modèles des machines.

Considérons la machine M_1 (figure 2.3.a). Dans son état initial (état q_0), la machine est à l'arrêt. L'événement d_1 modélise le début du cycle de la machine, l'occurrence de cet événement conduit la machine dans l'état de marche (état q_1). Dans notre exemple, nous supposons que d_1 est simultané avec la prise d'une pièce en amont. De même, la fin de cycle (événement f_1) est simultanée avec le dépôt d'une pièce en aval.

Notons respectivement Σ_1 et Σ_2 les alphabets des machines M_1 et M_2. Nous avons : $\Sigma_1=\{d_1, f_1\}$ et $\Sigma_2=\{d_2, f_2\}$. Le fonctionnement du système manufacturier est alors défini sur un alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$. Nous avons pris la convention de représenter par un arc barré, toute transition associée à un événement contrôlable. Ainsi, nous supposons que $\Sigma_c = \{d_1, d_2\}$ et $\Sigma_u = \{f_1, f_2\}$.

Un modèle accepteur de notre système manufacturier peut être obtenu en effectuant le composé asynchrone des modèles M_1 et M_2. le modèle P défini par $P = M_1 \parallel M_2$ est représenté par son graphe de transition d'états dans la figure 2.4.

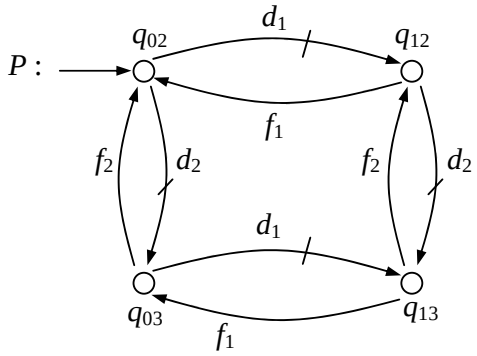


Figure 2.4. modèle accepteur du système manufacturier.

Dans l'accepteur P, un état est un couple (q_i, q_j) où q_i est un état de M_1 et q_j est un état de M_2. Cet état (q_i, q_j) est noté q_{ij} dans la figure 2.4.

Spécification : Le fonctionnement de notre système manufacturier doit respecter la présence d'un stock de capacité limitée à 1, situé entre les deux machines. Nous supposons donc à présent que les machines travaillent en série.

Figure 2.5. le système manufacturier sous la contrainte de stock

Conformément à la figure 2.5, une pièce doit visiter M_1 puis M_2. La présence du stock entre M_1 et M_2 impose que : (1) la machine M_2 ne peut commencer à travailler que si elle peut prendre une pièce dans le stock, c'est-à-dire, si le stock est plein, et (2) la machine M_1 ne peut déposer une pièce dans le stock que si celui-ci est vide. Le stock est supposé vide dans son état initial.

L'accepteur de la spécification S_{Spec} de la figure 2.6 permet de garantir le respect de cette

spécification de fonctionnement. Dans cet accepteur les états v_0 et v_1 correspondent respectivement aux états : "stock vide" et "stock plein".

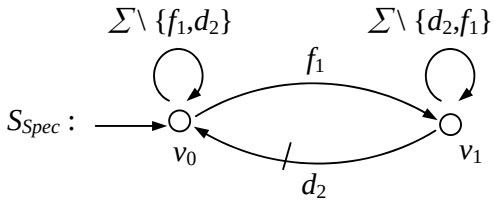


Figure 2.6. modèle accepteur de la spécification.

Lorsque le stock est vide, l'occurrence de l'événement contrôlable d_2 est interdit (début du cycle de M_2). Sur l'occurrence de l'événement f_1 (fin du cycle de M_1 et dépôt d'une pièce dans le stock), l'accepteur S_{Spec} change d'état et passe dans l'état v_1. Dans cet état, l'occurrence de l'événement contrôlable d_1 est interdit (début du cycle de M_1).

Fonctionnement en boucle fermée

On appelle *fonctionnement en boucle fermée*, le fonctionnement du procédé couplé à son superviseur. Conformément à la figure 1.1, un événement peut être généré par le procédé supervisé si, et seulement si, il peut être généré par le procédé P en isolation et s'il est autorisé (non interdit) par le superviseur S. Par extension, une séquence d'événements ω est possible dans le fonctionnement en boucle fermée, si elle est possible dans le procédé en isolation ($\omega \in L(P)$), et si elle est autorisée par le superviseur ($\omega \in L(S)$). Si on note S/P la machine constituée du procédé P couplé au superviseur S, le langage $L(S/P)$ représente alors le fonctionnement en boucle fermée du système. Le langage $L(S/P)$ est simplement défini par : $L(S/P) = L(P) \cap L(S)$. Le modèle accepteur reconnaissant $L(S/P)$ est obtenu en effectuant le composé synchrone de P et de S.

Définition 2-3

le langage $L(S/P)$ généré par le procédé supervisé en boucle fermée est défini par :

- $\varepsilon \in L(S/P)$
- $[\omega\sigma \in L(S/P)] \Leftrightarrow [(\omega \in L(S/P)) \wedge (\sigma \in S(\omega)) \wedge (\omega\sigma \in L(P))]$.

Un mot $\omega\sigma$ peut être généré par le procédé supervisé seulement si le mot ω a été généré par le procédé supervisé, si l'événement σ est autorisé par le superviseur et le mot $\omega\sigma$ est accepté par le procédé non supervisé. Le mot vide ε est compris dans le langage *L(S/P)*.

Ainsi, *L(S/P)* est inclus dans le langage *L(P)* et il est préfixe-clôt. Par conséquent, un superviseur ne peut que restreindre le comportement d'un procédé.

2.1.3 Concept de contrôlabilité et condition d'existence d'un superviseur :

2.1.3.1 Contrôlabilité

Ramadge et Wonham ont introduit la notion de contrôlabilité pour des SED afin de caractériser les langages supervisés d'un procédé [RAM 87b].

Etant donnés un procédé P et une spécification de fonctionnement S_{spec}, on souhaite synthétiser un superviseur S de façon à ce que le système en boucle fermée S/P, respecte la spécification.

C'est-à-dire qu'on doit chercher le langage $L(P) \cap L(S_{spec})$. Ce langage appelé *fonctionnement désiré* correspond à l'ensemble des séquences qui peuvent être générées par le procédé et qui sont tolérées par la spécification, ce langage est noté $L_{D.}$

Il n'est pas toujours possible (prise en compte d'événements incontrôlables Σ_u) de restreindre, par la supervision, le fonctionnement d'un procédé à n'importe quel sous-langage de ce fonctionnement. L'existence d'un superviseur S tel que $L(S/P) = L_D$ réside dans le concept de contrôlabilité.

Définition 2-4

Un langage K est dit contrôlable par rapport à un langage *L(P)* si

$$\overline{K}\,\Sigma_u \cap L(P) \subseteq \overline{K}$$

Dans cette définition K représente le langage de spécification et $L(P)$ le langage de procédé. Ainsi, le langage de spécification K est contrôlable par rapport à un langage $L(P)$ si pour toute chaîne ω de $\overline{K}$ et pour tout événement τ incontrôlable de Σ_u , la chaîne $\omega\tau$ appartient à $L(P)$, implique qu'elle appartient aussi à $\overline{K}$.

Dans l'exemple de notre système manufacturier, un modèle de fonctionnement en boucle fermée est obtenu en effectuant le composé synchrone des modèles de P (figure2.4) et de S_{Spec} (figure 2.6). L'accepteur $D = P \,||s\, S_{Spec}$ est représenté dans la figure 2.7. Dans ce modèle, chaque état est un couple (q, v) où q est un état du procédé P (figure 2.4) et v est un état du S_{Spec} (figure 2.6). Notons que les modèles du procédé et du superviseur sont, par définition, construits sur le même alphabet. Ainsi, il existe une transition associée à un événement σ à partir d'un état (q, v) de D s'il existe une transition associée à σ à partir de q dans P et à partir de v dans S_{Spec}.

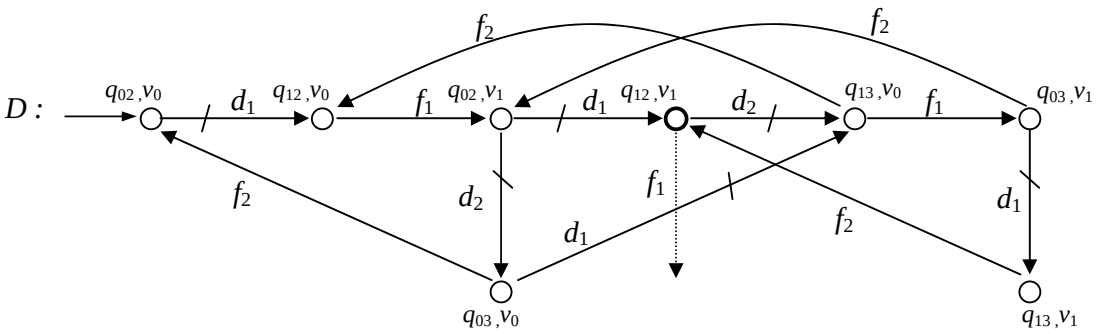


Figure 2.7. Modèle du fonctionnement en boucle fermée.

Considérons la séquence d'événements $d_1\ f_1\ d_1$ (la machine M_1 a débuté un cycle, terminé son cycle, puis débuté un nouveau cycle). Après cette séquence, le procédé P est dans l'état q_{12} (figure 2.5). L'état q_{12} représente l'état du système pour lequel la machine M_1 est en marche et la machine M_2 est à l'arrêt. Depuis cet état, il existe une transition de sortie associée à l'événement f_1. Cela signifie que f_1 peut être spontanément généré par le procédé. On dira alors que la séquence d'événements $d_1\ f_1\ d_1\ f_1$ est "possible" dans le procédé, c'est-à-dire $d_1\ f_1\ d_1\ f_1 \in L(P)$.

La séquence d'événements $d_1\ f_1\ d_1$ conduit la spécification de la figure 2.6 dans l'état v_1 qui correspond à l'état "stock plein". Depuis cet état, il n'existe pas de transition de sortie associée à l'événement f_1. Cela signifie que l'événement f_1 n'est pas toléré par la spécification après la séquence $d_1\ f_1\ d_1$ (le dépôt d'une pièce dans le stock plein n'est pas toléré). Ainsi, la séquence $d_1\ f_1\ d_1\ f_1$ n'est pas un élément du langage de spécification ($d_1 f_1 d_1 f_1 \notin L(S_{spec})$). Cependant, l'événement f_1 qui est incontrôlable ne peut pas être interdit après la séquence $d_1\ f_1\ d_1$. Le langage de spécification $L(S_{spec})$ sera alors qualifié de non *contrôlable* par rapport au langage du procédé $L(P)$.

2.1.3.2 Conditions d'existence d'un superviseur

Soit $L(P)$ le langage du procédé et L_D un fonctionnement désiré inclus dans $L(P)$. Quelle est la condition pour qu'un superviseur S existe tel que $L(S/P) = L_D$?.

Pour un langage L_D préfixe-clos, non vide et inclus dans le langage $L(P)$ du procédé, il existe un superviseur S tel que $L(S/P) = L_D$ si et seulement si L_D est contrôlable par rapport à $L(P)$ [RAM 89].

Si L_D n'est pas contrôlable par rapport au langage du procédé, il n'existe pas de superviseur S tel e $L(S/P)= L_D$. Dans ce cas, il est nécessaire d'être plus restrictif et la synthèse d'un superviseur requiert alors la détermination du *langage suprême contrôlable.*

2.1.3.3 Langage suprême contrôlable

Pour un langage donné L_D non contrôlable, le plus grand sous ensemble de L_D qui soit contrôlable sera appelé *langage suprême contrôlable* et sera noté $SupC(L_D)$.

$SupC(L_D)$ est inclus dans L_D. Or, puisque $SupC(L_D)$ est le plus grand élément contrôlable, alors le fonctionnement en boucle fermée obtenu est *le plus large possible* : il est dit *permissif* maximal. En ce sens, le fonctionnement est qualifié d'*optimal.*

2.1.4 Synthèse du superviseur

A partir des modèles automates d'un procédé P et d'une spécification de fonctionnement S_{spec}, deux algorithmes permettent de vérifier la contrôlabilité du langage de spécification $L(S_{spec})$ [WON 87] [KUM 91]. Si les spécifications sont contrôlables, l'automate P/S_{spec} décrivant le procédé supervisé est la composition synchrone des automates P et $S_{spec.}$, Dans le cas où le langage $L(S_{spec})$ n'est pas contrôlable, il est possible de synthétiser un modèle automate du langage suprême contrôlable du fonctionnement désiré $SupC(L_D)$ décrivant le plus grand ensemble possible des séquences contrôlables du procédé respectant les spécifications. Cet ensemble est appelé le maximum permissif. A partir de $SupC(L_D)$, l'obtention d'un superviseur S' tel que $L(S'/P) = SupC(L_D)$, est systématique [WON 87].

Intuitivement, la détermination de $SupC(L_D)$, revient à inhiber par la supervision les événements contrôlables qui précèdent, de "plus près", les états sources d'incontrôlabilité. Une telle inhibition d'événements contrôlables évite au procédé supervisé de se trouver dans un état à partir duquel l'occurrence d'un événement incontrôlable serait inévitable. La supervision permet ainsi d'assurer le fonctionnement spécifié de façon optimale.

Il existe dans la littérature plusieurs méthodes pour le calcul d'un langage suprême contrôlable [WON 87] [BRA 90] [KUM 91].

L'algorithme proposé dans [WON 87] est basé sur un calcul itératif.

L'algorithme proposé dans [KUM 91] est non itératif. Il est basé sur une détermination d'états défendus dans le modèle automate du fonctionnement désiré. Cet algorithme est présenté dans la section suivante.

Algorithme de Kumar

a) Présentation intuitive

A partir des modèles accepteurs P d'un procédé et S_{spec} d'une spécification de fonctionnement, l'algorithme de Kumar permet de vérifier la contrôlabilité du langage de spécification $L(S_{spec})$. De plus, dans le cas où le langage $L(S_{spec})$ n'est pas contrôlable, cet algorithme permet de synthétiser un modèle accepteur du langage suprême contrôlable du fonctionnement désiré, c'est-à-dire, $SupC(L_D)$. Ceci est illustré dans la figure 2.8.

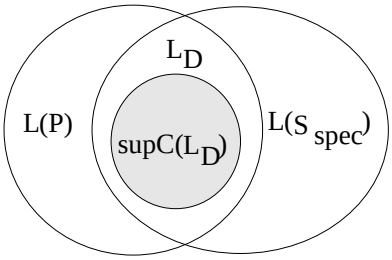


Figure 2.8. Langage de spécification, fonctionnement désiré et langage suprême contrôlable du fonctionnement désiré.

A partir du modèle accepteur de $SupC(L_D)$, l'obtention d'un superviseur S' tel que $L(S'/P)$ soit égal à $SupC(L_D)$ est alors systématique. Un tel superviseur S' permet le respect des contraintes de fonctionnement définies par S_{spec} de la façon la plus permissive possible.

L'algorithme de Kumar est simplement basé sur la détermination des états défendus du modèle accepteur du fonctionnement désiré.

b) Présentation formelle

Algorithme de Kumar. Soit $P = (Q, \Sigma, \delta, q_0)$ et $S_{spec} = (V, \Sigma, \xi, v_0)$ les modèles accepteurs du procédé et de la spécification. L'algorithme est basé sur les 4 pas suivants :

Pas 1. On construit le composé synchrone D de P et de S_{spec}, c'est-à-dire, $D = P \,||s\, S_{spec}$. Le langage $L(D)$ sera noté L_D.

Pas 2. On détermine l'ensemble des *états défendus*, où l'on appelle : *état défendu*, tout état (q, v) de D tel qu'il existe un événement incontrôlable $\sigma \in \Sigma_u$ où $\delta(q, \sigma)$ est définie dans P et $\xi(v, \sigma)$ n'est pas définie dans S_{spec}.

Pas 3. On détermine l'ensemble des *états faiblement défendus*, où l'on appelle : *état faiblement défendu*, tout état (q, v) de D qui n'est pas un état défendu et tel qu'il existe une séquence événements incontrôlables $s_u \in \Sigma_u^*$ qui conduit à un état défendu (q', v') de D.

Pas 4. On supprime de D l'ensemble des états défendus ainsi que l'ensemble des états faiblement défendus (ainsi que les transitions associées à ces états).

On supprime de D l'ensemble des états non accessibles, c'est-à-dire, tout état (q, v) tel qu'il n'existe pas de chemin permettant de rejoindre (q, v) depuis l'état initial. On appelle D' le modèle accepteur ainsi obtenu.

Le pas 2 signifie qu'un état (q, v) de D est un état défendu si on peut trouver un événement incontrôlable σ, tel que σ peut être généré par le procédé dans l'état q et σ n'est pas toléré dans la spécification dans l'état v. Dans notre exemple nous avons vu que l'état (q_{12},v_1) est un état défendu.

Selon le pas 3, on appelle état faiblement défendu tout état de D à partir duquel on peut rejoindre un état défendu par une séquence d'événements incontrôlables. Dans notre exemple, l'événement f_2 est incontrôlable, on trouve alors que l'état (q_{13},v_1) est un état faiblement défendu.

Le modèle accepteur du fonctionnement désiré de la figure 2.7 est à nouveau représenté dans la figure 2.9. Dans ce modèle, les états défendus et faiblement défendu sont symbolisés par des cercles barrés.

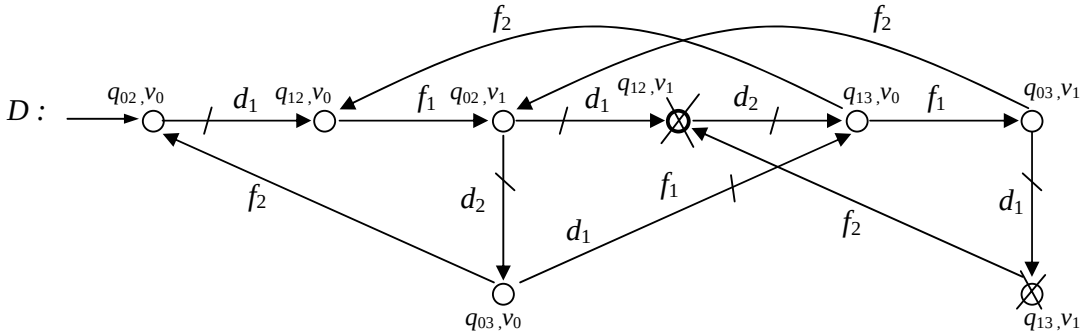


Figure 2.9. Etats défendu du modèle accepteur de fonctionnment désiré

Pour obtenir le modèle accepteur *D*', on supprime l'états défendu et l'état faiblement défendu de *D* (pas 4). Nous obtenons finalement l'accepteur *D*' de la figure 2.10.

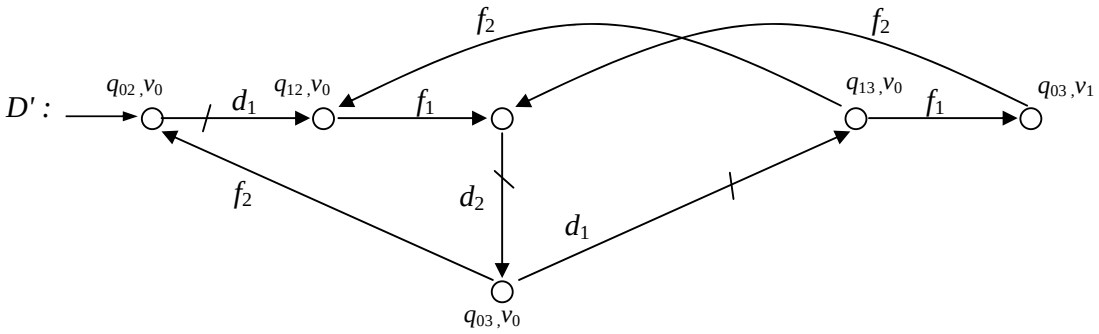


Figure 2.10. Superviseur pour la contraite de stock obtenu à partir de la figure 2.9.

Remarques et conclusion sur l'approche RW :

Si le modèle du procédé comporte n états et si le modèle de la spécification comporte m états, alors l'algorithme de Kumar permet de synthétiser un superviseur qui comporte (au plus) n.m états. Ainsi, la taille du superviseur est souvent bien plus grande encore que celle du procédé. Il résulte que dans bien des cas, l'explosion combinatoire due à l'utilisation de modèles automates rend impossible la synthèse de superviseurs.

Il apparaît clairement que maîtriser la taille des superviseurs est un objectif crucial pour l'applicabilité de l'approche RW. Un grand nombre des extensions qui ont été faites à partir de la théorie de R&W, visent à répondre à cet objectif.

La théorie de R&W constitue actuellement une activité importante au sein de plusieurs groupes de recherche sur le plan international, cette théorie possède diverses extensions. Les travaux basés sur cette théorie sont très nombreux.

L'approche classique utilise des automates à états et un point de vue centralisé. Les modèles obtenus sont de grande taille, ce qui pose un problème de calculabilité et de lisibilité. Différents travaux proposent des approches permettant de réduire cette taille par la remise en cause du point de vue centralisé : point de vue hiérarchique [ZHO 90], modulaire sous observation partielle [RAM 86] ou décentralisée [LIN 90]. Une fois la commande obtenue, une interprétation est nécessaire. Là aussi, différents travaux proposent des interprétations sous forme de schémas Ladder [LED 96][FAB 98] ou de Grafcets [CHA 96].

La théorie RW a été aussi étendue afin de tenir compte des contraintes temporelles [GAU 95][BRA 94][SAV 02] et des aspects de la concurrence [WAN 93]. Parmi les autres extensions de la théorie RW, nous citons l'extension aux systèmes hybrides [ANT 93,97], et l'extension aux SED à structure vectorielle [LI 94].

Un outil informatique (logiciel TCT) à été développé dans l'équipe du Professeur Wonham « System Control Group » à l'université de Toronto. En exploitant les concepts de la théorie RW, cet outil permet la synthèse de la commande par supervision des SED.

Superviseur ou Contrôleur

Dans le modèle des systèmes à événements discrets contrôlés proposé par Ramadge et Wonham, seul le système (procédé) modélisé peut émettre des événements alors que le superviseur a seulement un rôle d'inhibition de certains événements par un mécanisme qui est d'ailleurs absent du modèle. Entre autres conséquences, le système est seul maître du cadencement des événements. Dans de nombreuses applications, ce schéma n'est pas réaliste. Le système contrôlé émet des événements en réaction à des commandes qui lui sont envoyées manuellement ou par un contrôleur.

Pour résoudre le problème de ce modèle Balemi, dans [BAL 92] a proposé une approche *entrées/sorties* pour le contrôle des systèmes à événements discrets. Le sous alphabet Σ_c, s'interprète alors comme l'alphabet des commandes qui sont des *entrées* du procédé, les événements incontrôlables Σ_u modélisant les sorties du procédé. Le système réalise alors une espèce de fonction de transfert de l'action des entrées vers leur effet sur les sorties.

Le modèle du superviseur devient alors tout à fait symétrique: il reçoit en entrée les sorties Σ_u du système et produit en sortie les commandes Σ_c du système. Le comportement

du superviseur (que nous appellerons contrôleur), est encore modélisé par un langage sur $\Sigma = \Sigma_c \cup \Sigma_u$. Le comportement du procédé à contrôler est toujours décrit par le langage *L(P)* et la notion de langage marqué, tant pour le système que pour le contrôleur reste valable.

Dans ce nouveau schéma, le cadencement des événements est partagé entre le système et son contrôleur. Dans le modèle avec superviseur, le système impose au superviseur l'acceptation de tout événement incontrôlable.

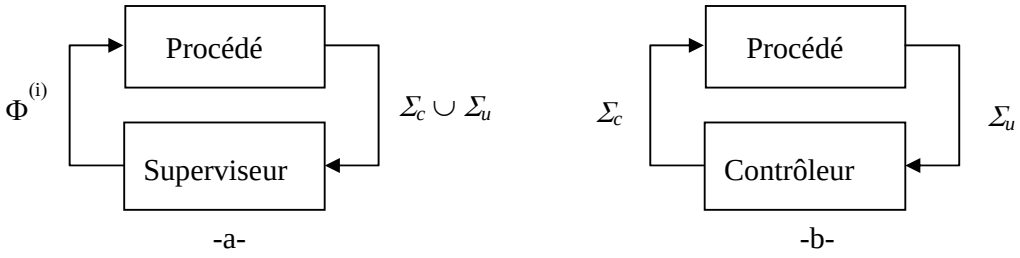


Figure 2.11. a. l'approche originale de Ramadge/Wonham, b. l'approche entré/sortie de Balemi

Superviseur : le superviseur interdit certains d'événements d'être générés dans le procédé.

Contrôleur : le contrôleur produit des événements qui agissent sur le procédé

Le contrôleur est déterministe alors que le superviseur peut être non déterministe.

2.2 Synthèse de la commande basée sur les réseaux de Petri

Dans le travail présenté dans cette thèse nous cherchons à résoudre le problème de synthèse de commande des SED pour des applications réelles, c'est pourquoi nous nous intéressons aux systèmes qui peuvent être modélisé par des outils de modélisation directement implantables dans un automate programmable tels que le Grafcet et puisque le Grafcet est un outil dérivé des RdP, nous allons étudier les méthodes de synthèse de commande basées sur RdP et les adapter au Grafcet. L'avantage pratique de synthèse de commande basée sur le Grafcet est qu'elle permet d'obtenir des modèles directement implémentables dans les automates programmables industriels.

