République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud MAMMERI, Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique
Département d'Automatique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme

Master en Automatique

(Option : Commande des Systèmes)

Thème

SYNTHESE D'UNE COMMANDE SURE POUR LES SYSTEMES A EVENEMENTS DISCRETS.

Proposé et dirigé par :

M^{er}. KARA. Redouane.

Présenté par :

M^{elle}. HADOUCHI. Assia.

Promotion 2011

Remerciments

Nous remercions dieu de nous avoir donné la force et la volonté pour mener à bien notre travail.

Nous tenons à exprimer nos vifs remercîments pour Monsieur Redouane KARA, pour nous avoir proposé ce sujet, pour son aide, conseils et son soutient constant tout au long de la réalisation de ce mémoire.

Nous remercions également les membres de jury qui nous ferons l'honneur d'évaluer ce modeste travail.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail, trouvent ici, l'expression de notre profonde gratitude.

HADOUCHI Assia.

Dédicace

Je dédie cet humble travail à ma très chère mère qui n'as jamais manquée de me soutenir tout au long de mes études, à mon très chère père qui m'as éduqué et soutenu.

A mes frères et mes sœurs.

A mes neveux AMAR et YANIS.

A mon cousin HADOUCHI Achour et sa petite famille.

A mes chers(es) amis(es).

Sommaire

Chapitre IV : Application sur un système industriel

Introduction générale

La théorie des systèmes et de leur commande, appelée aussi l'Automatique, s'est intéressée dès ses origines à des systèmes physiques généralement décrits par les équations différentielles ou aux dérivées partielles auxquelles obéissent les phénomènes physiques correspondant.

L'évolution technologique à conduit au développement de systèmes, dont l'impact socio-économique (en terme de quantité, de besoins de communication, de diversité de composants) est devenu très fort et dont la modélisation ne se fait pas par des équations différentielles (ou aux différences). Au début des années 80, l'automatique à pris en charge ces systèmes, appelés par la suite « systèmes à événements discret » (SED), le mot discret ne signifie ni « temps discret » ni « état discret », mais il se réfère au fait que la dynamique est composée d'évènement qui peuvent d'ailleurs avoir une évolution continue. Ce qui nous intéresse c'est le début et la fin des événements, dans la mesure où, les fins conditionnent de nouveaux débuts.

Les modèles SED sont utilisés dans le domaine de la production manufacturière, la robotique, le trafic de véhicules, la logistique, les réseaux de communications etc. L'étude des systèmes à événements discrets peut être menée avec différents outils tel que : la simulation sur ordinateur, les réseaux de filles d'attente, les modèles dynamiques algébriques, comme l'algèbre « max plus », les réseaux de Petri, les automates et les langages qui reposent sur l'ordre de l'occurrence des événements.

Ce mémoire présente les principales approches qui permettent la synthèse d'un superviseur/contrôleur qui impose au système le respect de spécifications logiques de fonctionnement dans le but de satisfaire un cahier de charges.

Les premiers pas menés dans le domaine de la supervision et de la commande ont été fait par Ramadge et Wonham (R&W). La théorie de la supervision tel qu'introduite par R&W est basée sur l'utilisation des automates à états finis et des langages formels. Un des freins à l'utilisation de cette théorie est le problème de taille des modèles obtenus. Deux voies ont été explorées dans la littérature : l'utilisation de

structures de commande décentralisées, modulaires ou hiérarchiques en gardant les automates comme modèle de base ; ou le changement de l'outil de modélisation lui-même. C'est ainsi que sont apparu par la suite des méthodes de synthèse de superviseur basées sur l'utilisation des réseaux de Petri (travaux de Krogh, Yamalidou, Giua, …).

L'objectif de notre travail est l'étude de la méthode de synthèse de superviseur basée sur l'utilisation des invariants de marquage tel que introduite par Yamalidou. Par la suite, le modèle réseau de Petri sera transformé en un grafcet implantable sur automates programmables industriels (API).

Le travail que nous présentons dans ce mémoire est organisé en trois chapitres. Nous présentons dans le premier chapitre un petit aperçu sur les SED, des notions de base sur les langages formels et les automates à états finis. Les automates constituent l'outil de base de la modélisation dans la théorie de supervision que nous présentons en détail. Cette théorie considère le procédé comme étant un SED qui génère des événements spontanément, et considère aussi le superviseur défini par l'ensemble des spécifications imposées au système comme un SED qui évolue conformément aux évolutions du procédé.

Nous proposons dans le deuxième chapitre, une approche basée sur les invariants de marquages proposée par Yamalidou et Moody qui apporte une solution au problème de taille rencontré dans la théorie de la supervision. Cela est dû à l'utilisation des Rdp comme outil de modélisation, du fait de sa puissance en représentation graphique. Cette théorie consiste à synthétiser un contrôleur en calculant des places de contrôle à partir des contraintes imposées par les spécifications.

Au troisième chapitre, nous traitons, dans un premiers temps le problème lié à l'utilisation de la théorie de Ramadge et Wonham pour la conception de la commande des SED. A cet effet, nous présentons l'approche de Charbonnier qui définit un procédé étendu constitué d'un système et son système de commande modélisé par un grafcet. Pour traiter les spécifications non contrôlables, Kattan a complété cette approche en passant d'un Grafcet à un automate pour pouvoir appliquer le concept de

contrôlabilité introduit dans la théorie de la supervision de R&W, pour reconstruire finalement le grafcet à partir du modèle automate superviseur. Dans un deuxième temps, nous appliquerons l'approche des invariants de marquages qui consiste à calculer les places de contrôles pour synthétiser un contrôleur à base de réseau de Petri. Le modèle réseau de Petri augmenté de son superviseur est transformé en un grafcet implantable sur API.

Nous terminons par une conclusion générale.

I.1. Introduction

Dés ses origines, la théorie des systèmes et leur commande s'est intéressé à des systèmes physiques, décrits généralement par des équations différentielles ou aux dérivées partielles, auxquelles obéissent les phénomènes physiques correspondants.

En 1980 le monde de l'automatique a pris en compte la gestion et la commande des « systèmes à événements discrets »SED, où l'espace d'état est un ensemble discret et le changement d'état s'effectue à des instants précis. A chacune de ces transitions on peut associer un événement. Les variables peuvent prendre des valeurs continues, par contre leur début et leur fin sont conditionnés par des événements.

Le vocable 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 soucis que des « débuts » et des « fins » de ces phénomènes (les événements discrets) et leur enchainement dynamique, logique ou temporel. Les modèles SED sont utilisés dans le domaine de la production manufacturière, la robotique, le trafic de véhicules, la logistique, les réseaux de communications…

La théorie des SED peut être divisée actuellement en deux grandes approches :

-l'approche quantitative qui s'adresse à l'aspect évaluation de performance voire à l'optimisation de performances d'où la distinction de deux autres approches : la première est l'analyse de perturbation initiée, la deuxième est l'approche « max plus » qui se caractérise par l'utilisation d'une algèbre adaptée.

-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 ont utilisé cette approche pour aborder la problématique de la commande qui fait l'objet de notre étude, et dont il est essentiellement question dans la suite de ce mémoire.

Dans ce chapitre nous avons présenté l'approche proposée par Ramadge et Wonham qui s'intéresse à l'existence et à la synthèse d'une supervision pour des systèmes à événements discrets modélisés par les automates à états finis.

I.2. Définition d'un SED

I.2.1. Concept de système

[15]Un système est l'un de ces concepts primitif dont la compréhension peut être mieux abordée par l'intuition qu'à une définition exacte. Nous pouvons fournir trois définitions représentatives existantes en littérature :

- une agrégation ou assemblage de choses combinées par nature ou par l'homme pour former une totalité intégrante ou complexe.
- une interaction régulière ou groupe interdépendant d'articles qui forment une totalité unifiée.
- une combinaison de composants qui agissent ensemble pour exécuter une fonction impossible en partie individuelle.