Les RdP sont un outil approprié pour étudier les SED à cause de leur puissance de représentation graphique et de leurs propriétés mathématiques. Ils ont été utilisés largement pour modéliser et simuler beaucoup de types de systèmes. Leur usage dans le contrôle est

plus récent, seules quelques études ont été menées vers cette direction [HOL 90][YAM 91][BOI 93] [KRO 87] [ICH 87].

Lorsque le système rencontré en pratique est simple, la solution aux problèmes de supervision peut être trouvée de façon intuitive. Cependant, devant l'accroissement de la complexité de ces systèmes, il est rapidement devenu trop difficile, voire impossible, de pouvoir spécifier de manière intuitive le superviseur. C'est dans cette optique que diverses théories de supervision des SED sont apparues, visant à automatiser en partie la synthèse de commande.

Comme nous l'avons vu dans l'approche proposée par Ramadge et Wonham le grand nombre d'états à considérer pour représenter le comportement du système ainsi que le manque de structure dans les modèles limite les possibilités de développer des algorithmes de calcul efficaces pour l'analyse et la synthèse.

Pour pallier ces inconvénients, plusieurs méthodes de synthèse de commande basées sur les RdP ont été proposées. Elles permettent d'exploiter la puissance de modélisation de cet outil tout en apportant de nombreux résultats mathématiques qui permettent une analyse qualitative du système.

Les spécifications dans la synthèse de commande basée sur les RdP, sont souvent données comme un ensemble d'états interdits en présence d'événements incontrôlables, ce type de problème a été étudié par plusieurs chercheurs. Dans [BAS 98], il est prouvé la propriété suivante : si l'ensemble des marquages admissibles L est exprimé par un ensemble de contraintes linéaires et si L est contrôlable, alors une solution basée sur un modèle RdP existe et elle est optimale. Un résultat similaire d'existence de superviseur sous forme de places de RdP pour le problème d'états interdits pour les réseaux saufs a été établi dans [GIU 93].

Holloway et Krogh formulent et résolvent le problème d'état interdit pour les SED qui peuvent être modélisés par des graphes d'événements cycliques et saufs [HOL 90] au moyen d'une évaluation en ligne de prédicats sur les marquages

Li et Wonham ont proposé une solution pour les réseaux où les sous-réseaux non contrôlables sont acycliques [LI 94], seulement, le superviseur doit résoudre en ligne à chaque changement d'état du procédé un certain nombre de programmes linéaires en nombres entiers, ce qui rend l'approche difficile à appliquer en temps réel. Des améliorations ont été proposées pour pallier cet inconvénient. Dans [YAM 96], les spécifications d'états interdits sont exprimées sous forme de contraintes généralisées

d'exclusion mutuelles. Les auteurs utilisent les invariants de marquage pour déterminer la matrice d'incidence du superviseur le plus permissif possible. Cette approche sera détaillée dans ce chapitre. Moody et al.[MOO 96] étendent l'utilisation du concept des invariants de marquage aux réseaux avec des transitions non contrôlables et non observables.

Boissel [BOI 93] a utilisé une méthode pour calculer un contrôleur des SED qui sont modélisés par RdP. Bien qu'elle soit admise pour produire un contrôleur optimal et maximal permissif, cette méthode est très peu applicable pour les grands systèmes puisqu'elle implique la construction de l'arbre d'atteignabilité plusieurs fois pendant l'exécution de l'algorithme.

Nous allons présenter ci-dessous les principales approches de synthèse de contrôleurs basées sur les RdP. Nous discuterons des avantages et inconvénients de chacune d'elles, cela nous permettra d'introduire notre travail de recherche.

2.2.1 Quelques approches sur le contrôle des RdP

Le contrôle des RdP qui suit, est celui qui est présenté par E. Holloway et par Bruce H. Krogh dans [KRO 91] et [KRO 93]. Les auteurs ont utilisé le contrôleur de RdP pour contrôler les systèmes qui peuvent être modélisés par des graphes d'événements contrôlés et cycliques qui sont une classe particulière des SED (pas de possibilités de conflit ou de choix). Ils ont montré comment synthétiser un contrôleur maximal permissif en boucle fermée qui garantit qu'aucun des états interdits ne sera atteint. Leur méthode ne nécessite pas une recherche exhaustive de l'espace d'état du système, L'inconvénient est que c'est applicable à une classe limitée de systèmes : les RdP qui sont des graphes d'événements. Ils reprennent l'idée de contrôler les RdP par inhibition d'événements appartenant à un sous-ensemble Σ_c des événements contrôlables. Cet ensemble n'est pas spécifié directement mais est défini en munissant certaines transitions de places de contrôle.

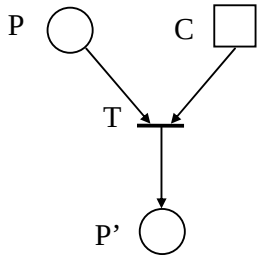


Figure 2.12. Modélisation des places de contrôle dans un RdP

La condition de franchissement de transition contrôlée *T* est alors la présence de jetons dans les places d'entrée de *T* et d'un jeton fictif dans les places de contrôle. L'ensemble des places de contrôle sera noté *C*. On peut interpréter un contrôle comme étant un commutateur binaire qui laisse passer, ou ne laisse pas passer les jetons des ports d'entrée aux ports de sortie.

La théorie de Krogh et Holloway sur le contrôle des SED représentés par des RdP tire avantage de la puissance d'expression des RdP pour modéliser les systèmes. Cette richesse permet de modéliser de façon simple des SED qui n'auraient pu être représentés pour des raisons de complexité. On peut cependant remarquer que les restrictions apportées à la structure des RdP font que ceux-ci ne peuvent par exemple pas modéliser les conflits, de même il ne doit y avoir qu'un seul jeton par cycle dans le RdP. Cela limite considérablement la portée d'une telle approche.

2.2.2 Synthèse de la commande avec la théorie des régions

Ce paragraphe représente une méthode optimale pour la synthèse de superviseurs sous forme de RdP basée sur la théorie des régions [BAD 95]. La notion d'optimalité est liée au nombre d'états atteignables par le système en boucle fermée, c'est-à-dire sous supervision. Cette méthode a été proposée par A. Ghaffari, N. Rezk [GHA 01].

Etant donné le modèle RdP borné du système, l'approche proposée permet de déterminer le RdP correspondant au comportement du système en boucle fermée. Le superviseur recherché doit être le plus permissif possible tout en tenant compte des spécifications, de la contrainte de vivacité et des transitions non contrôlables. Les spécifications considérées ici sont de type états interdits.

Dans ce paragraphe nous abordons le problème d'états interdits avec intégration de la contrainte de vivacité du procédé supervisé. Pour le modèle RdP, les spécifications du système sont données sous la forme d'une liste de marquages interdits. L'objectif est de déterminer un ensemble de places de supervision qui, une fois ajoutées au modèle du procédé, interdiront l'accès à ces états.

2.2.2.1 Méthodologie de synthèse de la commande

Etant donné un RdP borné quelconque et un ensemble de marquages interdits, on cherche à déterminer les places qu'il faudrait ajouter au réseau pour restreindre son graphe d'atteignabilité au sous-ensemble de marquages admissibles. Ces places de contrôle constituent la commande à imposer au système pour respecter les contraintes données.

La méthode opère en deux phases. Une première phase, basée sur le graphe de marquage, permet de déterminer l'ensemble des marquages admissibles. Puis, en utilisant la théorie des régions, des places de contrôle sont construites pour assurer le comportement désiré.

Soit M_0 le marquage initial du modèle RdP à superviser, et soit R_c le graphe d'atteignabilité du modèle RdP du système commandé, pour déterminer l'ensemble des marquages admissibles on suit les étapes suivantes :

Etape 1 : Détermination de l'ensemble des marquages interdits.

Etape 2 : Génération du graphe des marquages partiel.

Etape 3 : Détermination de l'ensemble des marquages dangereux.

Etape 4 : Recherche de l'ensemble des marquages admissibles.

Etape 5 : Recherche du comportement admissible du procédé R_c.

La commande par supervision doit restreindre le comportement du système commandé à R_c en inhibant toute transition contrôlable qui mène à un marquage bloquant ou interdit.

Le graphe d'atteignabilité obtenu correspond au comportement vivant maximal permissif du système commandé.

Réciproquement, les contraintes d'états interdits et de vivacité sont satisfaites par tous les marquages de R_c, et aucun marquage ne mène de façon *incontrôlable* à l'extérieur de R_c. On en conclut que le graphe R_c obtenu constitue le comportement admissible vivant maximal.

2.2.2.2 Synthèse de la commande avec la théorie des régions

Dans cette section, on explique comment, à partir du graphe des marquages admissibles maximal et en utilisant la théorie des régions, on peut construire les places de contrôle à ajouter au modèle initial pour réaliser le comportement demandé.

Pour le problème de synthèse de la commande, basée sur les RdP, les modèles RdP du système à commander et le graphe R_c (le comportement admissible maximal du modèle commandé obtenu à la section précédente) sont donnés. La théorie des régions est utilisée pour déterminer les places de contrôle à ajouter.

Considérons une place de contrôle P_c. Pour le RdP du Système à commander augmenté de P_c, chaque marquage du graphe R_c doit rester atteignable. Ceci implique que P_c doit vérifier *les conditions d'atteignabilité*, i.e. :

$$M(P_c) = M_0(P_c) + W(P_c, .)\ \vec{\Gamma}_M \geq 0 \qquad \forall \quad M \in R_c \qquad (1)$$

Où $\vec{\Gamma}_M$ est un chemin non orienté de R_c joignant M_0 et M.

D'autres part, P_c doit satisfaire les équations de cycle correspondant aux cycles de R_c, donc l'application de l'équation (1) le long d'un cycle non orienté quelconque γ de P conduit à la relation suivante :

$$\sum_{T_i \in T} W(P_c, T_i).\ \vec{\gamma}[T_i] = 0 \qquad \forall\ \gamma \in S_c \qquad (2)$$

où $\vec{\gamma}\ [T_i]$ désigne la somme algébrique de toutes les occurrences de T_i dans γ et S_c est l'ensemble des cycles de base du graphe R_c. $\vec{\gamma}$ est le vecteur d'occurrences de γ. L'équation (2) est appelée *l'équation de cycle*

Quant aux événements à séparer, pour obtenir R_c à partir du graphe de marquage initial, seules les transitions menant d'un marquage admissible à un marquage non admissible doivent être inhibées.

Par conséquent, chaque nouvelle place P_c doit interdire une transition contrôlable qui mène à un état interdit. Cette interdiction se fait par la solution de l'équation suivante :

$$M_0(P_c) + W(P_c, .)\ \vec{\Gamma}_M + W(P_c, T_i) < 0 \qquad (3)$$

Ainsi, pour chaque transition contrôlable qui mène à un état interdit, on résout le système composé des équations de relations (1), (2) et (3) pour déterminer une nouvelle place de contrôle P_c.

Le problème de commande par supervision peut être résolu de façon optimale par l'ajout d'un ensemble de places de contrôle au modèle du système en question si et seulement si il existe une solution $W(P_c\ ,\ .\ )$, $M_0(P_c)$ vérifiant les équations précédentes pour chaque transition contrôlable qui mène à un état interdit.

Exemple : Reprenons l'exemple d'application donné dans la section 2.1.2. Les machines ainsi que la spécification sont modélisées par des RdP. Les modèles des machines M_1, M_2 et la spécification sont donnés sur la figure 2.13 (a), (b) et (c) respectivement.

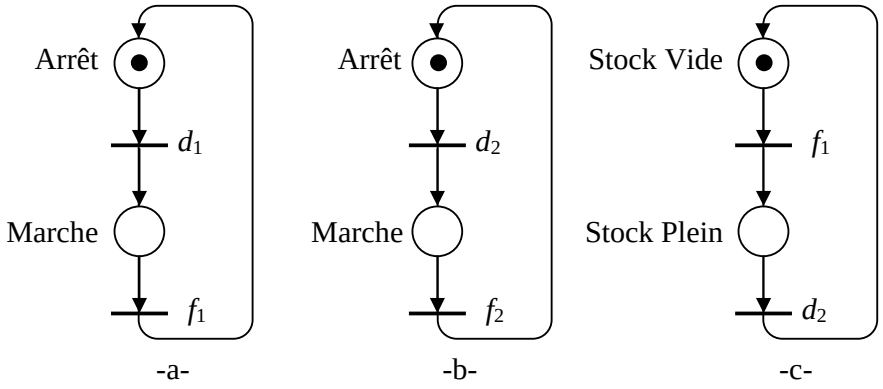


Figure 2.13. Modèle de RdP. a) Machine 1. b) machine 2. c) Spécification.

Les transitions d_1 et d_2 sont contrôlables et les transitions f_1 et f_2 ne sont pas contrôlables.

Le fonctionnement en boucle fermée de procédé global (M_1 et M_2) et la spécification nous donne le graphe de marquage de la figure 2.14.

Dans le fonctionnement en boucle fermée, une transition est franchissable si elle est validée par rapport au procédé et validée par rapport de la spécification.

Les transitions communes au modèle du procédé et à la spécification sont f_1 et d_2.

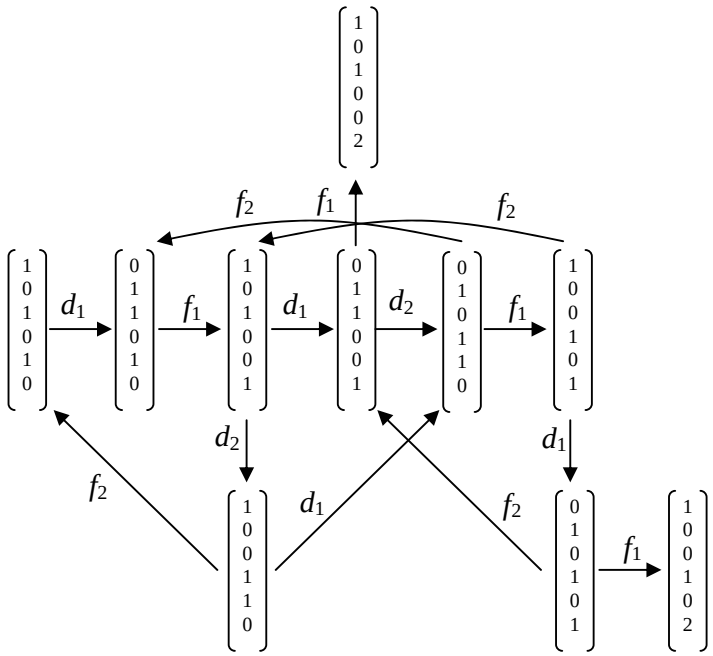


Figure 2.14. Graphe de marquage.

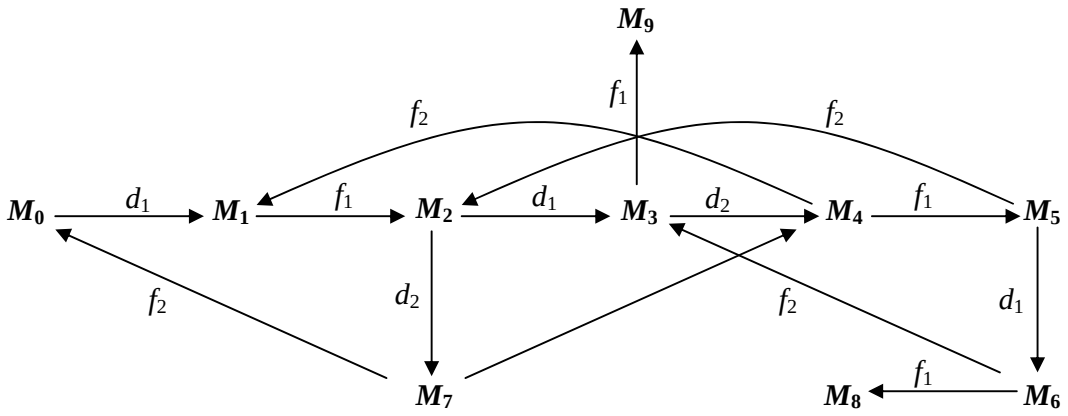


Figure 2.15. Graphe d'atteignabilité partiel.

L'ensemble des marquages interdits est { M_8 , M_9} ; M_9 = (1,0,1,0,0,2) , M_8 =(1,0,0,1,0,2).

L'ensemble des marquages dangereux est { M_3 , M_6}.

L'ensemble des marquages admissibles est {M_1 , M_2 , M_4 , M_5 , M_7}.

Pour ne pas atteindre le marquage M_3, il faut interdire la transition d_1 à partir du marquage M_2, ainsi que pour n'est pas atteindre le marquage M_6, il faut interdire la transition d_1 à partir du marquage M_5.

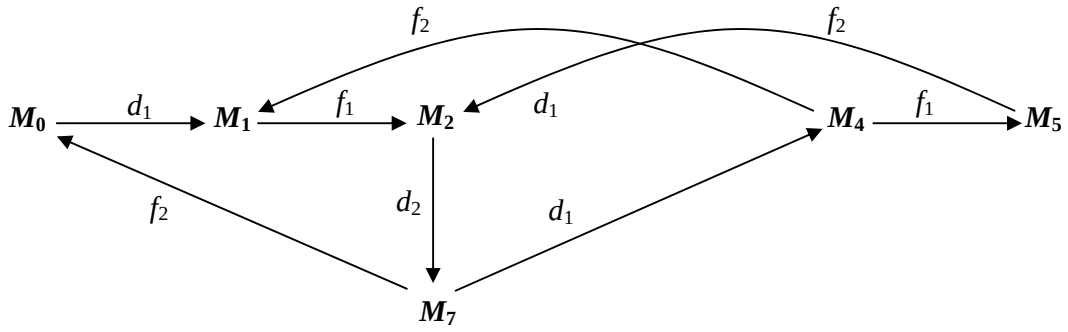


Figure 2.16. Graphe de comportement admissible R_C.

Pour la premiere transition à interdir on cherche la solution (W (P_{c1} , .) , M_0 (P_{c1})) :

$$M_3 (P_{c1}) = M_0 (P_{c1}) + W(P_{c1}, d_1) + W(P_{c1}, f_1) + W(P_{c1}, d_1) < 0$$

$$M_0 (P_{c1}) + 2\ W(P_{c1}, d_1) + W(P_{c1}, f_1) < 0$$

On résout le système composé des équations précedentes. On trouve la place de contrôle à ajouter P_{c1} telle que :

$$W(P_{c1}, .) = (-1, 0, 1, 0) \text{ et } M_0(P_{c1}) = 1$$

Pour la deuxième transition à interdir on a :

$$M_6(P_{c2}) = M_0(P_{c2}) + 2\,W(P_{c2}, d_1) + 2\,W(P_{c2}, f_1) + W(P_{c2}, f_2) + (P_{c2}, f_1) < 0$$

On résout le même système des équations, on trouve la place de contrôle à ajouter P_{c2} telle que :

$$W(P_{c2}, .) = (-1, 0, 1, 0) \text{ et } M_0(P_{c2}) = 1$$

On suprime la place P_{c2} parce qu'elle est redondante. Le modèle de RdP de système supervisé est donné dans la figure 2.17

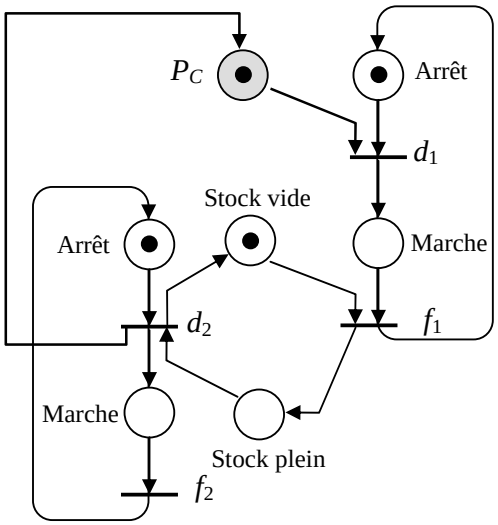


Figure 2.17. Le modèle de RdP de système supervisé.

2.2.2.3 Conclusion

La théorie des régions permet de synthétiser un RdP à partir d'un automate à états fini. Etant donné un RdP borné et un ensemble de marquages interdits par les spécifications, l'objectif est de déterminer les places à ajouter au RdP pour restreindre son graphe d'atteignabilité au sous-ensemble de marquages admissibles. Ces places de contrôle constituent la commande à imposer au système pour respecter les contraintes données. Deux algorithmes pour synthétiser le superviseur ont été proposés, le premier donne le comportement admissible du système commandé, c'est à dire le graphe d'atteignabilité, et le deuxième algorithme permet de faire de la synthèse en utilisant la théorie des régions (condition d'atteignabilité, équation de cycle, condition de séparation). Cette méthode donne

un superviseur optimal mais le nombre des inéquations en nombres entiers engendrées est important et il n'y a pas de méthodes efficaces pour résoudre de tels systèmes.

2.2.3 Synthèse de la commande basée sur les invariants de marquage

Cette méthode a été proposée par Yamalidou et Moody [YAM 96]. Elle présente une nouvelle approche pour construire un contrôleur en boucle fermée d'un système modélisé par un RdP. Le contrôleur se compose de places et d'arcs calculés à l'aide du concept d'invariants de marquage. Dans cette approche, les spécifications sont données par des contraintes linéaires entre les marquages des places du RdP. L'idée consiste à construire un invariant de marquage pour chaque contrainte, de telle manière que l'ensemble des marquages atteignables soit réduit aux seuls marquages vérifiant la contrainte.

Le contrôleur correspond à des places qui sont reliées aux transitions du RdP de procédé, de telle façon qu'il est garanti que le système n'atteindra pas un état interdit. Le procédé combiné du RdP du procédé et du contrôleur, possède des invariants de marquage nécessaires pour assurer que l'ensemble de contraintes ne sera pas violé.

Dans le cas général, le contrôleur n'est pas nécessairement optimal, mais il est calculé facilement et sa taille est égale au nombre des contraintes.

En outre, il peut être utilisé comme une très bonne estimation préliminaire pour les autres méthodes qui sont capables de calculer un contrôleur optimal d'un RdP. Sa dimension étant relativement petite, il simplifiera le calcul nécessaire pour atteindre le contrôleur optimal. Cette approche va être détaillée parce qu'elle sera utilisée dans notre travail de recherche.

Un rappel concernant les invariants de marquage est d'abord présenté dans la section suivante, la synthèse du contrôleur sera ensuite étudiée. Nous insisterons particulièrement sur le problème de l'optimalité de la solution obtenue.

2.2.3.1 Notation et définition

Une séquence du franchissement S réalisable à partir d'un marquage M_i peut s'écrire: M_i [S 〉 M_k. Un marquage M est atteignable depuis M_0 s'il existe une séquence S telle que M_0 [S 〉 M.

La matrice d'incidence *W* associée à un RdP correspond à sa structure (indépendante du marquage), une colonne de cette matrice correspond à la modification du marquage apportée par le franchissement de la transition correspondant.

Equation fondamentale

A chaque séquence de franchissement, est associé un vecteur caractéristique noté $\underline{S}$. C'est un vecteur de dimension *m* (nombre de transitions) où la composante numéro *j* correspond au nombre de franchissements de la transition T_j dans la séquence *S*.

Si la séquence de franchissement *S* est réalisable à partir d'un marquage M_i, le marquage atteint M_k est donné par l'équation fondamentale :

$$M_k = M_i + W \,.\, \underline{S}$$

$\underline{S}$ est le vecteur caractéristique d'une séquence *S* qui mène de M_i à M_k.

Invariants de marquage

L'invariant de marquage est une propriété structurelle importante pour analyser les RdP car il ne dépend pas du marquage. Il permet d'étudier la structure du réseau indépendamment de toute évolution dynamique. Des invariants de franchissement sont également définis dans les RdP, mais ils ne sont pas utilisés dans ce travail.

L'invariant de marquage correspond à une somme pondérée de marquages de places qui est constante quel que soit le marquage accessible.

Un invariant est déterminé à partir de vecteurs *X*, calculés en trouvant les solutions de l'équation suivante :

$$X^T . W = 0 \qquad (1)$$

Où *W* est la matrice d'incidence du RdP de dimension *n* x *m* et, *X* est un vecteur de dimension *n* appelé P-semi-flot.

L'invariant de marquage déduit de tout P-semi-flot est donné par la relation suivante :

$$X^T . M = X^T . M_0 \qquad (2)$$

Où : M_0 est le marquage initial du réseau, et *M* représente tout marquage accessible.

Nous allons décrire maintenant, comment les invariants de marquage peuvent être utilisés pour calculer de manière simple le contrôleur d'un procédé modélisé par un RdP.

2.2.3.2 Description de la méthode :

Le système qui doit être contrôlé est modélisé par un RdP, ce réseau s'appelle *modèle du procédé*, la matrice d'incidence du réseau du procédé est W_P. Il est possible que le comportement dynamique du procédé viole certaines contraintes imposées. Dans ce cas, un système de commande doit être synthétisé.

Le contrôleur fonctionne en parallèle avec le procédé, son rôle est de surveiller le procédé et de prendre des actions correctives pour éliminer tout comportement indésirable. Soit W_C la matrice d'incidence du RdP correspondant au contrôleur. Ce RdP est formé des transitions du modèle du procédé et d'un ensemble séparé des places. Le RdP du système commandé est formé du RdP du procédé et celui du superviseur.

L'objectif de superviseur est de forcer le système à respecter des contraintes de type :

$$\sum_{i=1}^{n} l_i \, M(P_i) \leq \beta \quad (3)$$

où :

$M(P_i)$ est le marquage de la place $P_{i,}$ et l_i, β sont des nombres constants entiers positifs.

Par exemple la contrainte $M(P_1) + M(P_2) \leq 1$ signifie que les places P_1 et P_2 ne peuvent pas être marquées en même temps. Ces contraintes sont ici des spécifications imposées par le cahier des charges.

Chaque inégalité de type (3) peut-être transformée en une égalité en ajoutant une variable positive et entière $M(P_c)$, et la contrainte devient : $M(P_1) + M(P_2) + M(P_c) = 1$, ou en général :

$$\sum_{i=1}^{n} l_i \, M(P_i) + M(P_c) = \beta \quad (4)$$

Cette variable représente une nouvelle place P_c qui contient le marquage supplémentaire nécessaire pour satisfaire l'égalité. Elle garantit que la somme pondérée des marques dans les places du réseau du procédé reste toujours inférieure ou égale à β. Elle respecte donc la spécification. La place qui maintient la contrainte appartient au RdP du contrôleur.

La structure du réseau de contrôleur sera calculée en observant que l'introduction de la variable M (P_c) introduit un invariant de marquage dans le système contrôlé défini par (4).

Calcul du contrôleur :

Chaque contrainte du type (3) imposée au réseau va avoir une variable associée avec elle, et chaque variable sera représentée dans le réseau du contrôleur comme une place. Donc la dimension (nombre de places) du réseau du contrôleur est égale au nombre des contraintes imposées. Chaque place de contrôleur va ajouter une ligne à la matrice d'incidence de procédé contrôlé W. Cette matrice sera composée de deux matrices, la matrice originelle de dimension n x m du modèle de procédé W_p et la matrice d'incidence du contrôleur W_c.

Les arcs reliant les places du contrôleur avec les transitions du RdP original seront calculés par l'équation des invariants de marquage (4), où les inconnues sont les éléments de la nouvelle ligne de la matrice W et le vecteur X est le P-semi-flot désiré qui est défini par l'équation (2), c'est-à-dire :

$$x^T = [l_1\ l_2 \ldots l_n\ 1].$$

Le problème du contrôle généralement peut être décrit comme suit : Toutes les contraintes du type (3) peuvent être groupées et écrites en forme matricielle suivante :

$$L\ M_P \leq b \quad (5)$$

où :

M_P est le vecteur de marquage du RdP de procédé.

L est une matrice de dimension $n_c \times n$, où n_c est le nombre des contraintes de type (3).

b est un vecteur de dimension n_c, dont les éléments sont de nombres entiers.

Toutes les équations des invariants du type (5), sont déterminées après l'introduction les variables M_C. Elles peuvent être écrites sous la forme matricielle suivante :

$$L.M_P + M_C = b \qquad (6)$$

où, M_C est un vecteur à n_c éléments qui représente le marquage des places de contrôleur.

Chaque invariant de marquage défini par (6) doit satisfaire :

$$X^T W = \begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} W_P \\ W_C \end{bmatrix} = 0$$

$$L.W_P + W_C = 0$$

Où la matrice [*L*, *I*] représente les n_c différents P-semi-flots.

I est la matrice unité avec $n_c \times n_c$ éléments.

La matrice W_C contient les arcs qui relient les places de contrôleur aux transitions du RdP du procédé. Etant donné un modèle de RdP d'un procédé W_P, et des contraintes que le procédé doit satisfaire (*L* et *b*), le RdP de contrôleur W_C est déduit de l'équation ci-dessus, soit:

$$W_C = -L.W_P \qquad (7)$$

C'est cette formule extrêmement simple qui a fait le succès de cette approche. Le marquage initial du RdP du contrôleur M_{C0} peut être calculé à partir de l'équation (6) :

$$L.M_{P0} + M_{C0} = b$$

Donc :

$$M_{C0} = b - L.M_{P0}$$

La matrice d'incidence du système supervisé est donnée par :

$$W = \begin{bmatrix} W_P \\ W_C \end{bmatrix}$$

Le marquage *M* et le marquage initial M_0 sont donnés par :

$$M = \begin{bmatrix} M_P \\ M_C \end{bmatrix} \qquad M_0 = \begin{bmatrix} M_{P0} \\ M_{C0} \end{bmatrix}$$

Considérons à nouveau l'exemple d'application donné dans la section 2.1.2.

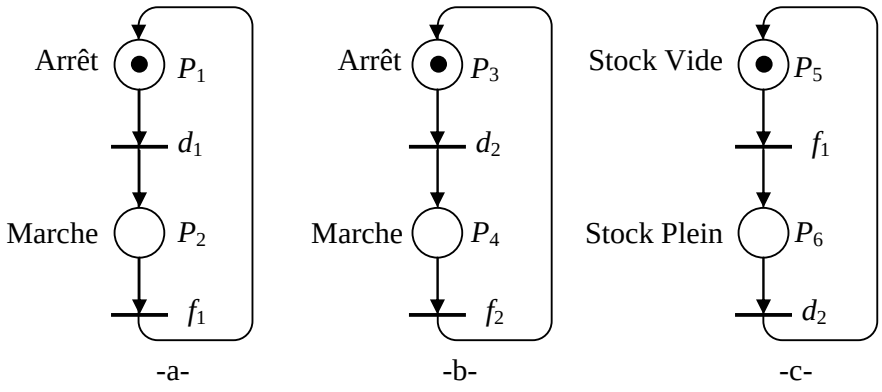


Figure 2.18. Modèle de RdP. a) Machine 1. b) machine 2. c) Spécification.

La spécification est modélisée par l'interdiction du marquage des places P_2 et P_6 simultanément, c'est-à-dire qu'il ne faut pas que la machine M_1 soit en état de marche si le stock est plein, ce qui est traduit par la contrainte suivante :

$M(P_2) + M(P_6) \leq 1$ le vecteur de contrainte L =[0 1 0 0 0 1]

La matrice d'incidence du procédé de fonctionnement en boucle fermé et W_P :

$$W_P = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad W_C = -L.W_P \Rightarrow W_C = [-1 \ 0 \ 1 \ 0]$$

Donc pour satisfaire la spécification, il faut ajouter une place de contrôle P_c où la matrice d'incidence de la partie de contrôle est W_C. Nous trouvons la même solution que celle que nous avons trouvée par l'approche de la section précédente (figure 2.19).

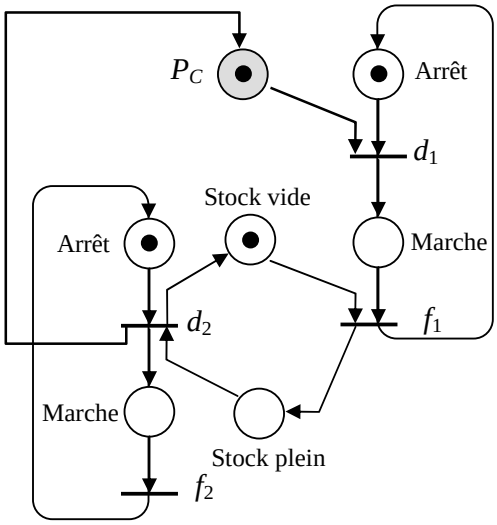


Figure 2.19. Le modèle de RdP de système supervisé est le même que de la figure 2.17.

Cependant, ceci est un cas favorable, nous allons voir ci-dessous que cela ne se passe pas aussi bien dans le cas général.

2.2.3.3 Problème des transitions incontrôlables

Soit W_U la sous matrice d'incidence représentant la partie incontrôlable. Elle contient les colonnes de W_P qui correspondent aux transitions incontrôlables.

$W_U \in \mathbb{Z}^{m \times n_U}$, n_U est le nombre des transitions incontrôlables.

Soit LW_u la sous-matrice de LW_p qui correspond aux transitions incontrôlables du procédé.

Et soit un ensemble de contraintes : $L\,M_P \leq b$.

Le RdP du contrôleur synthétisé par l'approche développée ici est donné par :

$$W_C = -L.W_P$$

Le contrôleur viole les contraintes si LW_u contient au moins un élément strictement positif, c'est-à-dire, s'il y a des arcs partant de la place de contrôle vers une transition incontrôlable. Donc si LW_u contient des valeurs positives, les contraintes ne sont pas satisfaites. Pour résoudre ce problème, une méthode intuitive consiste à remonter les branches jusqu'à trouver une transition contrôlable qui soit en aval de place de contrôle. Cette idée est présentée dans [YAM 96], mais il n'y a aucun algorithme qui permette de

trouver de manière systématique le contrôleur optimal. C'est l'inconvénient majeur de cette approche.

L'exemple ci-dessous illustre l'utilisation de cette technique.

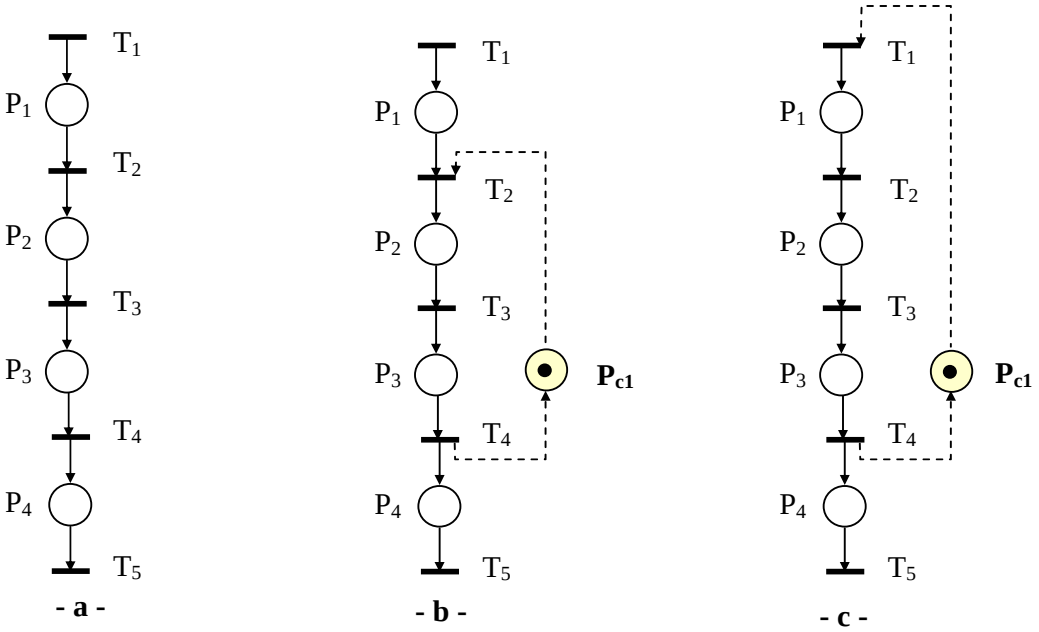


Figure 2.20. a) Système à contrôler b) Système avec place de contrôle c) Système contrôlé final

Supposons que l'on désire imposer au RdP de la figure 2.20.a, la contrainte suivante :

$$M(P_2) + M(P_3) \leq 1$$

Le système contrôlé est donné dans la figure 2.20.b. une place de contrôle a été ajoutée pour respecter la contrainte. Mais supposons que la transition T_2 est incontrôlable. Dans ce cas, une solution consiste ici à remonter la branche jusqu'à la transition contrôlable T_1 que l'on suppose contrôlable. Le système contrôlé final est donné dans la figure 2.20.c correspond au superviseur optimal. Mais cette méthode n'est pas générale.

En conclusion, cette approche est très puissante car elle permet de synthétiser de manière structurelle un contrôleur. Son principal inconvénient est qu'elle ne donne pas en général la solution optimale.

2.3 Objectif de la thèse :

Malgré les nombreux travaux basés sur la théorie de R&W [RAM 87], peu d'entre eux aboutissent à des résultats pratiquement mis en œuvre. Ceci met en évidence les difficultés

d'implantation déjà évoquées dans [BAL 93], que l'on peut regrouper selon les quatre points suivants :

- *L'interprétation du modèle RW* : dans la théorie de la supervision, il est supposé que le procédé génère des événements de façon spontanée. Le superviseur peut seulement interdire des événements contrôlables, et ne peut en aucun cas forcer des événements dans le procédé. Ce postulat va à l'encontre des approches classiquement adoptées par les propositions industrielles, selon lesquelles la commande se doit de forcer des événements dans le procédé.

- *L'implantation du superviseur* : le passage de la synthèse à l'implantation n'est pas systématique. En effet, la procédure de synthèse donne un modèle de superviseur qui n'est pas directement utilisable pour faire de la commande.

- *Les difficultés de modélisation* : la synthèse du superviseur requiert la connaissance des modèles automates du procédé et de ses spécifications. Une telle modélisation s'avére parfois difficile et ce même pour des exemples assez simples.

- *La complexité* : le problème classique de l'explosion combinatoire de l'espace des états du système présente souvent des difficultés d'implantation.

Dans ce chapitre, nous avons présenté trois approches apportant une réponse à certains problèmes mentionnés ci-dessus. L'approche de Krogh est limitative du point de vu de la structure du RdP utilisée et ne sera pas retenue. Certes, la théorie des régions donne une solution structurelle optimale mais difficile à mettre en œuvre sur le plan numérique. L'approche basée sur les invariants est la plus prometteuse même si elle ne garantit l'optimalité.

Dans le troisième chapitre, nous allons présenter l'approche de commande supervisée des SED proposée par Charbonnier [CHA 96]. Nous traiterons le cas où le langage des spécifications est non contrôlable.

Dans le chapitre 4, nous proposerons une méthode optimale et directe pour la synthèse de superviseurs sous forme de Grafcet. La notion d'optimalité est liée au nombre d'états atteignables par le système en boucle fermée, c'est-à-dire sous supervision. La notion directe est liée à son aspect d'implémentation sur les automates programmables industriels (API). La méthode utilise l'approche basée sur les invariants de marquage pour construire le Grafcet du superviseur.

Chapitre 3

Synthèse d'un superviseur basé sur le Grafcet

CHAPITRE 3

Synthèse d'un superviseur basé sur le Grafcet

Nous allons présenter dans ce chapitre les problèmes liés à l'utilisation de la théorie de Ramadge et Wonham pour la conception de commandes des SED ainsi que les différents travaux ayant traité ces problèmes. Puis nous allons présenter l'approche de commande supervisée des SED, proposée par Charbonnier [CHA 96], dans laquelle nous allons traiter le cas où le langage des spécifications est non contrôlable.

Chapitre 3

Synthèse d'un superviseur basé sur le Grafcet

3.1 Introduction

Dans la théorie de Ramadge et Wonham, un système à événements discrets est modélisé par un procédé et un superviseur en interaction. Le procédé est un générateur d'événements spontanés, instantanés et asynchrones [FER 99]. Tous ces événements sont en outre observables par le superviseur qui est un inhibiteur d'événements contrôlables. Ce système, le procédé supervisé, est de plus considéré comme isolé de son environnement (figure 3.1.(a)).

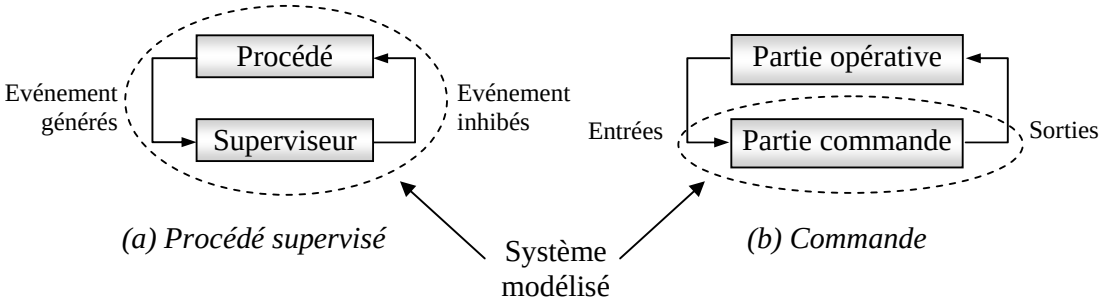


Figure 3.1. Les systèmes modélisés

La commande opérative d'un système de production est par contre considérée comme un système réactif en interaction avec une partie opérative. Lors de la modélisation de cette commande, le système n'est donc pas isolé de son environnement : il réagit aux évolutions de ses entrées et génère des sorties qui vont modifier l'état de la partie opérative (figure 3.1.(b)). Lorsque cette partie commande a un temps de réaction beaucoup plus court que celui de la partie opérative, une hypothèse simplificatrice consiste à considérer les entrées et les sorties qui en résultent comme synchrones [LES 98].

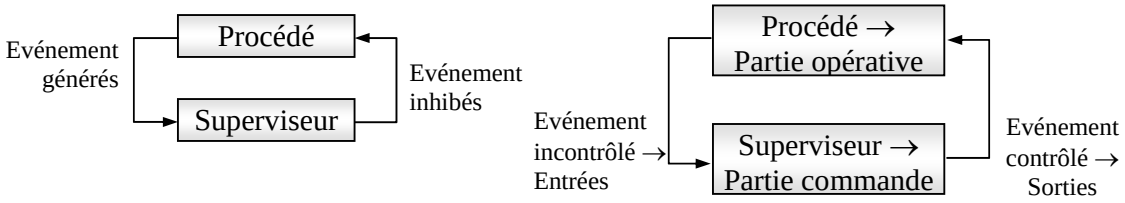
Ces différences sémantiques sont suffisamment importantes pour empêcher d'utiliser directement un modèle d'un procédé supervisé, sous la forme d'un automate à états par exemple, comme étant un modèle d'une commande opérationnelle. La conception de la commande à partir de la théorie de la supervision va donc nécessiter la construction de deux modèles : un modèle du procédé supervisé (sous la forme d'un automate à états par exemple) puis un modèle de la commande opérationnelle (en utilisant le Grafcet ou un langage de programmation d'API).

Les premières approches présentées [BRA 89][CHA 96] ici proposent de modifier la sémantique d'un des deux modèles construits afin de réduire l'écart sémantique entre eux, et faciliter ainsi le passage de l'un à l'autre. La seconde approche [NDJ 99] propose une comparaison des modèles produits afin d'en extraire le comportement commun. Nous présenterons chaque approche et discuterons de ses limites.

3.2 Différentes approches

3.2.1 Les événements forcés

Les travaux de [BRA 89] et [BAL 92] sont basés sur le concept d'événement forcé. Dans ces approches le superviseur ne conserve plus sa fonction originelle d'autorisation/interdiction d'événements générés par le procédé, il force maintenant certains événements à se produire dans le procédé. Le superviseur est alors considéré comme ayant la maîtrise des événements forcés qu'il génère (les événements contrôlables) alors qu'ils n'a pas celle des événements générés par le processus (les événements incontrôlables). Le superviseur est ensuite identifié comme étant la partie commande, le procédé comme la partie opérative, les événements contrôlables comme les sorties de la partie commande et les événements incontrôlables comme les entrées de la partie commande (figure 3.2).



(a) Ramadge et Wonham *(b) Brandin, Balemi*

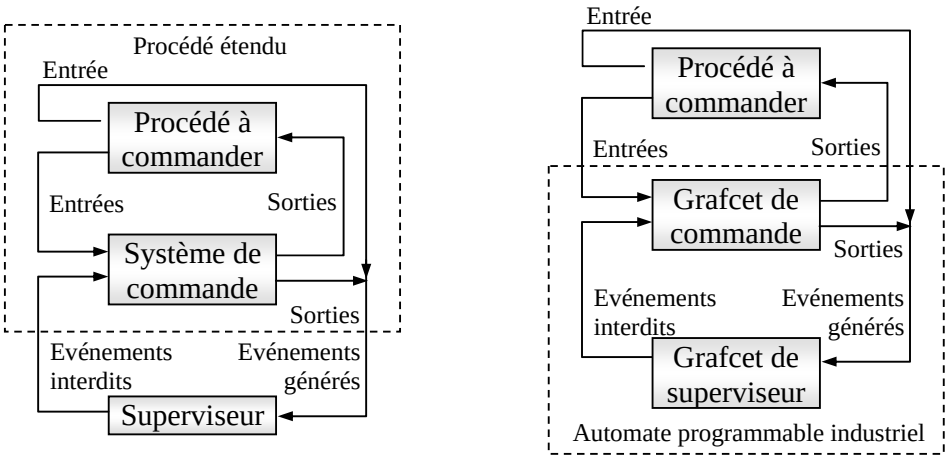
Figure 3.2. L'approche de Brandin et Balemi

Dans l'approche RW et dans le cas d'une supervision modulaire, un événement contrôlable peut être généré par le procédé si et seulement si il est autorisé par l'ensemble des superviseurs. En revanche, dans une approche à événements forcés, pour qu'un événement contrôlable soit forcé il suffit qu'il le soit par un seul superviseur. Les résultats de l'approche RW sur la contrôlabilité dans le cas modulaire ne sont donc pas transposables aux approches à événements forcés.

3.2.2 La commande supervisée

L'approche de Charbonnier [CHA 96] revient à considérer que le procédé est constitué d'un procédé à commander et d'un système de commande. Le procédé étendu alors obtenu est un générateur d'événement en interaction avec un superviseur, conformément à la théorie de la supervision (figure 3.3.(a)). La dynamique de la commande et de la supervision est modélisée par des Grafcets. Pour que la sémantique de la théorie de la supervision soit respectée, les Grafcets ne manipulent, en entrées et en sorties, que des événements. Les modèles Grafcets ont donc une sémantique un peu pauvre par rapport à celle du Grafcet normalisé puisqu'il n'y a plus de combinatoire dans les réceptivités, que l'évolution des situations n'est plus conditionnée par des états de variables ou des événements mais seulement par des événements, et que les actions ne sont que des impulsions. Par contre la séparation entre Grafcets de commande et Grafcets de supervision oblige à utiliser les notions d'actions conditionnelles et de variables internes utilisées pour conditionner les actions.

Les Grafcets sont volontairement de toute petite taille afin de simplifier l'analyse du problème et d'accroître la concision de la commande. Cela a aussi pour conséquence de transférer sur la coordination des graphes la complexité de la commande. En outre, la séparation des spécifications de fonctionnement entre la commande et la supervision est arbitraire. Elle l'est d'autant plus que commande et supervision sont implantées dans le même automate programmable industriel (figure 3.3.(b)).



a) Conceptuelle b) Pratique

Figure 3.3. La commande supervisée

3.2.3 Sûreté de fonctionnement

Outre la spécification du comportement nominal du système, la conception d'un système automatisé de production nécessite de prévoir des modes de fonctionnement d'exceptions. Ces modes de fonctionnement doivent assurer la sécurité du système et de son environnement, dont l'opérateur, tout en augmentant la durée de disponibilité du système : c'est l'approche sûreté de fonctionnement. Dans le cadre de cette approche, les modèles produits sont d'une taille beaucoup plus importante que lorsque seul le mode nominal (production *normale)* est étudié. Elle implique donc d'utiliser une démarche modulaire et hiérarchique permettant une conception, une adaptation et une maintenance plus faciles.

Nourelfath [NOU 97] propose une structure hiérarchique distinguant la commande, la supervision nominale et la surveillance réactive. La surveillance intègre alors les fonctions de détection, d'observation et de compensation et recouvrement. Comme dans l'approche précédente, le procédé et la commande sont vus comme un procédé étendu, le générateur d'événements, en relation avec une supervision. En revanche la supervision, l'inhibiteur d'événements, ne gère que le mode de fonctionnement nominal.

La démarche conceptuelle consiste à construire les automates à états décrivant d'une part le procédé et d'autre part les spécifications du comportement nominal. La modélisation de la détection nécessite la modification des automates décrivant le procédé afin d'y inclure les événements incontrôlables correspondant aux défaillances envisagées. L'observation en elle-même n'est pas modélisée par un automate à état mais est décrite par un ensemble de

relations entre les événements détectés sur le procédé et les événements activant les compensateurs.

En pratique les fonctions de détection et d'observation sont réalisées par l'ajout de procédures de chien de garde dans les Grafcets de commande. Tous les Grafcets construits (procédé, commande, supervision et surveillance) sont implantés dans le même automate programmable.

3.2.4 Remarques

Les travaux présentés ici montrent que non seulement la théorie de Ramadge et Wonham n'est pas directement implantable sous la forme d'une commande opérationnelle mais qu'elle n'est pas non plus facilement utilisable, soit pour concevoir une commande opérationnelle à partir de la spécification d'un procédé supervisé, soit pour valider une commande opérationnelle spécifiée par Grafcets. Ceci n'enlève rien à la théorie RW mais montre simplement que la distance sémantique entre cette théorie et la commande opérationnelle ne peut être franchie par des approximations ou des simplifications.

Le besoin existe toujours d'une méthode facilement utilisable, débarrassée des problèmes d'explosion combinatoire, permettant d'utiliser la théorie RW pour la conception d'une commande supervisée. Le problème de l'explosion combinatoire peut certainement être résolu par l'utilisation d'une méthode modulaire et hiérarchique, ou encore par l'utilisation d'un autre langage de spécification que l'automate à état : Le réseau de Petri par exemple.

❑

Dans la suite de ce chapitre nous allons détailler l'approche de commande supervisée proposée par Charbonnier [CHA 96]. Dans sont travail, il a trouvé le Grafcet du superviseur dans le cas où les spécifications sont contrôlables. Notre travail de recherche dans ce chapitre va consister à déterminer le modèle Grafcet du superviseur quand les spécifications ne sont pas contrôlables.