I.2.2. Concept d'événement

[15]C'est un concept primitif avec une bonne base intuitive qui doit être considéré comme se produisant instantanément et provoquant des transitions d'un état à un autre état. Un événement peut être identifié par une action spécifique prise (par exemple, quelqu'un qui appuie sur un bouton), un événement peut être considéré comme un phénomène spontané dicté par la nature (par exemple, un ordinateur tombe en panne pour une raison quelconque trop compliquée à comprendre), ou il peut être le résultat de plusieurs conditions qui sont soudainement toutes atteintes (par exemple, le niveau du liquide dans le réservoir dépasse une valeur).

I.2.3. Définition d'un SED

[15]Un système à événements discrets est un système à espace d'états discrets dont l'état change seulement à certains instants, de façon instantanée et dont les transitions entre états sont associées à l'occurrence des événements discrets asynchrones. En l'absence d'événements, 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 décrit 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 ou 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.

Exemple

Considérons l'exemple de la souris qui se déplace d'une manière spontanée à l'intérieur d'un labyrinthe comme montré en figure I.1.

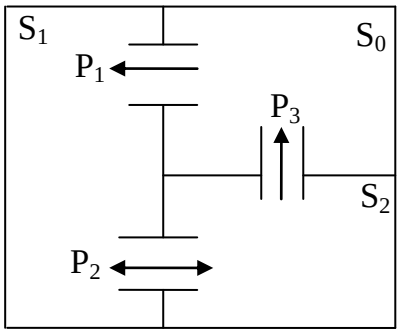


Figure I.1. Exemple de SED

Les salles communiquent par des portes unidirectionnelles P_1 et P_3 et bidirectionnelle P_2. On note « P_i » l'événement : « la souris passe par la porte P_i ». Soit $\sum$ l'ensemble des événements : $\sum = \{P_1, P_2, P_3\}$.

Les situations possibles sont :

-la souris est dans la salle S_0.

-la souris est dans la salle S_1.

-la souris est dans la salle S_2.

I.3. Langage et automate

La synthèse de la commande ou encore la modélisation des systèmes à événements discrets nécessitent des outils qui permettent de traiter mathématiquement les problèmes relatifs aux SED. La théorie des automates développée avec la théorie des langages formels permet de décrire essentiellement d'un point de vue logique (analyse qualitative) un SED comme étant un alphabet de langage et les séquences d'événement sont des mots de ce langage.

Un automate est alors un dispositif qui engendre un langage en manipulant l'alphabet.

I.3.1. Langage [8]

Pour formaliser des systèmes à événements discrets sous forme de langage, on associe à chaque SED un ensemble d'événements représentés par des symboles (σ_1…… σ_n).

Cet ensemble fini de symboles peut être vu comme un alphabet d'un langage et les séquences d'événements sont des mots (aussi appelés chaînes) de ce langage. Dans la suite nous utiliserons le mot « séquence ».

Définition 1

Un alphabet est un ensemble fini de lettres ou de symboles noté $\sum$

Exemple

Le biologiste intéressé par l'étude de l'ADN utilisera un alphabet à quatre lettres (symboles) {a, c, g, t} pour les quartes constituants des gènes : adénines, cytosine, gamine et thymine.

Dans le cas d'un SED, l'alphabet pourra représenter l'ensemble des événements possibles dans le système. Cet ensemble sera composé de tous les événements qui font évoluer le SED.

Définition 2

Une séquence sur $\sum$ est une suite finie de symboles. La langueur d'une séquence w notée $|w|$ est le nombre de symboles constituant cette séquence.

Exemple : abbac et ba sont deux séquences sur l'alphabet {a, b, c} et leurs longueurs sont $|abbac|=5$ et $|ba|=2$.

Définition 3

Soit $\sum^*$ l'ensemble de toutes les séquences qui peuvent être formées sur les éléments de$\sum$, y compris la séquence vide1.