3.3 La commande supervisée

Ce concept hiérarchise les rôles dévolus à la commande et à la supervision. Dans cette approche, un système de commande peut forcer des événements à se produire dans le procédé et le superviseur garde sa fonction originelle définie par Ramadge&Wonham. Comme nous le verrons, cette approche permet de clarifier les notions d'entrées/sorties et de systématiser le passage de la synthèse à l'implantation de la commande.

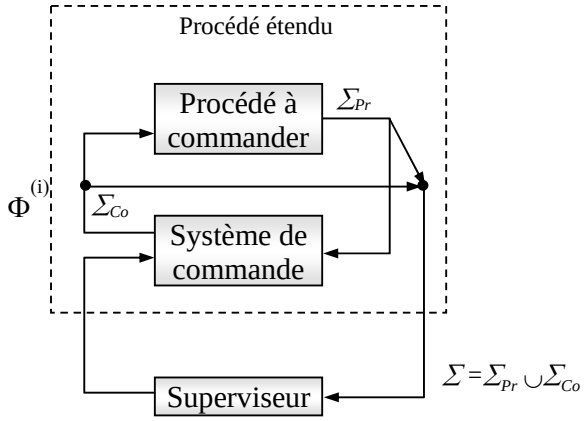


Figure 3.4. Schéma de commande supervisée

On appelle *procédé étendu* le procédé couplé à son système de commande (figure 3.4.). Pour un observateur extérieur, le procédé étendu est perçu comme un SED qui évolue spontanément en générant des événements. Si l'on souhaite imposer au procédé étendu le respect d'un ensemble de spécifications logiques de fonctionnement, celui-ci peut être couplé à un superviseur. Le superviseur observe de façon asynchrone (c'est-à-dire, non cadencée par une horloge) l'ensemble des événements qui sont générés dans le procédé étendu, c'est-à-dire, l'alphabet Σ. A l'instant logique *i*, le superviseur fournit au procédé étendu une liste d'événements interdits $\Phi^{(i)}$. Cela signifie que le procédé étendu peut uniquement évoluer en générant un événement de $\Sigma \setminus \Phi^{(i)}$.

Les sorties du procédé et du système de commande seront respectivement notées : Σ_{Pr} et Σ_{Co}. De plus, nous noterons Σ l'alphabet : $\Sigma_{Pr} \cup \Sigma_{Co}$.

Le schéma de commande supervisée est représenté dans la figure 3.4. Dans ce schéma, le procédé est perçu par le système de commande comme un SED qui génère des événements de Σ_{Pr}. Le système de commande est considéré comme un SED qui produit des événements de Σ_{Co}. Pour un observateur extérieur, c'est-à-dire, du point de vue du superviseur, le procédé étendu est perçu comme un SED qui génère spontanément des événements de $\Sigma = \Sigma_{Pr} \cup \Sigma_{Co}$.

Remarque 3-1 Dans le schéma hiérarchique de la figure 3.4, le superviseur *se cantonne à interdire l'occurrence de certains événements* dans le procédé étendu. Ainsi, le superviseur garde sa fonction originelle définie dans l'approche Ramadge.Wonham.

❑

Nous pouvons remarquer qu'un tel superviseur ne peut que restreindre le fonctionnement du procédé étendu. Dans la suite, nous appellerons *fonctionnement en boucle fermée*, le fonctionnement du procédé étendu couplé à son superviseur.

Remarque 3-2 : Dans la figure 3.4, les concepts de commande et de supervision sont clairement séparés. Il s'ensuit que la tâche de commande est indépendante de la tâche de supervision. Ainsi, si l'on modifie les spécifications à imposer au niveau de la supervision, le système de commande ne nécessite pas d'être modifié.

❑

Les sorties du procédé correspondent à des informations provenant de capteurs. En pratique, de tels événements ne peuvent pas être interdits par le superviseur. Les sorties du procédé seront donc considérées comme des événements *incontrôlables*, c'est-à-dire, $\Sigma_{Pr} \subseteq \Sigma_u$. En revanche, nous supposons que le superviseur peut toujours interdire au système de commande de produire un événement de Σ_{Co}, c'est-à-dire, d'envoyer une commande au procédé. Ainsi, l'ensemble des sorties Σ_{Co} de la commande est considéré comme un ensemble d'événements *potentiellement contrôlables*.

Pour un problème donné, il ne sera pas nécessaire d'interdire l'occurrence de tous les événements de Σ_{Co}. De façon générale, l'ensemble Σ_c des événements contrôlables sera donc défini comme un sous-ensemble Σ_{Co}, c'est-à-dire, $\Sigma_c \subseteq \Sigma_{Co}$. L'ensemble des événements incontrôlables sera alors naturellement défini par : $\Sigma_u = \Sigma \setminus \Sigma_c$ (où $\Sigma \setminus \Sigma_c$ représente l'ensemble des événements qui appartiennent à Σ et qui n'appartiennent pas à Σ_c).

Dans la figure 3.4, les sorties du superviseur correspondent à des entrées du système de commande. Ceci permet au superviseur d'interdire à tout instant, la production de certains événements contrôlables par le système de commande.

La synthèse d'une commande supervisée peut être basée sur les deux étapes suivantes :

(1) la synthèse du système de commande (s'il n'existe pas déjà) qui définit *spécifications de commande*.

(2) la synthèse du superviseur qui définit *spécifications de supervision*.

Le Grafcet est un outil puissant de description de systèmes logiques séquentiels qui offre un fort pouvoir de concision dans la modélisation. De plus, cet outil permet d'avoir une représentation claire des entrées et des sorties. Le Grafcet pourra alors être avantageusement utilisé pour la synthèse de la commande supervisée. En outre, le langage Grafcet est

couramment utilisé dans les automates programmables. Ainsi, l'utilisation du Grafcet rendra aisée l'implantation d'une commande supervisée.

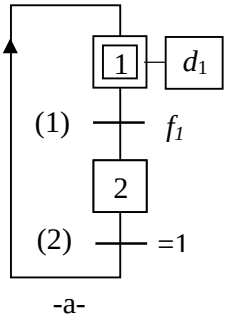
Le Grafcet permet d'utiliser aussi bien des événements que des grandeurs Booléennes pour ses entrées et ses sorties.

Dans notre étude, les spécifications de commande ainsi que les spécifications de supervision seront modélisées par le Grafcet. Cependant, la preuve de la contrôlabilité sera basée sur des modèles automates. Pour illustrer la démarche de la synthèse d'un superviseur on va prendre le même exemple des deux machines traité dans le deuxième chapitre.

Exemple : Considérons notre exemple de système manufacturier du chapitre précédent représenté dans la figure 2.5 ce système composé de deux machines identiques : M_1 et M_2, les deux machines travaillent de façon indépendante, puisent des pièces brutes en amont et rejettent des pièces usinées en aval.

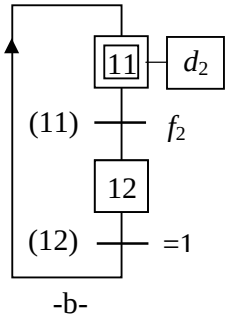
3.3.1 Spécification de commande :

Les deux machines sont supposées avoir un fonctionnement similaire. Les spécifications de commande peuvent être modélisées par le Grafcet de la figure 3.6.



-a-

Grafcet de commande de M_1



-b-

Grafcet de commande de M_2

Figure 3.5. Spécification de commande.

A l'instant initial, l'étape 1 est activée et l'action impulsionnelle d_1 (début de travail sur M_1) est exécutée. L'occurrence de l'événement f_1 (fin de travail sur M_1) conduit le Grafcet dans l'étape 2, la transition (2) qui est franchie immédiatement ramène le Grafcet dans son état initial, (un nouveau cycle recommence alors immédiatement sur M_1), la machine M_2 a le même fonctionnement. On pourrait être amené à considérer que les étapes 2 et 12 ainsi que les transitions 2 et 12 sont inutiles. Elles ne le sont cependant pas. En effet, un Grafcet qui aurait une seule étape et une seule transition modéliserait mal le cahier des charges. A

l'occurrence de f_1, l'étape 1 serait activée et désactivée, elle resterait active et l'action impulsionnelle ne se serait jamais exécutée.

L'ensemble des événement produits par le procédé est $\Sigma_{Pr} = \{f_1, f_2\}$ et l'ensemble des événement produits par la commande est $\Sigma_{Co} = \{d_1, d_2\}$.

Notation : soient a et b deux événements, la simultanéité de ces deux événements correspond à un événement que l'on notera $\{a, b\}$.

❑

Remarque 3-3 : Le temps de franchissement d'une transition d'un Grafcet est supposé infiniment petit, mais non nul [DAV 92]. Cependant, pour un observateur extérieur (c'est-à-dire, pour le superviseur), nous considérons que les événements f_1 et d_1 se produisent simultanément. Rappelons que les observations du procédé par le système de commande et par le superviseur (figure 3.4) sont asynchrones. Ainsi, dans notre exemple, le système de commande et le superviseur observeraient de façon simultanée l'occurrence de l'événement d_1 et f_1. Si le superviseur n'est pas un système plus rapide que le système de commande, le superviseur ne peut alors pas interdire l'occurrence de l'événement d_1 (sortie de la commande) après l'observation de f_1. Du point de vue du superviseur, l'occurrence de f_1 apparaît comme étant indissociable de celle de d_1.

❑

Le procédé étendu représente le procédé couplé à son système de commande.

3.3.2 Spécification de supervision :

Le fonctionnement de notre système manufacturier doit respecter la présence d'un stock de capacité limitée à 1 situé entre les deux machines. La présence du stock entre M_1 et M_2 impose que : *(1)* une pièce devra d'abord visiter M_1 puis M_2, *(2)* la machine M_2 ne peut commencer à travailler que si elle peut prendre une pièce dans le stock, c'est-à-dire, quand le stock est plein, et *(3)* la machine M_1 ne peut déposer une pièce dans le stock que si celui-ci est vide. Le stock est supposé vide dans son état initial. Le Grafcet de la figure 3.6.(a) permet de modéliser cette spécification.

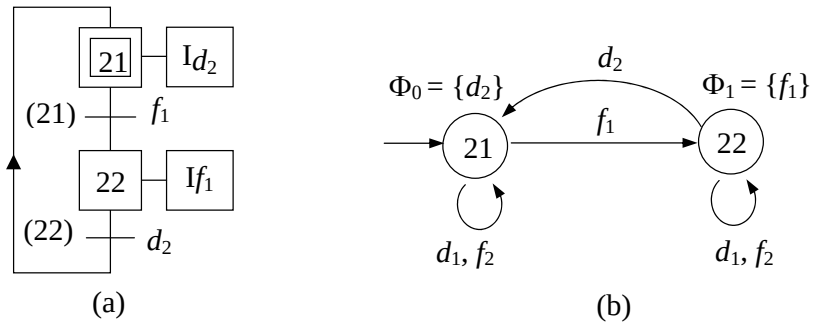


Figure 3.6. (a) Grafcet de spécification de supervision. (b) machine de Moore des spécification de supervision

Dans son état initial, l'étape 21 est active. L'occurrence de l'événement f_1 (fin de travail sur M_1 et dépôt d'une pièce dans le stock) conduit le Grafcet, dans l'étape 22. Dans cette étape, l'occurrence de d_2 (début de travail de M_2 et prise d'une pièce dans le stock) ramène le Grafcet dans son état initial. Nous pouvons remarquer que les états {21} et {22} correspondent respectivement aux états du système pour lesquels le stock est vide et le stock est plein. Le Grafcet comporte deux sorties Booléennes : I_{d_2} et I_{f_1}. Dans l'état {21}, l'action à niveau I_{d_2} associée à l'étape 21 est exécutée, c'est-à-dire, $I_{d_2} = 1$. Cela signifie que dans cet état où le stock est vide, le début de cycle de M_2 n'est pas toléré. De façon similaire, dans l'état {22}, $I_{f_1} = 1$ signifie que la fin de travail sur M_1 n'est pas tolérée. Ceci permet d'empêcher la fin de travail sur M_1 (événement f_1) et le dépôt d'une pièce dans le stock quand ce dernier est plein.

Dans la figure 3.4, les sorties du procédé étendu sont des événements de l'alphabet $\Sigma = \Sigma_{Pr} \cup \Sigma_{Co}$. Ainsi, les entrées du Grafcet de la figure 3.6.(a) sont, soit des événements de Σ_{Pr} (par exemple, l'événement f_1), soit des événements de Σ_{Co} (par exemple, l'événement d_2). Les sorties du Grafcet de la figure 3.6.(a) sont des grandeurs Booléennes I_σ où I_σ a la valeur 1 signifie que l'événement σ n'est pas toléré dans les spécifications de supervision.

Remarque 3-4 : Dans la figure 3.4, nous pouvons remarquer que le superviseur peut uniquement interdire l'occurrence d'événements en sortie du système de commande (événements de Σ_{Co}). Dans notre exemple (figure 3.6.(a)), les sorties du Grafcet sont I_{d_2} et I_{f_1} où l'événement d_2 est potentiellement contrôlable pourra alors être interdit par la supervision. Alors que l'événement f_1 (événement deΣ_{Pr}) est un événement incontrôlable ne pourra alors pas être interdit par la supervision. Lorsque la grandeur Booléenne I_{f_1} est égale à 1, cela signifie uniquement que l'occurrence de l'événement f_1 n'est pas tolérée. Notons que ceci est cohérent avec la définition d'une spécification.

❑

Formellement, un Grafcet de spécifications de supervision peut être défini de la façon suivante :

Définition 3-1

Un Grafcet de spécifications de supervision est un Grafcet tel que ses entrées sont des événements de Σ et tel que ses sorties sont des grandeurs Booléennes I_σ où σ est un événement de Σ. Pour tout état du Grafcet, l'événement σ est toléré si la grandeur Booléenne I_σ a la valeur 0.

❑

3.3.3 Obtention d'une machine de Moore des spécifications de supervision :

A partir du Grafcet de la figure 3.6.(a), on peut obtenir une machine de Moore des spécifications de supervision. Ceci est illustré dans la figure 3.6.(b). Dans ce modèle, les états {21} et {22} correspondent respectivement aux étapes 21 et 22 du Grafcet de la figure 3.6.(a). Depuis l'étape 21, l'occurrence de l'événement f_1 conduit le Grafcet de la figure 3.6.(a) dans l'étape {22}. Ceci est modélisé dans la figure 3.6.(b) par la transition {21}$\rightarrow$ {22} étiquetée par f_1. Cependant, si un événement autre que f_1 se produit dans l'étape 21, le Grafcet reste dans la même étape. Les événements d_1, f_2 sont alors associés à la boucle de l'état {21} dans la figure 3.6.(b). De façon similaire, la transition {22}$\rightarrow${21} est étiquetée par d_2 et Les événements d_1, f_2 sont associés à la boucle de l'état {22}. La sortie I_{d_2} associée à l'étape 21 du Grafcet de la figure 3.6.(a) signifie que l'événement d_2 n'est pas toléré dans l'état {21}. Ainsi, la liste des événements non tolérés Φ_0= {d_2} est associée à l'état {21} dans la machine de Moore. On obtient de façon similaire la liste Φ_1= {f_1} associée à l'état {22} dans la figure 3.6.(b). Un algorithme permet d'obtenir une machine de Moore à partir d'un Grafcet de spécification de supervision [CHA 96].

3.3.4 Synthèse de la commande supervisée :

Les spécifications de supervision qui sont modélisées par le Grafcet de la figure 3.6.(a), signifient qu'il est nécessaire d'interdire l'occurrence de l'événement d_2 lorsque le stock est vide (état {21}) ainsi que l'occurrence de f_1 lorsque le stock est plein (état {22}). Notons cependant que les événements d_2 et f_1 dans la figure 3.5, ne peuvent pas être inhibés. Ainsi, il sera nécessaire de modifier le Grafcet de la figure 3.5, (en conditionnant les sorties que l'on souhaite pouvoir interdire). Un tel système sera appelé *procédé étendu sous supervision.*

Remarque 3-5 : l'événement f_1 ne peut pas être interdit parce qu'il est incontrôlable. Pour cela on va interdire l'occurrence de l'événement contrôlable d_1 lorsque le stock est plein de manière à pouvoir garantir que l'événement f_1 ne puisse pas être généré par le procédé.

❑

3.3.5 Procédé étendu sous supervision :

Dans le Grafcet de la figure 3.5, nous pouvons remarquer que les sorties d_1 et d_2 ne sont pas conditionnées. Or, la supervision doit pouvoir interdire, à tout instant, l'occurrence des événements contrôlables d_2 et d_1. Pour ce faire, le Grafcet de la figure 3.5, devra être transformé conformément à la figure 3.7.

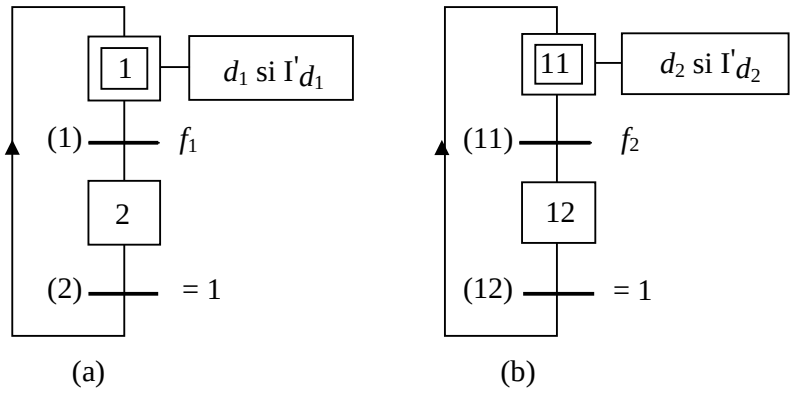


Figure 3.7. Procédé étendu sous supervision, obtenu à partit de la figure 3.5.

Considérons le Grafcet de commande de M_1 (figure 3.7.(a)) L'action impulsionnelle d_1 associée à l'étape 1 du Grafcet de la figure 3.5.(a) a été remplacée par l'*action impulsionnelle conditionnelle* : d_1 si I'$_{d_1}$, dans la figure 3.7.(a) (où I'$_{d_1}$ est le complémentaire de I$_{d_1}$). Dans la figure 3.7.(a), l'action d_1 peut être exécutée seulement si la condition associée est vraie, c'est-à-dire, I'$_{d_1}$ = 1 (ou de façon équivalente I$_{d_1}$ = 0, signifiant que l'événement d_1 n'est pas interdit). De façon similaire, l'action d_2 associée à l'étape 11 dans la figure 3.5.(b) a été remplacée par l'action : d_2 si I'$_{d_2}$, dans la figure 3.7.(b).

Définition 3-2

Une action conditionnelle impulsionnelle associée à une étape *i* est exécutée exactement quand l'étape *i* devient active si la condition associée est vraie. Mais, si la condition est fausse lorsque l'étape *i* devient active alors, l'action est exécutée sur le premier front montant de la condition si l'étape *i* est active, figure 3.8.

❑

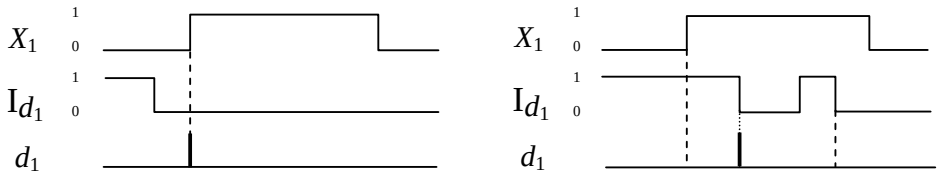


Figure 3.8. Exécution de l'action impulsionnelle conditionnelle d_1 si I'_{d_1}

De façon à systématiser l'obtention du modèle automate du procédé étendu sous supervision, le Grafcet de la figure 3.7 peut être transformé conformément à la figure 3.9.

Considérons le Grafcet pour M_1 (figure 3.9.(a)). Une transition (1') et une étape 1' ont été ajoutées en aval de l'étape 1. Dans la figure 3.7.(a), l'action d_1 si I'_{d_1} est associée à l'étape 1. Dans la figure 3.9, l'action impulsionnelle (non conditionnée) d_1 est associée à l'étape 1'. Nous pouvons remarquer que la condition I'_{d_1} est à présent associée à la transition (1') qui se trouve en amont de l'étape 1'.

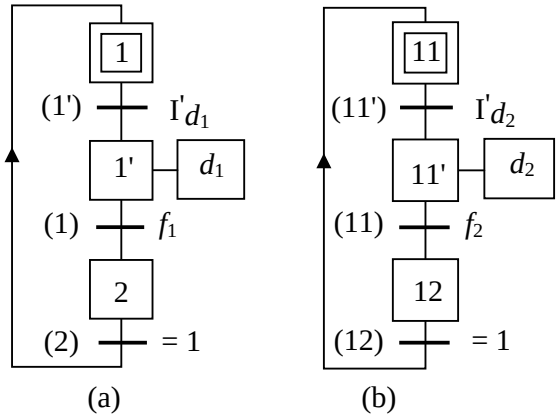


Figure 3.9. Deux Grafcet équivalents à ceux de la figure 3.7.

La règle de construction de Grafcet de la figure 3.9. à partir de la figure 3.7. est présentée dans [CHA 96]. Nous remarquons que le Grafcet de la figure 3.7 a le même comportement entrées/sorties que le Grafcet de la figure 3.9.

Dans la figure 3.9.(a), on peut supprimer maintenant les étapes 2 et 12 ainsi que les transitions (2) et (12), parce qu'elles n'ont pas aucune influence sur le fonctionnement du Grafcet. Le Grafcet de la figure 3.9 peut être transformé conformément à la figure 3.10.

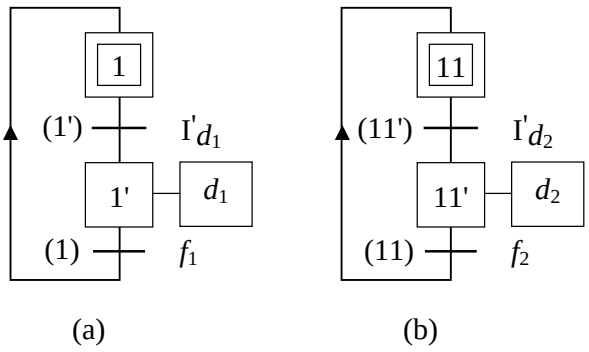


Figure 3.10. Deux grafcets équivalents à ceux de la figure 3.9.

3.4 Synthèse du superviseur

3.4.1 Modèle automate du procédé étendu sous supervision :

Soit P_1 (figure 3.11) le modèle automate du procédé étendu sous supervision pour M_1 obtenu à partir du Grafcet de la figure 3.10. (a).

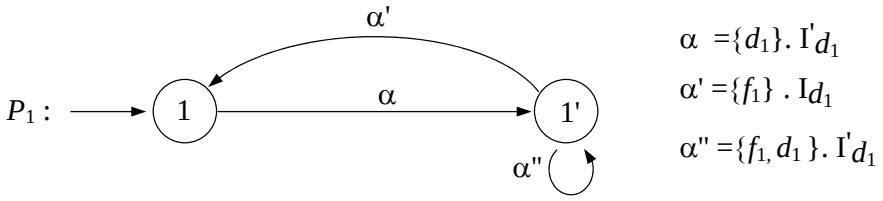


Figure 3.11. Automate du procédé étendu sous supervision pour M_1.

A l'instant initial, l'étape 1 du Grafcet de la figure 3.10.(a) est active et la machine 1 est en arrêt, quand I'd_1 est égal à 1 (Id_1=0) (l'événement d_1 autorisé) alors, l'étape 1' est immédiatement activée et l'action impulsionnelle d_1 est exécutée, L'étape 1' est active signifie qu'un travail est en cours sur M_1. Dans la figure 3.11, ce changement d'état est modélisé par la transition {1}→{1′} étiquetée par l'événement α={d_1).I'd_1. l'événement α représente l'occurrence de d_1 lorsque I'd_1=1. Quand l'événement f_1 se produit, à cet instant, si I'd_1 est égal à 0 (Id_1=1) alors l'événement f_1 ramène le Grafcet dans son état initial. L'événement α' = {f_1}. Id_1 associé à la transition {1'}→{1} dans la figure 3.11, modélise l'occurrence de f_1 lorsque Id_1 = 1. Dans le cas contraire, si Id_1 est égale à 0, l'étape 1' est alors activée de nouveau et l'action d_1 est exécutée. L'événement α" = {f_1, d_1}.I'd_1 associé à la boucle de l'état {1'} dans la figure 3.11, modélise l'occurrence simultanée de f_1 et d_1 lorsque I'd_1 = 1. Nous obtenons ainsi le modèle de la figure 3.11 qui est construit sur l'ensemble des événements {α, α', α"}.

Nous obtenons de façon similaire le modèle automate P_2 pour M_2. (Figure 3.12) qui est construit sur l'ensemble des événements {β, β', β"}.

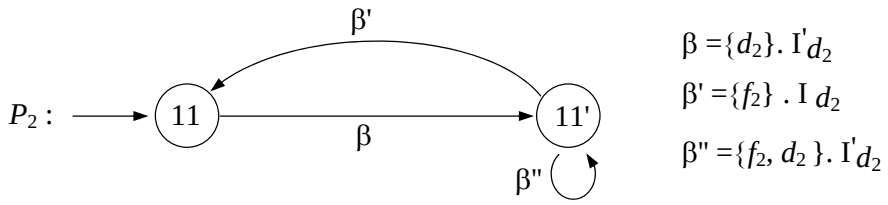


Figure 3.12. Automate du procédé étendu sous supervision pour M_2.

Remarque 3-6 : Le produit d'une grandeur Booléenne par un événement est un événement [DAV 92].

❑

On peut obtenir un modèle global P du procédé étendu sous supervision en composant les modèles P_1 et P_2, ce modèle est présenté dans la figure 3.13.

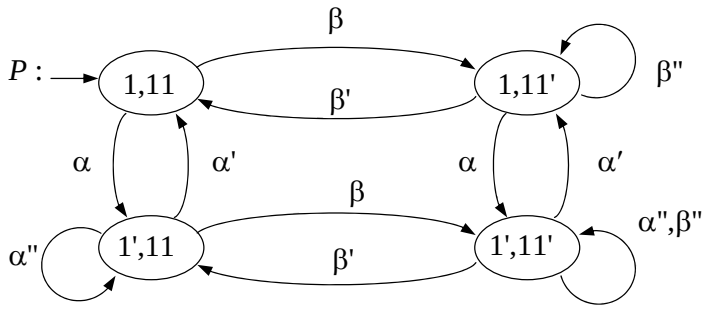


Figure 3.13. Modèle accepteur global P du procédé étendu sous supervision.

On notera Σ' l'alphabet des événements du modèle automate du procédé étendu sous supervision.

Dans notre exemple (Figure 3.13), l'alphabet Σ' est l'ensemble ={α,α', α", β, β', β"}. Notons que chaque événement de Σ' correspond à la simultanéité d'événement de Σ éventuellement conditionnée par les grandeur Booléennes I_{d_1}, I_{d_2} ou leurs complémentaires, nous avons la propriété suivante pour les événements de Σ' : tout événement de Σ' qui est conditionné est contrôlable.

En fait, à partir de l'état (1',11), les événements α', α" posent un problème, les deux événements sont contrôlables parce qu'ils sont conditionnés, mais mutuellement exclusifs,

c'est-à-dire un des deux est nécessairement incontrôlable (on ne peut pas interdire les deux en même temps). Si par exemple α" est contrôlable, on ne peut alors commander α'.

3.4.2 Modèle automate des spécifications de supervision :

La preuve de la contrôlabilité peut être basée sur des modèles accepteurs. Dans ce cas, il est nécessaire d'obtenir un modèle automate du procédé étendu sous supervision (3.11) ainsi que des spécifications de supervision définis sur l'alphabet Σ'. Ceci est illustré par le modèle automate de la figure 3.14.

Dans la figure 3.6. (b), la transition {21}→{22} est étiquetée par l'événement f_1. Dans la figure 3.14, cette transition est étiquetée par tout événement de Σ' qui contient l'événement f_1, c'est-à-dire, par l'événement α', α". Pour tout autre événement que f_1, la machine de Moore de la figure 3.6.(b) reste dans le même état.

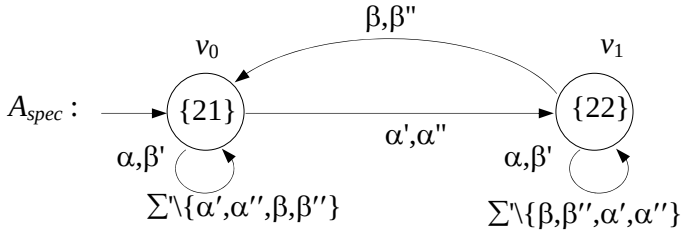


Figure 3.14. Modèle des spécification de supervision construit sur l'alphabet Σ'

De façon similaire, seuls les événements α', α" peuvent faire quitter l'état {21} dans la figure 3.14. Ainsi, la liste $\Sigma' \setminus \{\alpha', \alpha"\}$ doit être, dans un premier temps, associée à la boucle de l'état {21} dans la figure 3.14. La liste $\Phi_0 = \{d_2\}$ associée à l'état {21} de la machine de Moore (figure 3.6.(b)) signifie que l'événement contrôlable d_2 est interdit dans l'état {21}, c'est-à-dire, $I_{d_2} = 1$. Ainsi, tous les événements de Σ' qui sont conditionnés par I'_{d_2} sont interdits dans cet état. Les événements β et β" qui sont conditionnés par I'_{d_2} (figure 3.12) sont alors supprimés de la liste associée à la boucle de l'état {21} dans la figure 3.14 et nous obtenons la liste $\Sigma' \setminus \{\alpha', \alpha", \beta, \beta"\}$. On obtient de façon similaire, la liste des événements associés à la boucle de l'état {22}.

A partir des modèles accepteurs du procédé étendu sous supervision figure 3.13, ainsi que des spécification de supervision figure 3.14, on construit le composé synchrone de P et A_{spec}, c'est à dire $D = P \|_S A_{spec}$ (figure 3.15).

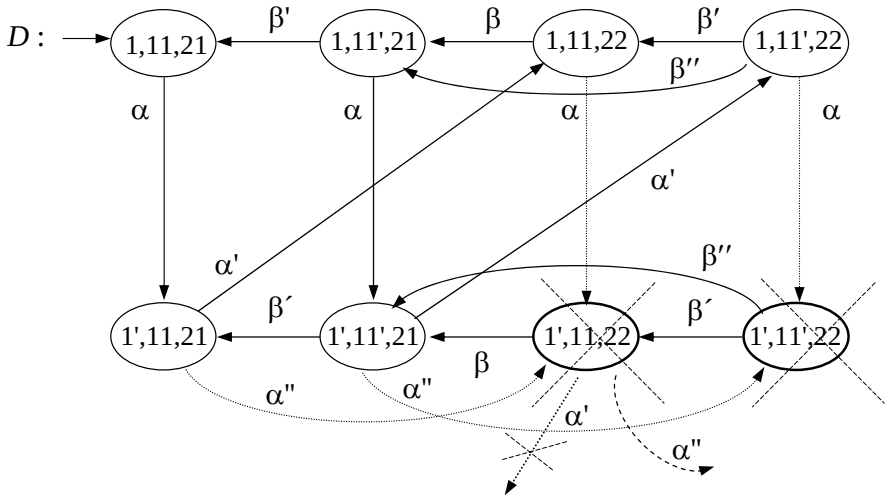


Figure 3.15 Modèle accepteur du fonctionnement en boucle fermée du système manufacturier.

3.5 Cas où les spécifications ne sont pas contrôlables

3.5.1 Contrôlabilité :

Considérons la séquence d'événements α α' α dans l'exemple de notre système manufacturier (la machine M_1 a débuté un cycle, terminé son cycle, puis débuté un nouveau cycle). Après cette séquence, le procédé P est dans l'état (1',11), cet état représente l'état du système pour lequel la machine M_1 est en marche et la machine M_2 est à l'arrêt. Depuis cet état, il existe une transition de sortie associée à l'événement α'. Cela signifie que α' peut être spontanément généré par le procédé. On dira alors que la séquence d'événements α α' α α' est possible dans le procédé, c'est-à-dire α α' α α' ∈ $L(P)$. La séquence d'événements α α' α conduit la spécification de la figure 3.14, dans l'état 22 qui correspond à l'état "stock plein". Depuis cet état, il n'existe pas de transition de sortie associée à l'événement α'. Cela signifie que l'événement α' n'est pas toléré par la spécification après la séquence α α' α (le dépôt d'une pièce dans le stock plein n'est pas toléré).

Ainsi, la séquence α α' α α' n'est pas élément du langage de spécification (α α' α α' ∉ $L(A_{spec})$). Cependant, l'événement α' qui est incontrôlable ne peut pas être interdit après la séquence α α' α. Donc le langage de spécification $L(A_{spec})$ est non *contrôlable* par rapport au langage du procédé $L(P)$.

- ∃ ω = α α' α ∈ $L(A_{spec})$.
- σ = α' ∈ $\sum_u$ tel que ωσ ∈ $L(P)$ et ωσ ∉ $L(A_{spec})$.

$\omega \in L_D$ et $\omega\sigma \notin L_D \Rightarrow$ Le langage L_D n'est pas contrôlable. Ainsi il n'existe pas de superviseur S tel que $L(S\backslash P)=L_D$ dans ce cas on va chercher un sous-ensemble de $L(D)$ qui est un langage contrôlable.

3.5.2 Synthétiser un modèle automate D' contrôlable à partir d'algorithme de Kumar

On va appliquer l'algorithme de Kumar [KUM 91] pour synthétiser un modèle accepteur du langage suprême contrôlable du fonctionnement en boucle fermée. Dans la figure 3.15, nous avons deux états défendus (1',11, 22), (1',11',22) nous les supprimons ainsi que les transitions associées à ces états, c'est-à-dire les arcs qui vont vers les états interdits, nous obtenons l'accepteur *D'* (Figure 3.16).

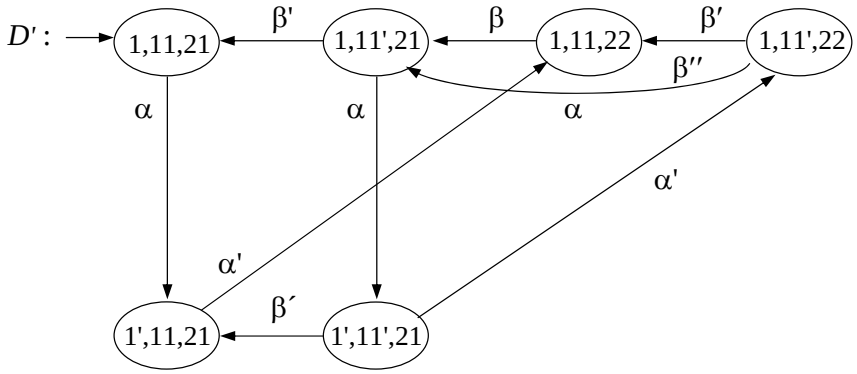


Figure 3.16. Modèle accepteur D', (automate de supervision).

3.6 Passage des automates au Grafcet

3.6.1 Obtention d'un modèle Grafcet à partir d'un modèle automate de supervision

Nous pouvons obtenir le Grafcet superviseur à partir de l'automate superviseur de manière systématique qui sera de plus structurelle.

Nous allons présenter cette approche, sur un exemple : passage de l'automate de la figure 3.16 au Grafcet de la figure 3.25. Nous considérons ensuite le cas général en décomposant le modèle global en structures élémentaires.

a) Structures élémentaires :

i. Transition autorisée :

Considérons la transition entre l'état *i* à l'état *i* +1 de la figure 3.17.(a). Elle a lieu sur l'occurrence de l'événement σ_k, s'il est autorisé ($I'\sigma_k = 1$). Ceci se traduit par le Grafcet de

superviseur de la figure. 3.17.(b). Celui-ci autorise l'événement et attend de voir son occurrence réalisée par le système de commande pour avancer (passage à l'étape *i* +1).

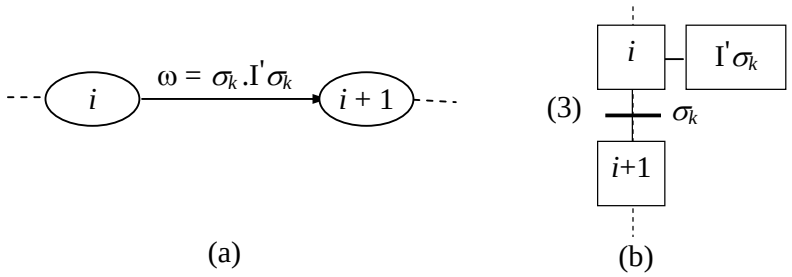


Figure 3.17. Passage de l'automate superviseur vers le Grafcet du superviseur pour un événement autorisé du type $\omega = \sigma_k . I'\sigma_k$.

Dans le cas où la commutation se fait sur l'occurrence d'événement successif, tel que α à partir de l'état *i* à l'état *i* + 1 (figure 3.18.(a)), on ajoute une transition et une étape supplémentaires. Ceci se traduit par le Grafcet de superviseur de la figure 3.18.(b). Le principe de commutation est similaire : Le Grafcet du superviseur observe σ_{k1}, produit $I'\sigma_k$ et observe ensuite σ_k.

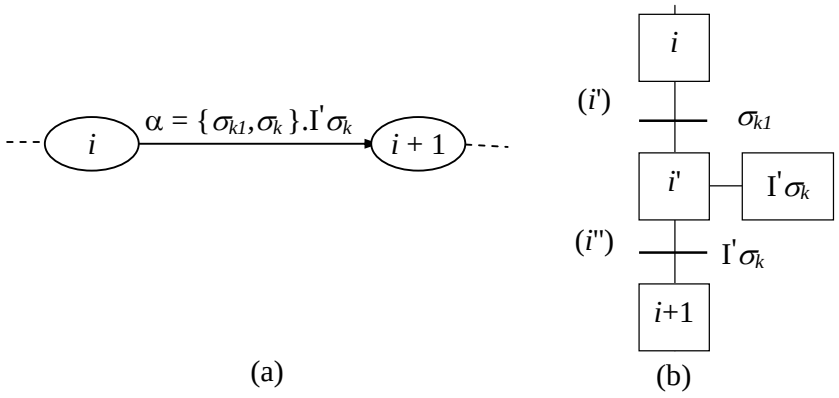


Figure 3.18. Passage d l'automate superviseur vers le Grafcet du superviseur pour un événement autorisé du type $\alpha = \{\sigma_{k1}, \sigma_k\}.I'\sigma_k$.

ii. Transition interdite :

Considérons la transition ω qui sort de l'état *i* de la figure 3.19.(a). On doit interdire l'occurrence de l'événement contrôlable σ_k, pour ne pas tomber dans un état interdit. Ceci se traduit par le Grafcet de superviseur de la figure 3.19.(b).

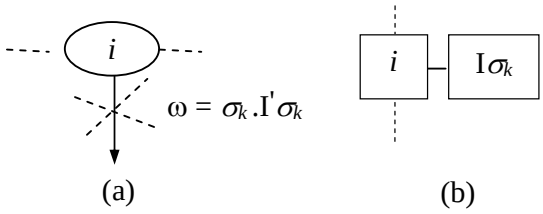


Figure 3.19. Passage de l'automate superviseur vers le Grafcet du superviseur pour un événement interdit du type $\omega = \sigma_k . I'\sigma_k$.

iii. Transition interdite et transition autorisée :

Ceci revient à faire la somme des deux structures : on autorise ω, i.e. $I'\sigma_k = 1$ et on interdit ω', i.e. $I\sigma_{k1} = 1$. Conformément à la figure 3.20.

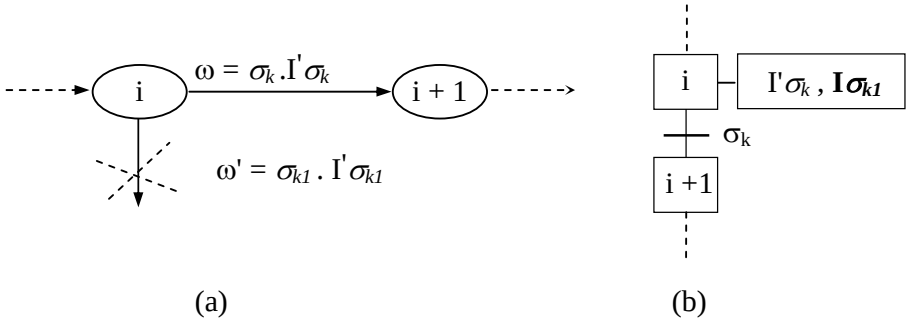


Figure 3.20 Passage de l'automate superviseur vers le Grafcet du superviseur pour un événement autorisé et un événement interdit

b) Procédure de construction du Grafcet dans le cas général :

Soient *i* et *i* +1 deux états de l'automate de supervision, et soit ω et ω' deux événements de Σ (figure 3.21) :

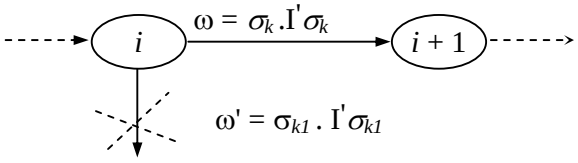


Figure 3.21

i. Etape (1) : A chaque état de l'automate, on crée une étape dans le Grafcet, les transitions entre les étapes sont celles qui relient les états correspondants de l'automate.

ii. Etape (2) : Pour réaliser la commutation sur l'occurance de ω il faut appliquer les transformations données dans la figure 3.22.

iii. Etape (3) : Enfin, pour interdire la commutation sur l'occurrence de ω' qui conduit à un état interdit, on ajoute à l'étape *i* du Grafcet précédant, l'événement $I\sigma_{k1}$ qui garantit de ne pas tomber dans un état interdit (figure 3.22)

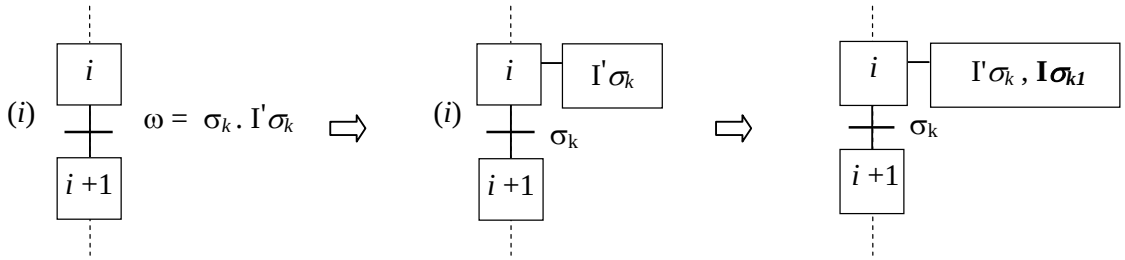


Figure 3.22. Les étapes de passage de l'automate superviseur vers le Grafcet superviseur dans le cas général

3.6.2 Grafcet de superviseur final

On appliquant la première étape, on obtient le Grafcet de superviseur présenté dans la figure 3.23.

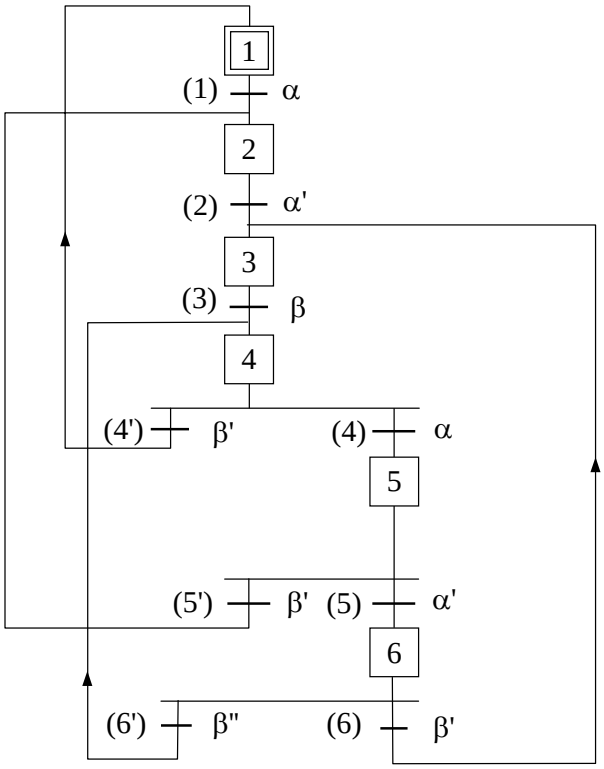


Figure 3.23. Création des étapes du Grafcet correspondant aux états de l'automate D'.

Nous remarquons qu'à partir de l'étape 6 dans la figure 2.23, nous avons deux transitions mutuellement exclusives, les deux transitions sont conditionnées donc elles sont contrôlables mais elles ne sont pas possibles en même temps, parce que si nous appliquons les procédures précédentes nous obtenons une évolution non déterministe (figure 24).

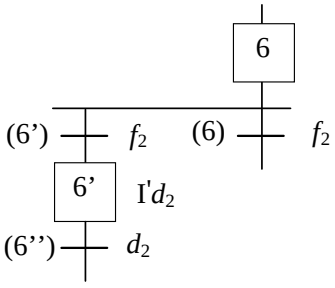


Figure 3.24

Donc il faut faire un choix. Si on choisit de contrôler β' on supprime la transition β" et on obtient la figure 2.25. si on choisit de contrôler β" on supprime la transition β' et on obtient la figure 2.26

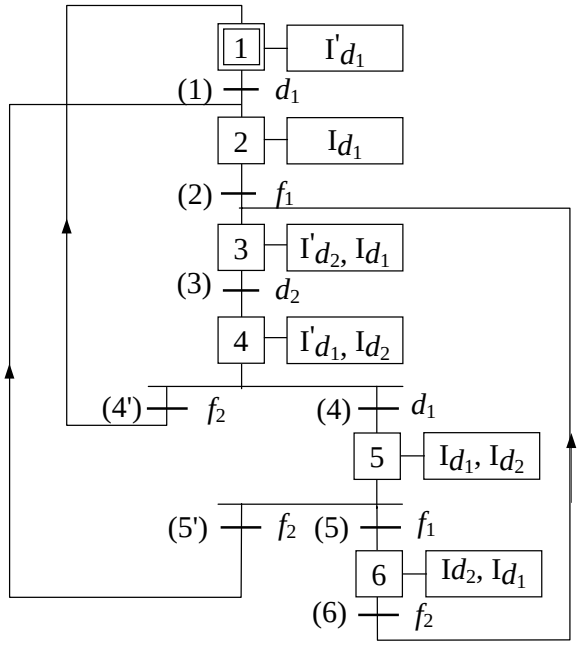


Figure 3.25. Grafcet de superviseur

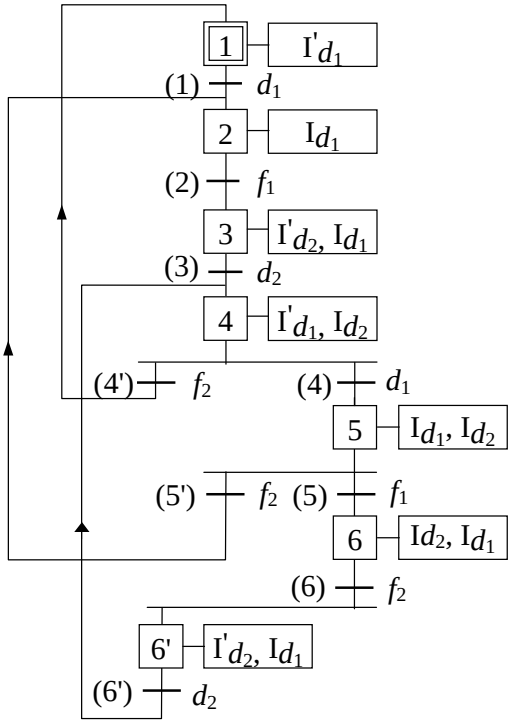


Figure 3.26

Les Grafcets des figures 3.25 et 3.26 sont équivalents et ils ont le même comportement où ils peuvent interdire les mêmes événements. Pour prouver cette équivalence, nous construisons les automates correspondant à chaque Grafcet, nous obtenons les automates de la figure 3.27 et 3.28. nous remarquons que dans l'automate de la figure 3.28 l'état 3 et l'état 6` vont interdire les mêmes événements, donc ils représentent un seul état celle qui est l'état 3 dans la figure 3.27.

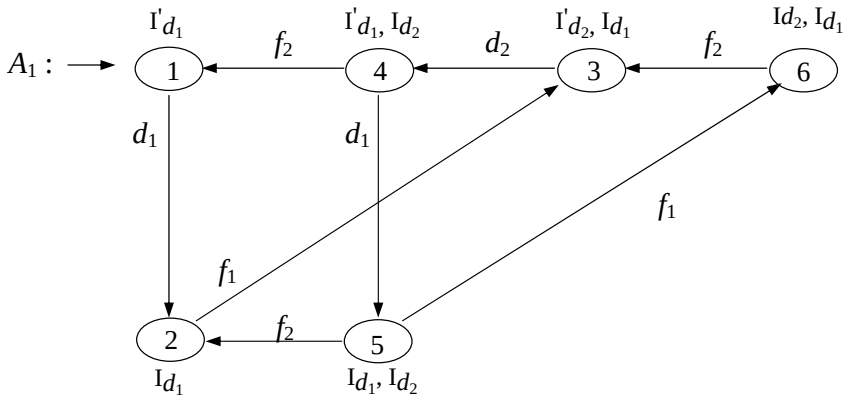


Figure 3.27 Automate correspond au Grafcet de la figure 3.25

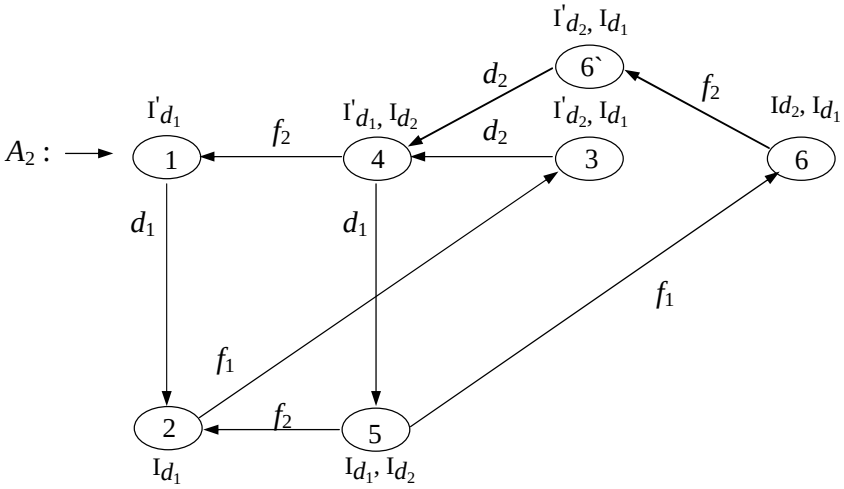


Figure 3.28 Automate correspond au Grafcet de la figure 3.26

Donc les événements β" et β' sont mutuellement exclusifs, et ils ne se produisent pas ensembles.