Soit $w = w_1 \ldots\ldots\ldots\ldots\ldots\ldots w_\ell$, une séquence sur $\sum$.

Les séquences 1, w_1, w_1w_2,……, $w_1 \ldots\ldots w_{\ell-1}$, $w_1 \ldots\ldots\ldots\ldots w_\ell = w$ sont les préfixes de w.Un préfixe de w différent de ε et de w est dit propre.

De façon semblable,

$$1, w_1, w_1w_2, \ldots\ldots\ldots\ldots\ldots, w_1 \ldots\ldots w_{\ell-1}, w_1 \ldots\ldots\ldots\ldots w_\ell = w$$

Sont les suffixes de w. Un suffixe de w est qualifié de propre s'il diffère de 1 et de$w$.

Définition 4

Un langage sur $\sum$ est simplement un ensemble (fini ou infini) de séquences sur $\sum$. En d'autres termes, un langage est une partie de $\sum^*$. On distingue en particulier le langage vide.

Exemple

Considérons l'alphabet de $\sum$= {a, b, c}. L'ensemble {a, aa, bbc, ccca, ababab} est un langage fini. L'ensemble L_a des séquences sur $\sum$ comprenant un nombre pair de a est aussi un langage (infini)

L_a= { 1, b, c, aa,……, aba, aca,…….., abaacaa,.....}.

I.3.2. Quelques opérations sur les langages[5]

Soient l'alphabet $\sum$ et deux langages A et B construits sur ce même alphabet.

- *L'union*

Noté $A \cup B = A+B$ tel que : $A \cup B = \{w \backslash w \in A \text{ ou } \in B\}$

L'union possède un élément neutre qui est l'ensemble vide noté $\emptyset$.

$A+\emptyset = \emptyset+A = \emptyset$

L'union est idempotente : $A \subset \sum^*$ $A + A = A$

- *La concaténation*

$A.B = \{w \in \sum^* / \exists w_1 \in A, \exists w_2 \in B, w=w_1.w_2\}$;

{1}=1, c'est l'élément neutre de la concaténation.

Définition 1

Nous pouvons à présent définir la préfixe-clôture (la fermeture préfixielle) d'un langage L, comme le langage contenant tous les préfixes des séquences de L. Nous noterons par pref (L) la préfixe-clôture du langage L :

$$\text{Pref}(L) = \{w_1 \in \sum^* / \exists\, w_2 \in \sum^*, w_1.w_2 \in L\}$$

- **Langage préfixe-clos :**

Un langage L est fermé par ses préfixes (ou préfixe-clos) s'il est égal à sa préfixe-clôture. C'est-à-dire, L=pref (L).

Définition 2

Soit A un langage dans $\sum^*$ tel que : $A.A=A^2$, $A^0=1$

L'étoile de Kleene de A est définie par :

$A^*= A^0+A+A^2+A^3+\ldots+A^n+\ldots$

$=\oplus A^i$

i ϵN

I.4. Les automates à états finis[5][8]

L'automate à états finis représente l'outil le plus ancien et le plus classique pour décrire le comportement d'un SED.

Un automate est une machine à états constituée d'états et de transitions associés à des événements. Son comportement dépend du mot fourni en entrée : l'automate passe d'état en état, suivant les transitions à la lecture de chaque symbole de l'entrée. L'automate est « fini » s'il possède un nombre fini d'états distincts.

Définition1

Un automate fini $\mathcal{A}$ peut être défini par un quintuplet $\mathcal{A}= (X, \sum, \delta, x_0, X_m)$ où :

X : est l'ensemble fini des états ;

$\sum$: est un alphabet fini ;

δ: est la fonction de transition d'états $\delta : X \times \sum \rightarrow X$;

x_0: est l'état initial ;

X_m est l'ensemble des états finaux (marqués).

Un automate peut être représenté par son graphe de transition d'état. Le graphe de transition d'état d'un automate fini $\mathcal{A}$ est donné à la figure I.2.

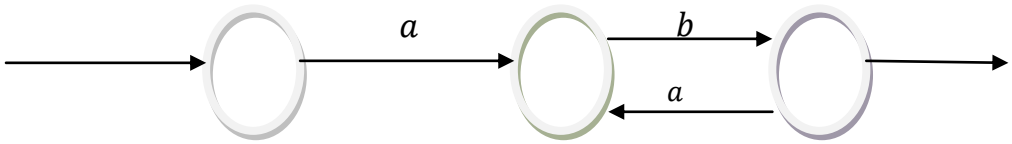


Figure I.2. Graphe de transition d'états d'un automate à états finis.

Dans la figure I.2, les états de $\mathcal{A}$ sont représentés par des cercles. Nous avons : $X= \{x_0, x_1, x_2\}$ l'ensemble des états. L'état initial x_0 est repéré par une flèche entrante. L'état final est représenté par une flèche sortante, ainsi : $X_m= \{x_2\}$.

La fonction δ de transition d'état est représentée par des arcs associés à des symboles de l'alphabet ∑. Dans notre exemple, l'alphabet ∑ correspond à l'ensemble $\{a,b\}$.

Il existe une transition d'états associée au symbole « a » entre x_0 et x_1. Cela signifie : δ $(x_0,a)=x_1$.

Définition 2

Un état $x \in X$ est dit accessible (atteignable) s'il existe une séquence $w \in \sum^*$ telle que :

x=δ (x_0,w), c'est-à-dire que l'automate peut y accéder depuis l'état initial.

Par extension, l'automate $\mathcal{A}$ est accessible si tout état $x \in X$ est accessible.

Définition 3

Un état $x \in X$ est dit Co-accessible (Co-atteignable) s'il existe une séquence $w \in \sum^*$ telle que :

$\delta\ (x,w) \in X_m$ c'est-à-dire qu'à partir de cet état l'automate peut atteindre un état marqué.

Par extension, l'automate $\mathcal{A}$ est Co-accessible si tout état $x \in X$ est Co- accessible.

Définition 4

On dit que l'automate $\mathcal{A}$ est émondé (vivant) lorsqu'il est à la fois accessible et Co-accessible.

Définition5

La trace d'un chemin est la suite des étiquettes qui le compose.

Exemple

$w = \sigma_1\ \sigma_2\ \sigma_3 \ldots..\ \sigma_n$

La trace du chemin précédent constitue la séquence w.

Définition 6

Un langage $L \subseteq \sum^*$ est régulier s'il est accepté par un automate fini.

Tous les langages finis sont réguliers spécifiés a priori par des expressions régulières.

Exemple

Si a est une lettre, alors c'est une expression régulière qui représente $\{a\}$

I.4.1. Langage généré par un automate fini :

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

Définition

Le langage généré par un automate fini $\mathcal{A}$ est donné par :

$$L(\mathcal{A})= \{w \in \sum^{*} / \delta (x_0,w) \in X\}$$

Ce langage représente l'ensemble de toutes les séquences 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 pref $(L(\mathcal{A}))=L(\mathcal{A})$.

I.4.2. Langage marqué d'un automate fini

Définition 1

Le langage marqué d'un automate fini $\mathcal{A}$ est donné par :

Ce langage représente l'ensemble de toutes les séquences qui permettent de rejoindre un état marqué à partir de l'état initial, de telles séquences sont appelées taches.

Cette définition implique que $L_m(\mathcal{A})$ est inclus $L(\mathcal{A})$, puisque $X_m \in X$.

Définition 2

On dit qu'un automate est déterministe s'il ya bijection entre l'ensemble des chemins dans l'automate et l'ensemble des traces de chemins correspondants.