3.7 Conclusions et Remarques

La commande supervisée du système manufacturier est simplement effectuée en implantant le Grafcet de commande ainsi que le Grafcet de superviseur obtenu, dans un automate programmable.

Dans ce chapitre nous avons proposé une approche pour synthétiser un superviseur dans le cas où le langage des spécifications de supervision est non contrôlable par rapport au langage du procédé étendu sous supervision, et nous avons proposé une méthode systématique de passage de l'automate superviseur vers le Grafcet superviseur de manière structurelle. Cependant dans cette approche, un inconvénient majeur subsiste : Le Grafcet final de superviseur obtenu par cette approche contient un nombre important d'étapes qui est identique à l'automate synthétisé. On imagine alors la complexité des superviseurs que l'on obtient dans le cas d'applications complexes.

C'est pour cela que dans la suite de ce travail, on se propose de synthétiser le Grafcet de superviseur directement à partir d'un Grafcet de spécification, sans passer à des modèles automates. Ceci constitue l'objet du chapitre suivant.

Chapitre 4

Synthèse structurelle

CHAPITRE 4

Synthèse structurelle

Nous avons présenté dans le chapitre précédent une méthode de synthèse d'un superviseur calculé directement à partir de l'automate contrôlable. Nous avons vu comment passer automatiquement de cet automate vers le Grafcet superviseur. L'inconvénient majeur de cette approche est que la complexité du Grafcet obtenu est celle de l'automate. Il est alors dommage que, partant de modèles concis du procédé et des spécifications, on obtienne des modèles de taille considérable. Ceci nous a amenés à chercher une nouvelle approche qui puisse fournir un système de commande dont la taille est voisine de celle des modèles de départ. Il devient alors nécessaire de contrôler le système avec les mêmes objets que ceux qui l'ont défini. Cela signifie que les modèles du système physique contiennent des étapes et des transitions. Il faut alors concevoir le système de commande par des étapes et des transitions. Pour atteindre cet objectif, nous allons utiliser les invariants de marquage pour construire un contrôleur qui sera un Grafcet, en s'affranchissant de la théorie R&W.

Chapitre 4

Synthèse structurelle

4.1 Introduction

Dans le concept de commande supervisée, le Grafcet est utilisé pour faire la synthèse du superviseur [CHA 96]. Les modèles Grafcet du procédé et des spécifications sont donnés. La synthèse est obtenue en passant par les modèles automates et en appliquant l'algorithme de Kumar. On obtient ainsi un modèle automate qui correspond au superviseur optimal. Malheureusement, il n'est pas possible de retrouver la structure de Grafcet du départ. La seule solution consiste à construire un Grafcet dont la taille est celle de l'automate du superviseur optimal. Cette taille est souvent considérable, ce qui limite l'application de cette technique pour les problèmes réels [KAT 01]. Cela nous a conduit à poser le problème suivant: est-il possible de construire un superviseur de taille voisine du celle du modèle de départ ? Le point initial est constitué par le Grafcet modélisant le comportement désiré en boucle fermée. Ce modèle doit être ensuite complété pour pouvoir respecter les spécifications du cahier des charges. Pour pouvoir appliquer de manière directe la théorie du contrôle par supervision, nous avons considéré la classe de Grafcet où seuls des événements sont associés aux transitions.

Le respect des spécifications revient souvent à déterminer des états interdits qui doivent devenir inaccessibles par une action du superviseur. Un moyen d'y arriver est d'ajouter des places de contrôle. Ce problème a été étudié en considérant les RdP, il repose sur la théorie des régions [GHA 01]. La méthode proposée consiste à recenser tous les états admissibles en construisant le graphe des marquages et à ajouter des places de contrôle pour interdire l'accès aux états interdits. On obtient ainsi un système d'inéquations en nombres entiers à résoudre. L'inconvénient majeur de cette approche est qu'il n'y a pas de méthodes efficaces pour résoudre ce système comme nous l'avons vu dans le deuxième chapitre (2.2.2). Une autre approche, basée également sur RdP exploite le concept d'invariant de marquage de places. L'ensemble des états admissibles est donné par des relations d'inégalités. L'ajout d'une place de contrôle permet d'interdire tous les marquages qui ne respectent pas l'inégalité. Il s'agit

ici, d'une méthode structurelle et simple d'utilisation cette méthode a été présentée dans le deuxième chapitre (2.2.3). Malheureusement, cette technique n'est pas optimale. Nous l'avons cependant retenue car elle constitue le point de départ de notre recherche.

Dans le travail que nous présentons ici, nous considérons le Grafcet comme outil de modélisation. Pour pouvoir utiliser l'approche basée sur les invariants, il faut d'abord déterminer l'ensemble des états admissibles sous la forme d'inégalités sur les variables représentant les étapes du Grafcet. Ces inégalités seront déterminées à partir de l'automate représentant le fonctionnement désiré en boucle fermée. Le marquage dans un Grafcet est booléen, une étape est active ou inactive, cette propriété nous facilitera la détermination de ces inégalités. La synthèse du superviseur sera effectuée par la détermination d'étapes de contrôle.

Dans la suite nous allons utiliser le terme de contrôleur à la place de superviseur parce que le superviseur défini dans le deuxième et troisième chapitre constitue une partie séparée du système de commande. Il est implanté en parallèle avec le système. Dans ce chapitre le système de commande et le superviseur sont confondus et le résultat sera appelé contrôleur.

La méthode de Commande basée sur les invariants de marquage a été proposée par Yamalidou et Moody [YAM 96]. Elle présente une nouvelle approche pour construire un contrôleur en boucle fermée d'un système modélisé par un RdP. Le contrôleur se compose de places et d'arcs calculés à l'aide du concept d'invariants de marquage. Dans cette approche, les spécifications sont données par des contraintes linéaires entre les marquages des places du RdP. L'idée consiste à construire un invariant de marquage pour chaque contrainte, de telle manière que l'ensemble des marquages atteignables soit réduit aux seuls marquages vérifiant la contrainte. Une des difficultés de cette technique est qu'il faut que les places de contrôle aient une transition de sortie qui soit contrôlable. Lorsque ce n'est pas le cas, il faut remonter le chemin jusqu'à ce que l'on rencontre une transition contrôlable. Ce dernier point n'est actuellement pas résolu.

La présentation formelle de cette approche sera donnée dans la suite de ce chapitre.

Association Grafcet-théorie de supervision- Invariants de marquage

Notre objectif est de faire la synthèse d'un contrôleur basé sur l'outil de modélisation Grafcet. La figure 4.1 représente les étapes nécessaires pour synthétiser le contrôleur.

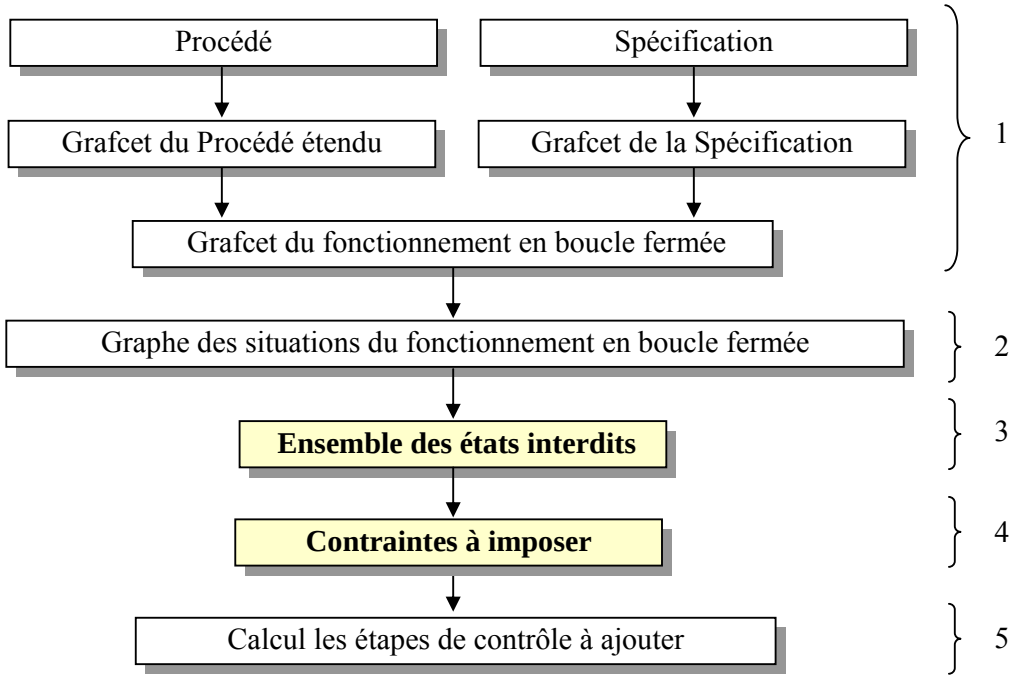


Figure 4.1 Les étapes nécessaires pour synthétiser le contrôleur.

Le point de départ consiste à modéliser le procédé étendu et les spécifications par des Grafcets. Le comportement désiré en boucle fermée est obtenu par la synchronisation des deux Grafcets (étape 1 dans la figure 4.1). La construction du graphe des situations à partir du Grafcet synchronisé permettra de construire l'automate du comportement désiré en boucle fermée (étape 2). On en déduira alors l'ensemble des états interdits quand ils existent par l'étude de la contrôlabilité de la spécification par rapport au procédé (étape 3). Pour pouvoir appliquer l'approche basée sur les invariants, il faut déterminer l'ensemble des contraintes donnant l'ensemble des marquages autorisés (étape 4). On verra que ces deux ensembles ne sont pas équivalents dans le cas général. Mais du fait que dans un Grafcet le marquage est booléen, on montrera alors qu'il y a équivalence entre l'ensemble des contraintes et l'ensemble des situations autorisées. La dernière étape est l'étape de synthèse du contrôleur et le calcul les étapes du contrôle à ajouter au modèle Grafcet du fonctionnement en boucle fermée, qui interdiront l'accès aux états interdits.

4.2 Modélisation par le Grafcet :

4.2.1 Grafcet du procédé étendu

Un procédé physique n'évolue jamais spontanément, il reçoit des entrées (actions provenant d'un système de commande) et produit des sorties (état des capteurs). C'est pour cela que la notion de procédé étendu a été définie dans [CHA 96].

Définition 4.1

On appelle procédé étendu le procédé couplé à son système de commande. Le modèle Grafcet du procédé étendu sera noté G_p.

❑

Exemple : Pour illustrer les idées que nous développons dans ce chapitre, nous allons considérer l'exemple d'un système manufacturier composé de deux machines indépendantes M_1 et M_2, équipées chacune d'un robot de transfert vers un banc de test où les pièces usinées sont testées (Figure 4.2).

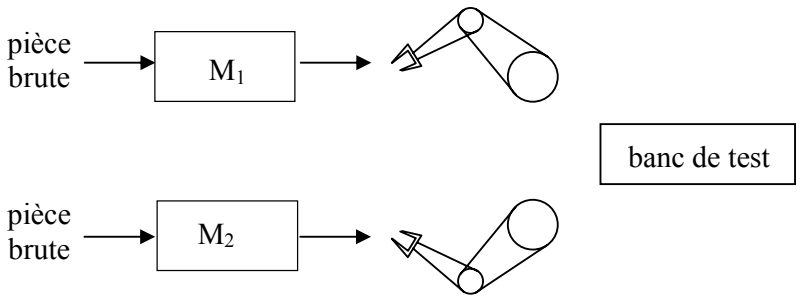


Figure 4.2. Système manufacturer

Chaque machine M_i ($i \in [1, 2]$) a le cycle de fonctionnement suivant: à l'occurrence de l'événement c_i le travail sur la machine commence (action D_i). Lorsque le travail est fini (occurrence de l'événement f_i), la pièce est transférée par le robot sur le banc de test (action T_i) et un nouveau cycle peut démarrer (occurrence de l'événement t_i). Les événements c_1 et c_2 sont les seuls événements contrôlables, on aura donc les ensembles d'événements suivants:

$\Sigma_c = \{c_1, c_2\}$ et, $\Sigma_u = \{f_1, t_1, f_2, t_2\}$.

Le Grafcet du procédé étendu pour ce système manufacturier est donné à la figure 4.3.

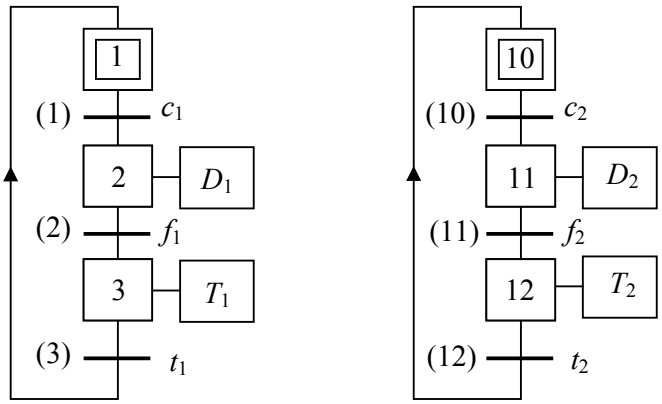


Figure 4.3. Grafcet du procédé étendu.

4.2.2 Grafcet de la spécification

Les spécifications sur l'état sont données souvent comme un ensemble de situations interdites ou de situations autorisées pour le procédé à contrôler. Les contraintes peuvent se décliner en :

- Conflits de ressources : plusieurs composants du procédé veulent accéder à la même ressource cette contrainte peut être modélisée en termes d'exclusion mutuelle.
- Blocages : une tâche ne se termine jamais.
- Dépassement de capacité : pour un stock de capacité finie.
- Séquencement : accès prioritaire à des ressources et relations de précédence dans des opérations (notions de séquences interdites et de séquences imposées). C'est le cas de notre exemple.

Définition 4.2

On appelle spécifications l'ensemble des contraintes séquentielles auxquelles sera soumis le procédé étendu.

❑

Naturellement, il n'est pas toujours possible de modéliser les spécifications par un Grafcet. Cela dépend de la définition exacte de ces spécifications. C'est seulement un problème de modélisation qui est exactement identique au problème rencontré avec les automates.

Dans l'exemple du système manufacturier, on désire que les pièces soient transférées dans l'ordre : pièce produite par la machine M_1 suivie d'une pièce produite par la machine M_2 et ainsi de suite. Cette spécification est modélisée par le Grafcet de la figure 4.4.

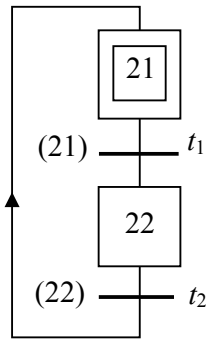


Figure 4.4. Grafcet de Spécification

Définition 4.3

On appelle Grafcet de spécification le modèle Grafcet qui définit les spécifications. Ce modèle sera noté G_s.

❑

4.2.3 Grafcet du fonctionnement en boucle fermée

4.2.3.1 Synchronisation de deux Grafcets

Le fonctionnement en boucle fermée correspond au comportement du procédé sous l'influence du superviseur. Il est obtenu par la synchronisation des Grafcets G_p et G_s. Il s'agit d'une opération structurelle plus efficace que celle du produit de deux automates puisqu'elle est indépendante du marquage initial. Elle consiste à fusionner structurellement les transitions identiques. C'est l'un des avantages de l'utilisation de Grafcet. Dans notre cas, la complexité de cette opération est polynomiale, $O(m^2)$, où m est le nombre de transitions. Ce résultat est bien connu dans le domaine des RdP [DAV 92].

Définition 4.4

Soient G_1 et G_2 deux Grafcets ayant un ensemble $\{<T_1^1, T_2^1>, \ldots, <T_1^r, T_2^r>\}$ de couples de transitions portant le même événement, c'est-à-dire. à chaque couple de transitions $<T_1^i, T_2^i>$ où $T_1^i \in G_1$ et $T_2^i \in G_2$, est associé le même événement e^i, $i \in [1, r]$.

La synchronisation de G_1 et G_2 consiste à construire un Grafcet G obtenu en remplaçant le couple $<T_1^i, T_2^i>$ par une seule transition T^i. L'événement e^i étant associé à cette transition.

❑

Cette opération de synchronisation est illustrée dans la figure 4.5.

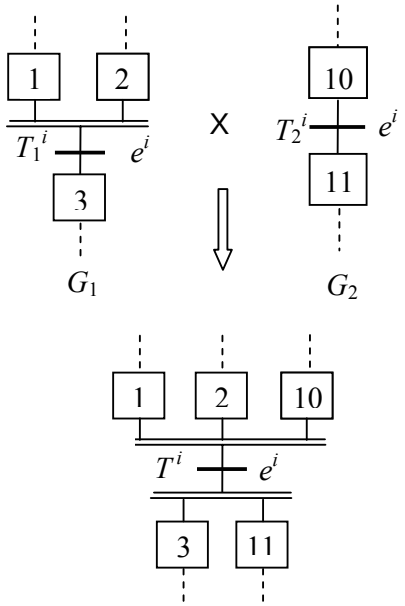


Figure 4.5. Synchronisation de deux Grafcets

4.2.3.2 Grafcet du fonctionnement en boucle fermée

Le modèle Grafcet du fonctionnement en boucle fermée est obtenu par la synchronisation de Grafcet du procédé étendu G_P avec celui de la spécification G_S.

Définition 4.5

On appelle modèle quasi-Grafcet du fonctionnement en boucle fermée le modèle Grafcet qui représente le fonctionnement désiré obtenu par la synchronisation de G_P et G_S. Ce modèle sera noté G_d.

❑

Remarque 4.1 : Nous avons appelé le modèle G_d quasi-Grafcet parce qu'avant d'étudier la contrôlabilité, nous ne savons pas à l'avance si ce modèle est le Grafcet du contrôleur ou non. Pour étudier le problème de la contrôlabilité, nous allons être amenés à faire évoluer le grafcet du fonctionnement désiré en boucle fermée selon des règles qui ne lui sont pas propres. Il est bien entendu que le contrôleur qui sera implanté respectera totalement les règles du Grafcet.

❑

Pour l'exemple du système manufacturier, le Grafcet du fonctionnement en boucle fermée G_d est obtenu par la synchronisation du Grafcet du procédé étendu G_p avec celui de la spécification G_s (Figure 4.6). Les seuls événements communs sont t_1 et t_2.

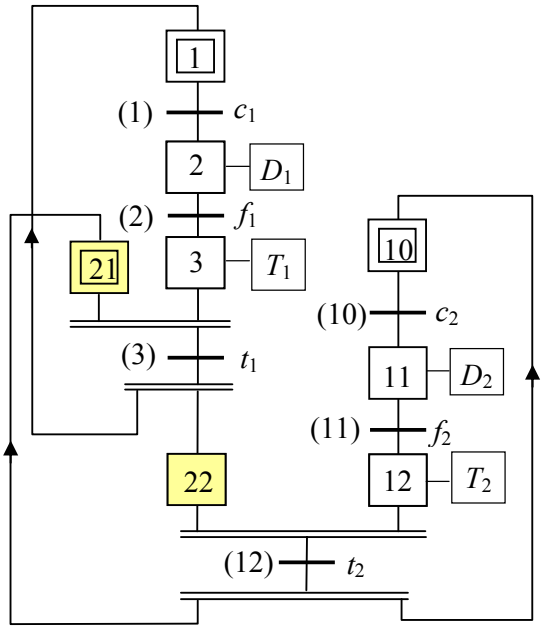


Figure 4.6. Grafcet du fonctionnement en boucle fermée G_d.

A cause des transitions incontrôlables, le modèle quasi-Grafcet précédent ne respecte pas totalement les règles du Grafcet, où il existe des transitions franchissables même si toutes ses étapes d'entrée ne sont pas actives. En effet si une transition incontrôlable est validée dans le modèle du procédé, alors elle sera franchie à l'occurrence de l'événement incontrôlable. Ce franchissement aura lieu quel que soit la situation atteinte dans la grafcet des spécifications. Cela nous amène à apporter les définitions suivantes concernant ce modèle.

Définition 4.6

Une transition incontrôlable est validée dans le modèle quasi-Grafcet si toutes ses étapes d'entrée qui appartiennent au Grafcet du Procédé G_P sont actives.

❑

Dans la définition ci-dessus, l'ensemble des étapes est divisé en deux parties : les étapes appartenant au grafcet du procédé G_P et les étapes appartenant au grafcet des spécifications G_s. Ces deux ensembles d'étapes n'ont pas un comportement symétrique comme cela est stipulé dans la définition 4.6. Le franchissement d'une transition incontrôlable permet de déterminer les situations interdites.

Définition 4.7

Si une situation est telle qu'une transition T_j incontrôlable est validée pour le modèle quasi-grafcet mais pas pour le modèle grafcet, alors cette situation est interdite.

❑

Par exemple dans la figure 4.6 avec la situation (1, 12, 21), à l'occurrence de l'événement incontrôlable t_2 la transition (12) est franchie. Cela signifie que la situation interdite (1, 12, 21) est interdite.

Cette approche fournit une alternative à la théorie R&W pour la détermination des situations interdites. Elle est plus efficace dans la mesure où l'analyse se fait directement sur la structure grafcet.

Le modèle grafcet du fonctionnement désiré en boucle fermée constitue la structure de base du contrôleur. Pour contrôler ce modèle et lui interdire d'évoluer vers les situations interdites il faut d'abord trouver toutes les situations atteignables qui ne sont pas désirées.

4.3 Détermination de l'ensemble des états interdits

La détermination des états interdits se fait par l'étude de la contrôlabilité du langage de spécification par rapport au langage généré par le modèle du comportement en boucle fermée. Dans notre approche la détermination des états interdits ne nécessite pas de revenir à la théorie de R&W. Nous allons construire deux automates : le premier décrit **le fonctionnement désiré** normal de modèle quasi-Grafcet c'est-à-dire qui respecte les règles normales d'évolution du Grafcet, et le deuxième décrit le **fonctionnement possible** en présence d'une transition incontrôlable selon la définition 4.7. Nous allons donner dans la suite les définitions de chaque automate et nous discutons comment trouver l'ensemble des états interdits à partir de ces automates.

4.3.1 Graphe des situations du fonctionnement en boucle fermée

Le graphe des situations est un outil d'identification du comportement du Grafcet. L'intérêt d'élaborer le graphe des situations est d'étudier la contrôlabilité du langage de la spécification par rapport au langage généré par l'automate correspondant au Grafcet $G_{d.}$

Définition 4.8

L'automate qui décrit de manière exhaustive l'ensemble des évolutions possibles du Grafcet G_d est appelé graphe des situations, ce modèle sera noté S_d.

❑

Le graphe des situations S_d correspond à une machine d'état classique $S_d = \{S, \Sigma, \delta, s_0\}$, où S est l'ensemble des états, $\Sigma = \Sigma_c \cup \Sigma_u$ est l'alphabet des événements, δ est la fonction de transition d'états et s_0 est l'état initial.

Les sommets du graphe des situations d'un Grafcet représentent les situations atteignables et les arcs caractérisent les passages possibles entre ces situations. Les principales difficultés associées à l'élaboration d'un graphe des situations sont relatives à l'explosion combinatoire du nombre de situations accessibles, au comportement temporel du Grafcet et aux difficultés d'interprétation du Grafcet.

Une situation correspond à une combinaison d'étapes actives, il existe alors au maximum, pour un Grafcet qui possède N étapes 2^N situations possibles.

La figure 4.7 donne le graphe des situations S_d du Grafcet G_d de l'exemple.

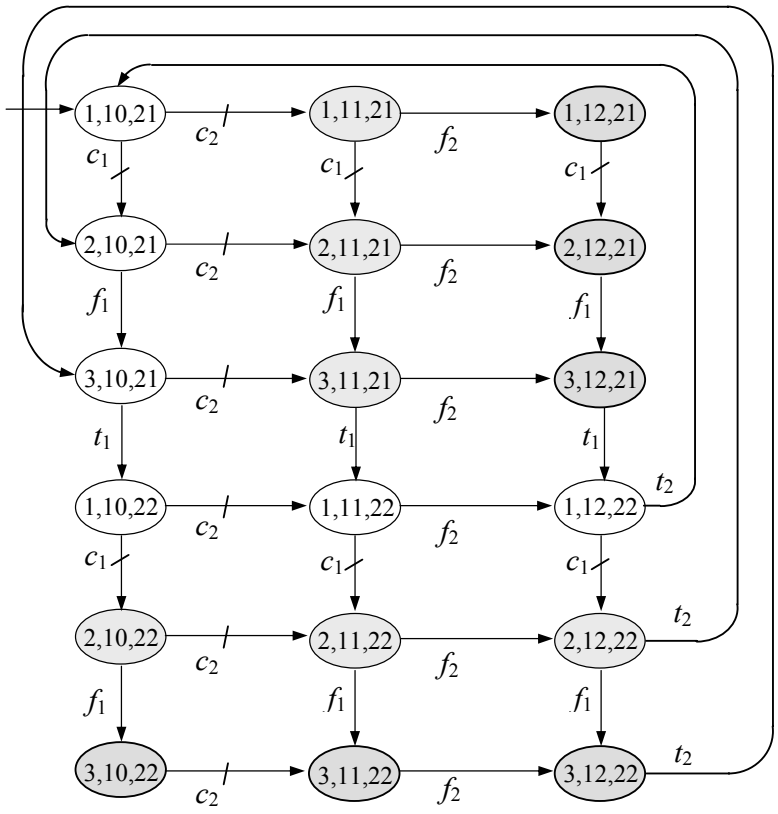


Figure 4.7. Graphe des situations du fonctionnement désiré en boucle fermée S_d.

A partir du graphe des situations donnant le fonctionnement désiré en boucle fermée, la théorie développée par Ramadge et Wonham permet d'obtenir le superviseur le plus permissif. Si le langage généré par le graphe des situations est contrôlable, alors le problème

est résolu et le Grafcet G_d constitue le contrôleur. Dans le cas contraire, il faut déterminer l'ensemble de tous les états interdits.

L'ensemble des transitions dans S_d est subdivisé en deux sous-ensembles disjoints; T_c, l'ensemble des transitions contrôlables et T_u celui des transitions non contrôlables, tels que $T = T_c \cup T_u$. Par conséquent, pour empêcher le système d'atteindre les états interdits, il faut interdire tous les états à partir desquels le système peut évoluer par des transitions non contrôlables vers des états interdits.

Dans le Grafcet de fonctionnement en boucle fermée de notre exemple, nous avons :

$T_c = \{c_1, c_2\}$, $T_u = \{f_1, t_1, f_2, t_2\}$. Nous allons construire maintenant le graphe des situations qui décrit le fonctionnement possible de modèle quasi-Grafcet où le franchissement des transitions incontrôlables se fait selon la définition 4.7. Cet automate sera noté S'_d.

La figure 4.8 donne le graphe des situations S'_d du Grafcet G_d de l'exemple.

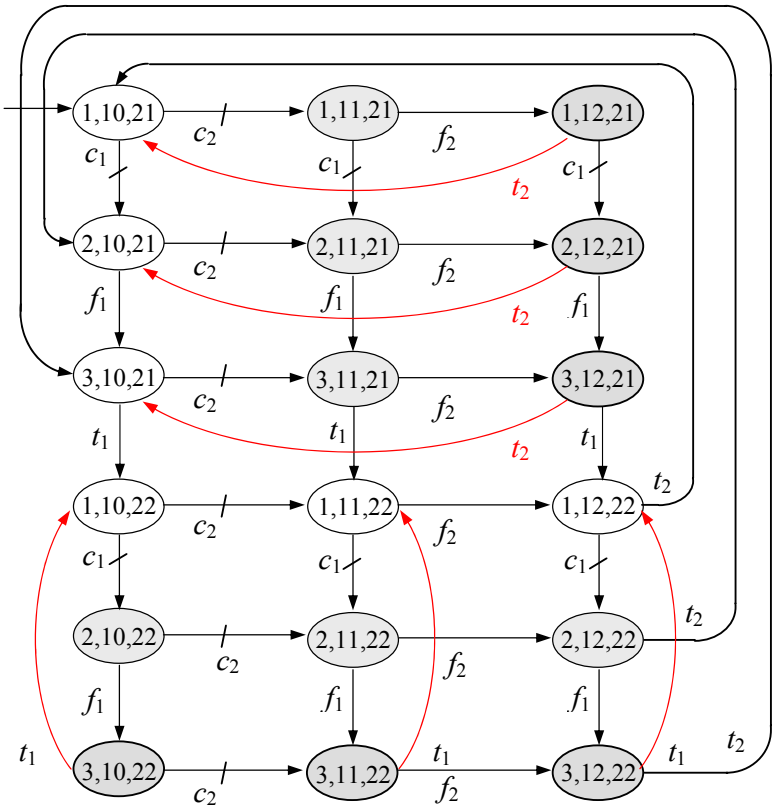


Figure 4.8 Graphe des situations du fonctionnement possible en boucle fermée S'_d.

Propriété 4.1 :

a) Le langage généré par le grafcet correspondant au fonctionnement désiré en boucle fermée est contrôlable si les deux graphes des situations S_d et S'_d sont identiques.

b) Si le graphe S_d est strictement inclus dans S'_d , alors l'ensemble des états et des transitions qui sont dans S'_d mais pas dans S_d correspondent à des états interdits et à des transitions interdites.

❑

Nous remarquons que le graphe des situations S'_d (figure 4.8) contient plus de transitions que celui S_d (figure 4.7). Les transitions supplémentaires représentent les transitions incontrôlables qui sont validées dans le procédé et ne le sont pas par les spécifications. Tout état dans la figure 4.8 qui a une transition de sortie qui n'existe par dans la figure 4.7 correspond à un état interdit (définition 4.7).

Par exemple, l'état (1,12, 21) a une transition de sortie t_2 qui n'existe pas à partir de cet état dans la figure 4.7, donc il est interdit. Par la même technique, nous déterminons l'ensemble des états interdits.

4.3.2 Ensemble des états interdits

Définition 4.9

Soit S l'ensemble des états dans le graphe des situations S'_d. Une spécification de commande de type état interdit est une fonction linéaire $Y : S \rightarrow \{0,1\}$. L'ensemble des états interdits E est celui pour lequel Y est violée. Il est donné par :

$$E = \{e \in S \mid Y(S) = 0\} \subseteq S.$$

❑

Par la technique présentée ci-dessus, on trouve l'ensemble des états interdits suivant :

$E = \{(1, 12, 21), (2, 12, 21), (3, 12, 21), (3, 10, 22), (3, 11, 22), (3, 12, 22)\}$.

Définition 4.10

On appelle un état faiblement interdit dans le graphe de situation S_d tout état à partir duquel un état interdit peut être atteint par l'occurrence d'une séquence de transitions incontrôlables. L'ensemble de ces états sera noté H. Formellement, l'ensemble des états faiblement interdits est représenté par :

$$H = \{ h \in S \mid \exists\, e \in E \wedge \sigma \in \Sigma_U^* \,,\, \delta(h,\sigma) = e \}.$$

❑

La figure 4.10 représente la notion de l'état faiblement interdit

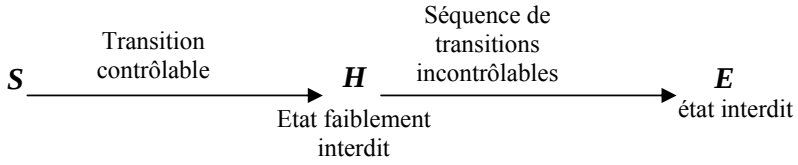


Figure 4.9 Etat faiblement interdit.

Par exemple dans le graphe de la figure 4.7 l'état (2, 10, 22) est un état faiblement interdit parce qu'à partir de cet état on peut atteindre l'état interdit (3, 10, 22) par l'occurrence de l'événement incontrôlable f_1. Ainsi on peut déterminer l'ensemble *H* suivant :

H= {(1, 11, 21), (2, 11, 21), (3, 11, 21), (2, 10, 22), (2, 11, 22), (2, 12, 22)}

Définition 4.11

L'ensemble des états formé de l'ensemble des états interdit et des états faiblement interdits est appelé l'ensemble des états dangereux et cet ensemble sera noté *D*.

$$D = E \cup H$$

❑

Ainsi l'ensemble des états dangereux (les états interdits et les états faiblement interdits) est :

D = {(1, 11, 21), (2, 11, 21), (3, 11, 21), (2, 10, 22), (2, 11, 22), (2, 12, 22), (1, 12, 21), (2, 12, 21), (3, 12, 21), (3, 10, 22), (3, 11, 22), (3, 12, 22)}.

Définition 4.12

Soit *F* l'ensemble des états dangereux tels que :

$$F = \{ f \in D \mid \forall\, s \in S \backslash D \text{ et } \delta(s,\sigma) = f,\ \sigma \in \Sigma_c \}$$

F correspond à l'ensemble des états interdits tel que les événements associés aux arcs arrivant dans ces états soient tous contrôlables. C'est l'ensemble des états frontières.

❑

Dans le cas de l'exemple, nous obtenons l'ensemble :

F = {(1, 11, 21), (2, 11, 21), (3, 11, 21), (2, 10, 22), (2, 11, 22), (2, 12, 22)} (4.1)

Notre objectif est de construire un contrôleur tel que les états de l'ensemble F ne soient jamais atteignables.

Définition 4.13

On appelle un état autorisé tout état qui appartient à S_d et n'appartient pas à D. Cet ensemble sera noté $S_a = S_d - D$

❑

Définition 4.14

Soit T_{cF} l'ensembles des transitions tel que :

$T_{cF} = \{T_i \in T_c \mid \exists (s, f) \in S_a \times F$ et $\delta (s, T_i) = f\}$. C'est l'ensemble des transitions frontières.

❑

Nous pouvons maintenant définir le contrôleur optimal.

Définition 4.15

Un contrôleur est maximal permissif si, tout état admissible de S_a est atteignable sous contrôle et toutes les transitions d'états de T_{cF} sont interdites. Un tel contrôleur est dit optimal.

❑

Pour utiliser l'approche basée sur les invariants de marquages il est nécessaire de construire un ensemble de contraintes linéaires qui sera déterminé à partir de l'ensemble F. Cet ensemble doit être calculé d'une façon systématique.

4.4 Des états interdits aux contraintes linéaires

Nous allons montrer ci-dessous comment construire de manière systématique un ensemble de contraintes linéaires équivalent à l'ensemble des états interdits.

Un état du Grafcet correspond à un ensemble d'étapes actives, le mot *active* signifie que la valeur booléenne correspondante est égale à 1. Soit f_i un état interdit de l'ensemble F et soit $\{X_{i1}, X_{i2}, \ldots, X_{ir}\}$ l'ensemble des étapes actives qui lui correspondent. On notera :

$$f_i = \{X_{i1}, X_{i2}, \ldots, X_{ir}\}$$

Propriété 4.2 :

Soient V_1 et V_2 deux états atteignables tels que $V_1 \supseteq V_2$[1].

Si V_2 est un état interdit, alors V_1 est également un état interdit.

Preuve :

Soient V_1 et V_2 deux états atteignables. L'inclusion $V_1 \supseteq V_2$ signifie que l'ensemble des étapes actives de V_2 est inclus dans celui de V_1. Dans le Grafcet, une transition est validée si toutes ses étapes d'entrée sont actives. Nous en déduisons que toute transition incontrôlable qui est validée par V_2 est également validée par V_1. Avec la connaissance que V_2 est un état interdit, donc V_1 est également un état interdit.

❑

Généralement dans un modèle Grafcet, l'ensemble d'états ordonnés par la relation d'inclusion se compose d'un élément seulement. Les modèles Grafcet habituels sont bornés. Si ce n'est pas le cas, l'état interdit considéré sera le plus petit vecteur de cet ensemble. Ceci permet de réduire la taille de l'ensemble *F*. Prenons le Grafcet de la figure 4.10 suivant :

[1] La relation ordre $V_1 \supseteq V_2$ signifie que : si $V_1 = \{X_{11}, X_{12}, \ldots, X_{1n}\}$ et $V_2 = \{X_{21}, X_{22}, \ldots, X_{2n}\}$, donc $\forall\ i \in [1, n]$, nous avons toujours $X_{1i} \geq X_{2i}$, où n est le nombre des étapes grafcet.

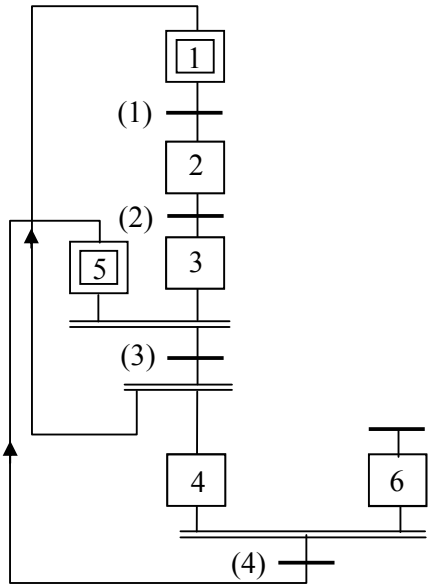


Figure 4.10

Si l'état (X_3, X_4) est interdit donc l'état (X_3, X_4, X_6) est également interdit.

Corollaire 1 : Soit F' l'ensemble réduit de l'ensemble F, modulo la relation d'ordre $\supseteq$.

❑

Propriété 4.3 :

Soit $f_i = \{X_{i1}, X_{i2}, \ldots, X_{ir}\}$ un état interdit quelconque de F. Il y a une équivalence entre l'ensemble des situations autorisées S_a et l'ensemble de celles définies par la contrainte linéaire :

$$\sum_{k=1}^{r} X_{ik} \leq r - 1 \qquad (4.2)$$

Où r est le nombre des étapes actives de f_i, et les indices ik sont les numéros de ces étapes.

Preuve :

Dans la démonstration, ne sont comparées que les r étapes qui sont actives dans f_i.

***Condition nécessaire* :** Soit $V = \{Y_1, Y_2, \ldots, Y_n\}$ un état atteignable quelconque de S_a.

Nous n'avons pas ($V \supseteq f_i$), sinon V sera un état interdit selon la propriété 4.2.

Donc, $\exists\ _{ik} \in [1, r]$, tel que $Y_{ik} < X_{ik}$. Et $Y_{ik} = 0$, la relation (4.2) est vérifiée.

Condition suffisante **:** Soit $V = \{Y_1, Y_2, \ldots, Y_n\}$, un état quelconque qui vérifie la relation (4.2). Comme les variables Y_i sont Booléennes, donc nous n'avons pas ($V \supseteq f_i$), et V est un état autorisé. (f_i est le plus petit vecteur interdit).

❑

L'ensemble des états interdits f_i divise l'espace d'état par deux sous-ensembles disjoints : 1) S_a, l'ensemble des états autorisés, et 2) S_f, l'ensemble d'états interdits qui sont supérieurs ou égaux aux vecteurs f_i. L'ensemble de marquages autorisés correspond à un espace d'état défini par des hyper plans. Chaque hyperplan correspond à une contrainte linéaire.

En appliquant la propriété 4.3 à l'ensemble F des états interdits, on trouve l'ensemble des contraintes linéaires. Chaque contrainte imposée sera représentée dans le Grafcet de contrôleur par une étape. Donc la dimension (nombre des étapes) du Grafcet du contrôleur est égale au nombre des contraintes imposées.

Reprenons l'exemple du système manufacturier et considérons l'ensemble des états interdits donné par la relation (4.1). La propriété 4.3 permet de déterminer les six contraintes linéaires suivantes :

$$X_1 + X_{11} + X_{21} \leq 2$$
$$X_2 + X_{11} + X_{21} \leq 2$$
$$X_3 + X_{11} + X_{21} \leq 2$$
$$X_2 + X_{10} + X_{22} \leq 2$$
$$X_2 + X_{11} + X_{22} \leq 2$$
$$X_2 + X_{12} + X_{22} \leq 2$$

Dans cet exemple, $F = H$. Chaque contrainte va être associée à une étape de contrôle. C'est-à-dire que nous avons besoin d'ajouter 6 étapes de contrôle. Dans certain cas, grâce à la propriété d'invariants de marquages bien connus dans la théorie des RdP, il est possible de réduire cet ensemble de contraintes.

Pour notre exemple, en additionnant les trois premières contraintes, nous trouvons :

$$\left.\begin{array}{l} X_1 + X_{11} + X_{21} \leq 2 \\ X_2 + X_{11} + X_{21} \leq 2 \\ X_3 + X_{11} + X_{21} \leq 2 \end{array}\right\} = (X_1 + X_2 + X_3) + 3\,X_{11} + 3\,X_{21} \leq 6$$

En utilisant l'invariant : $X_1 + X_2 + X_3 = 1$ nous trouvons :

$$1 + 3\,(X_{11} + X_{21}) \le 6 \quad \Leftrightarrow \quad 3\,(X_{11} + X_{21}) \le 5 \quad \Leftrightarrow \quad X_{11} + X_{21} \le 1$$

Donc nous pouvons remplacer ces trois contraintes par une contrainte plus simple équivalente :

$$X_{11} + X_{21} \le 1 \qquad (4.3)$$

La contrainte $X_{11} + X_{21} \le 1$ signifie qu'il ne faut pas que les étapes 11 et 21 de la figure 4.6, soient actives en même temps. Nous remarquons que la boucle qui contient les étapes X_1 , X_2 , X_3 est indépendante des étapes 11 et 21 c'est-à-dire quel que soient les étapes actives dans la boucle de l'invariant X_1 , X_2 , X_3 la contrainte (4.3) reste toujours maintenue. Donc les trois contraintes sont bien équivalentes à une seule contrainte.

De la même manière, on trouve pour les trois autres contraintes :

$$\left.\begin{array}{l} X_2 + X_{10} + X_{22} \le 2 \\ X_2 + X_{11} + X_{22} \le 2 \\ X_2 + X_{12} + X_{22} \le 2 \end{array}\right\} \quad = 3\,X_2 + (X_{10} + X_{11} + X_{12}) + 3\,X_{22} \le 6$$

En utilisant l'invariant : $X_{10} + X_{11} + X_{12} = 1$ nous trouvons :

$$1 + 3\,(X_2 + X_{22}) \le 6 \quad \Leftrightarrow \quad 3\,(X_2 + X_{22}) \le 5 \quad \Leftrightarrow \quad X_2 + X_{22} \le 1$$

Ici aussi nous pouvons remplacer ces trois contraintes par une contrainte plus simple équivalente :

$$X_2 + X_{22} \le 1 \qquad (4.4)$$

Il est évident que cette technique de réduction n'est pas générale. Elle dépend des propriétés de modèle, c'est-à-dire de l'existence d'invariants. Si de tels invariants n'existent pas, alors le contrôleur peut être déterminé avec le premier ensemble de contraintes et le contrôleur final contiendra plus d'étapes de contrôle. Mais le comportement en boucle fermée sera identique.

A cette étape de notre approche, l'ensemble des états interdits est caractérisé de manière bijective par un ensemble de contraintes. La dernière étape consiste à synthétiser des étapes de contrôle qui respectent ces contraintes.

4.5 Synthèse de superviseur

Nous avons présenté dans le deuxième chapitre comment les invariants de marquage peuvent être utilisés pour calculer simplement le contrôleur d'un procédé modélisé par un

RdP. Dans notre travail nous allons adopter cette méthode de synthèse dans le cas du Grafcet. La spécificité de notre outil vient du fait que les marquages sont Booléens.

Nous illustrerons notre approche dans le cas de l'exemple manufacturier, puis nous mettrons en évidence les problèmes posés par le Grafcet.

Calcul des étapes de contrôle à ajouter

Soit W_p[2] la matrice d'incidence du Grafcet S_d, correspondant au fonctionnement désiré en boucle fermée. L'approche de synthèse consiste à compléter cette matrice d'incidence par la partie W_c correspondant aux étapes de contrôle. Soit W la matrice d'incidence du système contrôlé, chaque place du contrôleur va ajouter une ligne à la matrice d'incidence de procédé contrôlé W. Cette matrice sera composée de deux matrices, la matrice originelle du modèle de procédé W_p et la matrice d'incidence du contrôleur W_c :

$$W = \begin{bmatrix} W_p \\ W_c \end{bmatrix} \qquad (4.5)$$

Dans le modèle Grafcet, nous n'avons pas de marquage mais des étapes actives ou non actives, dans le cas général, l'ensemble des contraintes linéaires peut s'écrire sous la forme matricielle suivante :

$$L.X_P \leq b \qquad (4.6)$$

Le principe du contrôle consiste à créer des invariants en ajoutant des étapes :

$$L.X_P + X_c = b \qquad (4.7)$$

On en déduit de manière simple la partie de la matrice d'incidence qui correspond à la partie contrôle et le marquage des étapes initiales :

$$W_c = - L.W_P \qquad (4.8)$$

et

$$X_{cinit} = b - L.X_{pinit} \qquad (4.9)$$

---

2 Les indices p et c sont utilisés pour désigner respectivement le procédé et le contrôleur.

Le contrôleur est connu par la détermination de sa matrice d'incidence W_c et son marquage initial $X_{cinit.}$

Nous obtenons ainsi une solution structurelle facile à mettre en œuvre.

Nous avons vu dans le deuxième chapitre le problème des transitions incontrôlables (c'est le cas de notre exemple) pour lesquelles il faut garantir qu'il n'y a pas d'étapes de contrôle en entrée. Dans notre approche, le calcul de la matrice [- $L.W_U$] où W_U est la matrice d'incidence de la partie incontrôlable est inutile. En effet, dans la synthèse du contrôleur nous avons calculé les étapes de contrôle à partir des états interdits frontières où toutes les transitions d'entrées à ces états sont contrôlables c'est pourquoi la détermination de l'ensemble des transitions T_{cF} ainsi que les états interdits frontaliers F est une étape très importante dans la synthèse du contrôleur.

Résultat fondamental **:** La solution obtenue par notre approche donne le contrôleur optimal. Cette optimalité vient de l'équivalence entre l'ensemble des situations autorisées S_a et l'ensemble des contraintes linéaires.

Remarque 4.2 : L'équivalence entre les réseaux de Petri interprétés de commande et le Grafcet, nous permet d'appliquer les résultats existant pour les RdP au Grafcet. Il faut prouver pour cela que les grafcets modélisés sont des grafcets sains. Pour cela il faut que les 3 points soient vérifiés [DAV92] :

1) A aucun moment une étape active ne peut être telle que : une de ses transitions d'entrée est franchissable, et aucune de ses transitions de sortie n'est pas simultanément franchissable (on dit que le grafcet n'est pas réactivable).

2) Pour chaque conflit réel $<P_1, \{T_1, T_1, \ldots\}\ldots>$ qui peut exister, il y a des arcs $T_1 \rightarrow P_1, T_2 \rightarrow P_1 \ldots$

3) Aucune étape ne peut avoir deux transitions d'entrées simultanément franchissable (sauf le cas 2 qui précède).

❑

Les grafcets de notre exemple sont des grafcets sains.

Appliquons ce résultat à notre exemple.

Soit W_p la matrice d'incidence du Grafcet G_d donné dans la figure 4.6 :

$$W_p = \begin{array}{c} \begin{matrix} c_1 & f_1 & t_1 & c_2 & f_2 & t_2 \end{matrix} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 10 \\ 11 \\ 12 \\ 21 \\ 22 \end{matrix}$$

Nous avons deux contraintes donc la matrice de contrainte L contient deux lignes, à partir des contraintes (4.3) et (4.4), on détermine cette matrice L :

$$L = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 10 & 11 & 12 & 21 & 22 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

La matrice d'incidence de la partie contrôlée est alors :

$$W_c = - L.W_P \Rightarrow \quad W_c = \begin{bmatrix} -1 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix}$$

Les deux étapes de contrôle P_{c1} et P_{c2} à ajouter ont pour marquage initial :

$X_{c1} = 1$ et $X_{c2} = 0$.

A ce niveau, la synthèse de contrôleur est terminée et nous avons trouvé le contrôleur optimal. Il faut noter qu'il y a des transitions de sortie incontrôlables associées aux places de contrôle. Mais, comme nous l'avons dit, cela ne conduit jamais à un mauvais fonctionnement, c'est-à-dire quand l'étape de contrôle n'est pas active, aucune autre étape d'entrée de cette transition incontrôlable n'est active. Cette propriété est garantie grâce à la détermination de l'ensemble d'états interdits frontières F.

Le Grafcet du contrôleur final est représenté dans la figure 4.12. Dans ce modèle nous pouvons remarquer que les événements contrôlables ont disparu. L'autorisation pour faire l'action est déterminée par l'activation de la place de contrôle. Ce contrôleur peut être directement implémenté dans un automate programmable industriel. C'est un avantage important du Grafcet. On s'affranchit ainsi d'une transformation du modèle synthétisé vers le modèle implantable, transformation qui est souvent source d'erreurs.

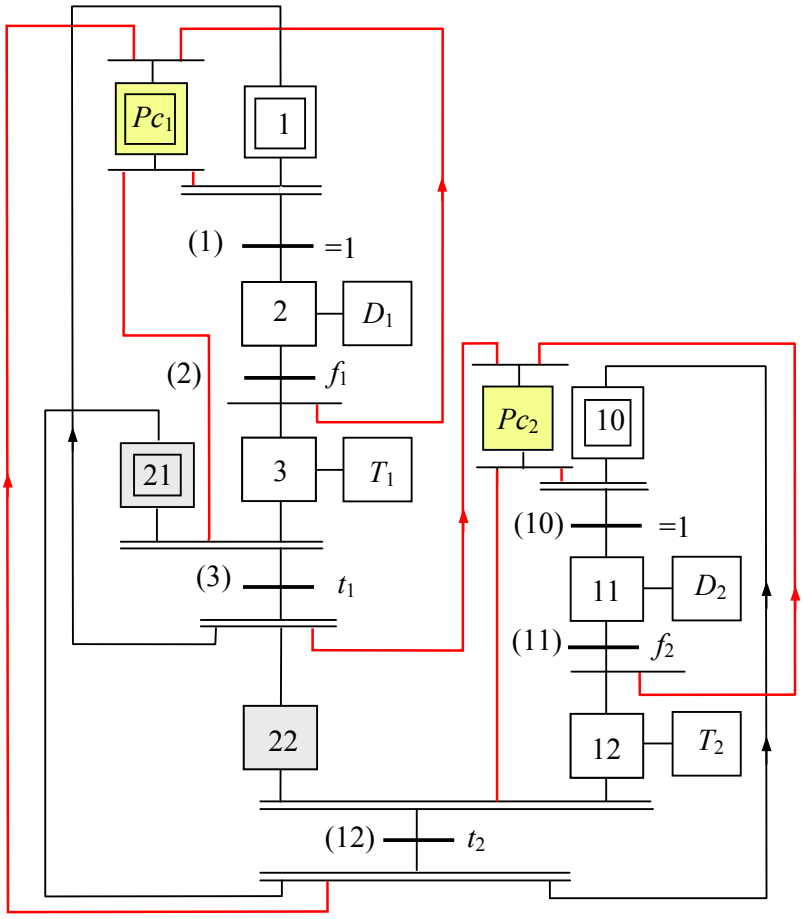


Figure 4.11 Grafcet du contrôleur du système manufacturier

Remarque 4.3 : Il peut arriver que le marquage initial de l'étape de contrôle calculé par l'équation (4.9) soit plus grand que 1. La transformation en Grafcet pose alors un problème puisque le marquage est booléen, On utilise alors un compteur pour représenter le nombre d'activations de l'étape de contrôle. L'idée de compteur est illustrée dans la figure 4.12.

Quand le marquage de l'étape de contrôle $X_{cinit} \geqslant 1$, l'étape 2 sera active initialement ($X_2 = 1$) et la valeur du compteur C sera égale à $C_0 = X_{cinit}$.

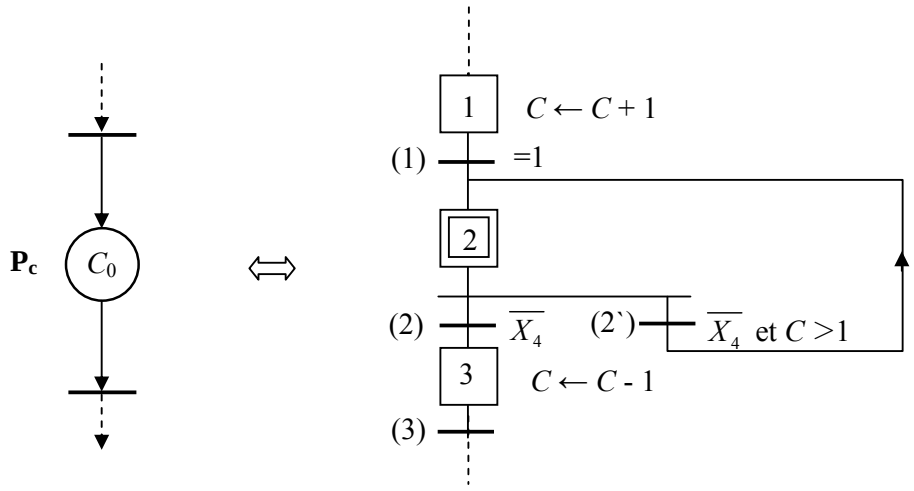


Figure 4.12

La réactivation de l'étape 2 par la transition (2`) permet d'envoyer le nombre d'activations du Grafcet en aval en un nombre égal au marquage X_{cini}. Les actions impulsionnelles $C \leftarrow C+1$ et $C \leftarrow C-1$ permettent de réaliser l'invariant de l'étape de contrôle.

Supposons par exemple que $X_{cinit} = 2$, dans la figure 4.12 l'étape 2 est active à l'initialisation, et la valeur initiale du compteur $C_0 = 2$. Les transitions (2) et (2`) sont simultanément franchies (l'étape 3 n'est pas active et $C > 1$). Les conséquences de ces 2 franchissements sont : 1) activation de l'étape 3 et décrémentation du compteur C, et 2) l'étape 2 reste active. Dès que l'étape 3 est désactivée, il y a à nouveau le franchissement de la transition 2, mais cette fois-ci seule. Le nombre d'activations donné par C_0 a été atteint. Il fait attendre la prochaine incrémentation du compteur lors de l'activation de l'étape 1.

4.6 Conclusion

Dans ce chapitre nous avons présenté une approche de synthèse d'un contrôleur basée sur le Grafcet. Après avoir modélisé le procédé à commander ainsi que les spécifications par des modèles Grafcets, nous avons synchronisé les deux Grafcets pour avoir un modèle Grafcet de fonctionnement en boucle fermée que nous avons appelé quasi-Grafcet. Ce modèle constitue la structure de base du contrôleur. Le fait d'avoir des transitions incontrôlables dans ce modèle peut conduire vers des situations interdites. Le concept du modèle quasi-grafcet a permis de déterminer de manière élégante l'ensemble des situations interdites, en s'affranchissant de la théorie R&W.

L'approche basée sur les invariants de marquage des RdP a été utilisée pour faire la synthèse de contrôleur basée sur le Grafcet. L'idée principale est de trouver des étapes de

contrôle qui permettent d'interdire l'accès aux situations interdites. Cette approche exige que les contraintes soient données comme des inégalités linéaires. Cela nous a amené à établir des propriétés générales qui ont permis de trouver l'ensemble des contraintes linéaires à partir de l'ensemble des états interdits. La dimension de contrôleur est égale aux nombre des contraintes. Cet ensemble peut être réduit dans certains cas grâce à la propriété d'invariants de marquages.

Par la suite les étapes de contrôle ont été aisément calculées. Cela nous a conduit a faire les remarques importantes suivantes :

- Le problème posé par les transitions incontrôlables en aval des places de contrôle n'apparaît plus dans notre solution car les situations interdites utilisées dans la synthèse sont des situations frontières. Celles-ci sont toujours atteignables à l'occurrence d'événements contrôlables.

- La solution obtenue donne un contrôleur optimal. Cette optimalité vient de l'équivalence entre l'ensemble des situations autorisées S_a et l'ensemble des contraintes linéaires.

Conclusion et perspectives

CONCLUSION ET PERSPECTIVES

La synthèse de contrôleurs pour les systèmes à événements discrets a été largement étudiée dans les travaux de Ramadge et Wonham. Dans leur approche, le système à événements discrets à contrôler est modélisé par un automate et son comportement par un langage formel. Comme nous l'avons vu dans la présentation de cette approche le grand nombre d'états à considérer pour représenter le comportement du système ainsi que le manque de structure dans les modèles limite les possibilités de développer des algorithmes de calcul efficace pour l'analyse et la synthèse.

L'approche classique utilise un point de vue centralisé. Les modèles obtenus sont de grande taille. Différents travaux proposent des approches permettant de réduire cette taille par la remise en cause du point de vue centralisé : point de vue hiérarchique, modulaire sous observation partielle ou décentralisé. Une fois la commande obtenue, une interprétation est nécessaire. Là aussi, différents travaux proposent des interprétations sous forme de schémas Ladder ou de Grafcet.

Nous avons présenté dans cette thèse une contribution au problème de la synthèse de contrôleur pour les systèmes à événements discrets modélisés par des Grafcets.

La première méthode de synthèse est une suite du travail de François Charbonnier sur la commande supervisée, il a montré que si le langage des spécifications est contrôlable par rapport au langage du procédé étendu, le Grafcet de spécification lui-même constitue le Grafcet de superviseur. Mais dans le cas où le langage des spécifications est non contrôlable, il faut chercher le plus grand langage contrôlable. L'algorithme de Kumar permet de trouver l'automate superviseur qui reconnaît ce langage. Pour implanter l'automate superviseur obtenu par l'algorithme de Kumar, il faut revenir vers le Grafcet. Pour cela nous avons proposé une méthode systématique de passage de l'automate superviseur vers le Grafcet superviseur de manière structurelle. Cependant dans cette approche, un inconvénient majeur subsiste : Le Grafcet final de superviseur obtenu par cette approche contient un nombre important d'étapes qui est identique à l'automate synthétisé. Donc la complexité des superviseurs que l'on obtient dans le cas d'applications complexes reste toujours prohibitive. Pour éliminer ces difficultés d'implantation, nous avons cherché une autre solution.

La deuxième méthode de synthèse est basée sur l'utilisation du modèle Grafcet du système à contrôler et des spécifications, auxquels des invariants de marquage sont associés, l'objectif étant de construire le modèle Grafcet du système contrôlé. Les invariants de

marquage permettent de déterminer un certain nombre d'étapes à ajouter au modèle initial pour faire respecter les spécifications de commande.

L'avantage pratique de cette méthode est qu'elle utilise directement les modèles Grafcet de commande implantables dans les automates programmables industriels. La méthode comporte deux phases. La première phase consiste à déterminer l'ensemble des états interdits par la construction du graphe des situations de deux modèles Grafcet (fonctionnement désiré et fonctionnement possible). Dans la deuxième phase, nous avons utilisé les invariants de marquage pour construire le Grafcet du contrôleur en déterminant un ensemble d'étapes de contrôle.

Nous avons établi des propriétés générales qui ont permis de trouver l'ensemble des contraintes linéaires à partir de l'ensemble des états interdits. La dimension de contrôleur est en général égale au nombre des contraintes. Cependant, cet ensemble peut être réduit dans certains cas grâce à la propriété d'invariants de marquages exploitée une seconde fois. La solution obtenue donne un contrôleur optimal. Cette optimalité provient de l'équivalence entre l'ensemble des situations autorisées et l'ensemble des contraintes linéaires. Elle a été prouvée grâce au caractère booléen du marquage d'un grafcet.

Nous avons vu que pour appliquer de manière directe la théorie du contrôle par supervision, nous avons considéré la classe de Grafcet où seuls des événements sont associés aux transitions. Cette contrainte provient de la théorie de base sur les langages qui ne considère que les événements. Cependant, Il est connu qu'une réceptivité dans un grafcet peut contenir également des conditions logiques correspondant soit à des entrées provenant de l'environnement, soit à des activités d'étapes. Il est également possible d'avoir des temporisations associées aux étapes et des événements de fin de temporisation associée aux transitons. La synthèse de contrôleur pour les SED temporisés est un thème de recherche très actif dans la communauté automatique. Une perspective importante de mon travail consiste à rechercher une approche de synthèse de contrôleur qui soit structurelle et qui soit utilisable pour un grafcet général. Cette approche doit nécessairement combiner les résultats obtenus dans le domaine des SED autonomes et des SED temporisés.

Références

Références :

[ALL 04] H. Alla, B. Kattan, *Structural Synthesis of a Controller Based on the Grafcet model,* to appear in IEEE Transactions on Control Systems Technology 2004.

[ANT 93] P. Antsaklis, M. Lemmon, J. Stiver, *Hybrid System Modeling and Event Identification*, Technical report, of the ISIS Group, University of Notre Dame, January, 1993.

[ANT 97] P. Antsaklis, editor. *Hybrid Systems IV,* Volume 1273 of lecture Notes in Computer Science. Springer-Verlag, New York, 1997.

[ASA 98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis, *Controller synthesis for timed automata,* Proc. IFAC Symposium on System Structure and Control, pages 469-474, 1998.

[BAD 95] E. Badouel, L. Bernardinello and P. Darondeau, *Polynomial algorithms for the synthesis of bounded nets,* Proc. of CAAP'95 , Springer Verlag LNCS 915 (364-378), 1995.

[BAL 92] S. Balemi, *Control of Discrete Event Systems: Theory and Application*, Thesis for the degree of Doctor of Technical Sciences, Swiss Federal Institute of Technology, Zurich, Mai 1992.

[BAL 93] S. Balemi, G.J. Hoffman, P. Gyugyi, H. Wong-Toi, G.F. Franklin, *Supervisory Control of a Rapid Thermal Multiprocessor*, IEEE Trans. on Automatic Control, Vol. 38, No. 7, pp. 1040-1059, Juillet 1993.

[BAS 98] F. Basile, P. Chiacchio and A. Giua, *Supervisory control of Petri nets based on suboptimal monitor places,* Proc. WODES'98 Sardinia, Italy, pp. 85-87, 1998.

[BLA 79] M. Blanchard, *Comprendre, Maîtriser et appliquer le GRAFCET*. CEPADUES-Editions, Toulouse, 1979.

[BOI 93] O. Boissel, *Optimal feedback control design for discrete-event process systems using simulated annealing,* MS thesis. University of Notre Dam, 1993.

[BRA 89] B. A. Brandin. *Supervisory control of Discrete Event Systems with forcible events,* Thesis for the degree of master of applied science, University of Toronto,1989.

[BRA 93] Y. Brave and M. Heymann. *On optimal attraction in discrete-event processes.* Information sciences, vol. 67, p. 245-276, 1993.

[BRA 94] B. A. Brandin and W. M. Wonham. *Supervisory control of Timed Discrete Event Systems*. IEEE Transactions on Automatic Control,Vol. 39 (2) : 329-342, Fev. 1994.

[CAS 99] C. Cassandras. *Introduction to Discrete Event Systems*. Kluwer Publishing Company 1999.

[CHA 96] F. Charbonnier, *Commande supervisée des systèmes à événements discrets*, thèse Ph.D. thesis (120pp.). INP de Grenoble, France 1996.

[COH 97] G. Cohen, S. Gaubert and J.-P. Quadrat, *Linear projectors in the max-plus algebra*. In: Proc. 5th IEEE Med. Conf. on Cont. and Syst., Paphos, Cyprus. 1997.

[DAV 92] R. David, H. Alla, *Du Grafcet aux réseaux de Petri*, Hermes, Paris, 1992.

[FAB 98] M. Fabian and A. Hellgren. *PLC-based implementation of supervisory control for discrete event systems*. Proc. of the 37th Conf. on Decision and Control. Tampa, USA, 1998.

[FER 99] J. L. Ferrier, *Systèmes dynamiques à événement discrets : du modèle à la commande,* Journées Doctorales d'Automatique JDA'99, Nancy, 1999.

[GAU 95] S. Gaubert, *Performance evaluation of (max,+) automat,* IEEE Transactions on Automatic Control,Vol. 40 (12) : 2014-2025, Dec. 1995.

[GHA 01] A. Ghaffari, N. Rezg, X. Xie, *Conception du Superviseur Optimal Vivant à l'aide de la Théorie des Régions*, MSR'01, Toulouse, France, Hermes (Editeur), Octobre 2001.

[GIU 92] A. Giua, F. DiCesare and M. Silva. *Generalized mutual exclusion constraints on nets with uncontrollable transitions.* In Proc. IEEE International Conference on Systems, Man, and Cybernetics, Chicago, IL, pp. 974-979, 1992.

[GIU 93] A. Giua, F. DiCesare and M. Silva, *Petri net supervisors for generalized mutual exclusion constraints,* Proc. 12th IFAC World Congresse, Sidney, Australia, Juillet, 1993.

[Ho 91] Y. C. Ho and X. Cao. *Perturbation Analysis of Discrete Event Dynamical Systems*. Kluwer Publishing Company, 1991.

[HOL 90] L. E. Holloway and B. H. Krogh, *Synthesis of feedback logic for a class of controlled Petri nets.* IEEE Trans. Autom. Control, AC-35, 5, 514-523, 1990.

[HOL 94] L. E. Holloway and B. H. Krogh. *"Controlled Petri nets: a tutorial survey"*. In G. Cohen and J.-P. Quadrat (Eds), Proc. 11th International Conf. on Analysis and Control, Discrete Event Systems, pp. 158-168. Lecture Notes in Computer Science, Vol. 199, Sprmger-Verlag, Berlin, 1994.

[ICH 87] A. Ichikawa and Iraishi, *analysis and control of discrete-events systems represented by Petri nets,* Discrete Event Systems: Models and Applications, IIASA Conf., Sopron, Hungary, Août 1987.

[KAT 01] B. Kattan, H. Alla, *Commande supervisée des systèmes à événements discrets* , JDA'01 (Journées Doctorales d'Automatique), Toulouse (France), 25-27 Septembre 2001.

[KAT 03] B. Kattan, H. Alla, *Structural Synthesis of a Controller Based on the Grafcet model*, In CESA'2003 - Computing Engineering in Systems Applications, Lille France, 9-11 July 2003.

[KRO 87] B. H. Krogh, *Controlled Petri nets and maximally permissive feedback logic.* Proc. Of the 25th Annual Allerton Conf., University of Illinois, Urbana., 1987.

[KRO 91] B.H. Krogh and L.E. Holloway, *Synthesis of Feedback Control Logic for Discrete Manufacturing Systems*, Automatica, Vol. 27, No 4, pp. 641-651, Juillet 1991.

[KRO 93] B. H. Krogh, *Supervisory control of Petri nets*, Belgian-French-Netherlands, Summer School on Discrete Event Systems, Juin 1993.

[KUM 91] R. Kumar, *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems*, Thesis for the degree of Doctor of Philosophy, Université du Texas, Austin, Août 1991.

[LED 96] R.J.Leduc, *PLC Implementation of a DES Supervisor for a Manufacturing Testbed: An Implementation Perspective*, M.A.Sc. Thesis, Dept. of Elec. and Comp. Eng., University of Toronto, Canada,1996.

[LEP 94] P. Leparc, *Apports de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet*, PhD thesis, Université de Rennes 1, Janvier 1994.

[LES 98] J.-J. Lesage, J.-M. ROUSSEL, J.-M. FAURE, P. LHOSTE, J. ZAYTOON, *Réactivité et déterminisme du comportement temporel du grafcet*, 3ème conférence internationale sur l'Automatisation Des Processus Mixtes, ADPM'98, Reims (France), pp. 99-106, Mars 1998.

[LI 93] Y. Li and W. M. Wonham, *Control of vector discrete-event systems I-The base model*, IEEE Trans. Automatic Control, 38:1214-1227, Août 1993.

[LI 94] Y. Li, W. M. Wonham, *Control of vector discrete-event systems-part ii: controller synthesis.* IEEE Trans. Automatic Control, 39(3):512-531, Mars 1994.

[LIN 90] F. Lin and W. M. Wonham, *Decentralized control and coordination of discret-event systems with partial observation*, IEEE Transactions on Automatic Control, 35(12):1330-1337, December 1990.

[MOO 94] J. O. Moody, M. Lemmon and P. J. Antsaklis, *Supervisory control of Petri nets with uncontrollable/unobservable transitions*, Proc. Of the 35th IEEE CDC, Kobe, Japan, 1996.

[MOO 98] J. O. Moody and P. J. Antsaklis, *Deadlock Avoidance in Petri nets with Uncontrollable Transitions*, Proc. of American Control Conference, Philadelphia, PA, Juin 1998.

[NDJ 99] C. Ndjab, J. Zaytoon, *Methods and tools for the synthesis of an optimal control implementation for Grafcet*, APII-JESA, vol. 33, n° 8-9, p. 1073-1092,1999.

[NOU 97] M. Nourelfath, *Extension de la théorie de la supervision à la commande et à la surveillance des systèmes à événements discrets: application à la sécurité opérationnelle des systèmes de production*. Ph.D. thesis (145pp.). INSA de Lyon, France 1997.

[RAM 82] J.G. Ramadge, W.M. Wonham, *Supervision of Discrete Event Processes*, 21st IEEE Conference on Decision and Control, Orlando, Floride, pp. 1228-1229, Décembre 1982.

[RAM 83] P. J. Ramadge, and W. M. Wonham, *Supervisory control of a class of discrete event processes*, Vol. 63 of LNCIS, pp. 477-498. Springer-Verlag, Berlin, Germany, 1983.

[RAM 86] P. J. Ramadge. *Observability of discrete event systems*. In Proceedings of 1986 IEEE Conference On Decision Control, Pages 1108-1112, Athens, Greece, December 1986.

[RAM 87a] P. J. Ramadge, and W. M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM Joural of Control and Optimization, 25 (1):206-230, Janvier 1987.

[RAM 87b**]** P. J. Ramadge, and W. M. Wonham, *Modular feedback logic for discrete event systems*, SIAM Journal of Control and Optimization, 25 (5):1202-1218, Septembre1987.

[RAM 89] P. J. Ramadge, and W. Wonham, *The Control of Discrete Event Systems*, Proceedings of the IEEE; Special issue on Dynamics of Doscrete Event Systems, Vol. 77, No. 1:81-98, 1989.

[ROU 94] J. M. Roussel, *Analyse de Grafcets par génération logique de l'automate équivalent*, Thèse de doctorat de l'Ecole Normale Supérieure de Cachan,1994.

[SAV 02] A. T. Sava, *Sur la synthèse de la commande des systèmes à événements discrets temporisés*, Ph.D. thesis, INP de Grenoble, France 2002.

[VAL 85] R.Valette, M. Courvoisier, H. Demmou, J. M. Bigou and C. Desclaux, *Putting Petri nets to work for controlling flexible manufacturing systems*, In Proc. ISCAS‘85, Kyoto, pp. 929-932, 1985.

[WON 87] W. M. Wonham and P. J. Ramadge. *On the supremal controllable sublanguage of a given language.* SIAM Journal of Control and Optimization, 25, 637-659, 1987.

[WON 94] W. M. Wonham, *Notes on Control of Discrete-Event Systems*, Technical Report, ECE 1636F/1637S, 1994.

[WON 00] W. M. Wonham, *Supervisory Control of Discrete-Event Systems: An Introduction*, Proc. IEEE Intnl. Conf. on Industrial Technology 2000 (ICIT2000), Goa, India, 474-479, January, 2000.

[WON 01] W. M. Wonham, *Notes on Control of Discrete-Event Systems*, Technical Report, Departement of Electrical and Coputer Engineering, University of Toronto, 2001.

[YAM 91] E.Yamalidou and J. C. Kantor. *"Modelling and optimal control of discrete-event chemical processing using Petri nets"*. Comput. Chem. Engng, 15, 503-519, 1991.

[YAM 94] E.Yamalidou, J. O. Moody, M. D. Lemmon and P. J .Antsaklis. *Feedback control of Petri nets based on place invariants.* Technical Report ISIS-94-002.2, ISIS Group, University of Notre Dame, 1994.

[YAM 96] K. Yamalidou, J. O. Moody, M. Lemmon and P. Antsaklis, *Feedback Control of Petri Nets Based on Place Invariants*, Automatica, Vol. 32, No. 1, pp. 15-28, 1996.

[ZHO 90] H. Zhong and W. M. Wonham. *On the consistency of hierarchical supervision in discrete event systems,* IEEE Transactions on automatic Control, 35(10): 1125-1134, October 1990.

Synthèse structurelle d’un contrôleur basée sur le Grafcet

Résumé :

Nous avons présenté dans cette thèse deux contributions au problème de la synthèse de contrôleur pour les systèmes à événements discrets modélisés par des Grafcets.
La théorie de la supervision des systèmes à événements discrets (SED) initiée par les travaux de Ramadge et Wonham, n'est pas directement implantable sous la forme d'une commande opérationnelle. Nous pouvons trouver dans la littérature diverses extensions de l'approche RW dans lesquelles le superviseur peut forcer des événements, parmi celles-ci il y a la commande supervisée proposée par François charbonnier dans sa thèse (1996). Nous avons fait une extension de cette approche, dans le cas où le langage des spécifications n’est pas contrôlable par rapport au langage du procédé étendu. Pour implanter l’automate de superviseur obtenu, nous avons proposé une méthode systématique de passage de l'automate superviseur vers le Grafcet superviseur de manière structurelle. Cependant dans cette approche, le Grafcet final de superviseur obtenu contient un nombre important d'étapes qui est identique à l'automate synthétisé. Donc la complexité des superviseurs reste toujours prohibitive. C’est pourquoi nous avons proposé une approche de synthèse structurelle, dans laquelle la taille de Grafcet obtenu est réduite et implantable sur les automates programmables. Cette méthode est basée sur les invariants de marquage qui permettent de déterminer un certain nombre d'étapes à ajouter au modèle initial pour faire respecter les spécifications de commande. Nous avons établi des propriétés générales qui ont permis de trouver l’ensemble des contraintes linéaires à partir de l’ensemble des états interdits. La solution obtenue donne un contrôleur optimal. Cette optimalité provient de l’équivalence entre l’ensemble des situations autorisées et l'ensemble des contraintes linéaires. Elle a été prouvée grâce au caractère booléen du marquage d'un Grafcet.

MOTS-CLES : Systèmes à Evénements Discrets, Synthèse d’un Contrôleur, Automates, Grafcet, Langages formels.

Abstract:

In this thesis, we have presented two contributions for the problem of synthesis of a discrete event systems controller using the Grafcet Model.
The supervisory control theory of discrete event systems (DES) introduced by Ramadge and Wonham, is not directly implemented under the form of an operational control. We can find in the literature various extensions of the approach RW in which the supervisor can force events, among those there exist the supervised control suggested by François Charbonnier in his thesis (1996). We have made an extension of this approach, in the case where the language of the specifications is not controllable compared to the language of the extended process. To implement the supervisor's automat obtained, we have proposed a systematic method of passage of the automaton of the supervisor to the Grafcet of the supervisor in a structural way. However in this approach, the final Grafcet of the supervisor obtained contains a significant number of steps that is identical to the synthesized automat. Thus the complexity of the supervisors always remains prohibitive. This is why we have proposed an approach of structural synthesis, in which the size of Grafcet obtained is reduced and directly implemented in the PLC. This method is based on the place invariants which allow determining a certain number of steps to be added to the initial model to make respect the specifications of control. We have established general properties that made it possible to find the set of the linear constraints starting from the set of the forbidden states. The solution obtained gives an optimal controller. This optimality comes from the equivalence between the set of authorized states and the set of linear constraints. It was proven thanks to the Boolean character of the marking of Grafcet.

KEY WORDS : Discrete Event Systems, Synthesis of a controller, Automat, GRAFCET, Formals languages