Soit l'automate $\mathcal{A}= (X, \sum, \delta, x_0, X_m)$, $\mathcal{A}$ est déterministe si et seulement si :

1) il contient un seul élément qui représente l'état initial.
2) si $x, x', x'' \in X, w \in \sum$ alors :

$(x,w, x'), (x, w, x'') \in \delta \Longrightarrow x'=x''$

- $\delta(x,w)=x'$, δ est une fonction partielle on écrit :

$\delta(x,w)=x'$ si $(x,w,x') \in \delta$

$\delta(x,w)$ peut ne pas être défini, on écrit :

$\delta(x,w)!$ lorsque $\exists\, x'/\ \delta(x,w)=x'$

On peut alors étendre δ à $\sum^*$ en écrivant :

$\delta(x,\varepsilon)=x, \forall\, x \in X$ et $\delta(x,\sigma w)=\delta(\delta(x,w),\sigma)$

Si $\delta(x,w)$! et $\delta(x,\sigma)$! avec $\delta(x,w)=x'$.

I.4.3. Langage reconnaissable :

L'ensemble des mots acceptés par un automate est par définition le langage reconnu par l'automate. On dit qu'un langage est reconnaissable s'il existe un automate déterministe qui le reconnait.

- **Théorème de Kleene [13]**

Le théorème de Kleene constitue un pont entre les langages réguliers et les langages reconnaissables par l'automate. Il nous permet de décrire tout langage reconnaissable par une expression régulière.

Théorème

Si $L \subseteq \sum^*$, alors les affirmations suivantes sont équivalentes.

1) L est rationnel. Autrement dit, L est obtenu par un nombre fini d'addition, de produit, d'étoiles, de langages finis.

Exemple

$L=\alpha(\beta+(\alpha\beta)^*)^*\beta\,\alpha^*$

2) L est reconnaissable. Autrement dit, $L=L_m(\mathcal{A})$, $\mathcal{A}$ étant un automate fini.
3) L est solution d'un système d'équation unilatérale linéaire, de type :

$L=CX$ solution de $x=Ax+B$ ou $L=yB$, $y=yA+C$ y où :

C_j, B_i, A_{ij} sont des langages finis sur $\sum$

I.4.4. compositions des automates

En réalité le procédé est composé de plusieurs procédés élémentaires en interaction plus ou moins forte. Le comportement du procédé global peut être décrit par l'automate résultant de la composition des automates décrivant les procédés élémentaires. On parle alors, de la composition asynchrone (absence d'événements communs), dans le cas d'existence de liaisons, qui amène les automates à partager au moins un événement, on parlera de composition synchrone.

- **La composition synchrone**

Est effectuée quand les alphabets des langages associés aux automates considérés ont au moins un événement en commun.

Soient $L1 \subseteq \sum_1^*$, $L_2 \subseteq \sum_2^*$, avec $\sum_1 \cap \sum_2 \neq \emptyset$ et $\sum=\sum_1 \cup \sum_2$

Soient deux automates :

$\mathcal{A}_1= (Q, \sum_1, \xi, q_0, Q_m)$ et $\mathcal{A}_2= (X, \sum_2, \delta, x_0, X_m)$

Le composé synchrone $\mathcal{A}_s$ de $\mathcal{A}_1$ et $\mathcal{A}_2$, noté $\mathcal{A}_s=\mathcal{A}_1||\mathcal{A}_2$ est défini par 5-uplet :

$(Q \times X, \sum_1 \cup \sum_2, \xi \times \delta, q_0 \times x_0, Q_m \times X_m)$

Où : $Q \times X$: est l'ensemble des états ;

$\sum_1 \cup \sum_2 =\sum$: représente l'alphabet de $\mathcal{A}_s$;

$q_0 \times x_0$: est l'état initial ;

$Q_m \times X_m$: est l'ensemble des états finaux ;

Et où la fonction de transition d'états $\xi \times \delta$ est défini par :

$(\xi \times \delta)((q, x), \sigma)= (q', x')$ si $\delta(q, \sigma)=q'$! et $\xi(x, \sigma)=x'$!

$(\xi \times \delta)((q, x), \sigma) = (q', x)$ si $\xi(q, \sigma)$! avec $\sigma \in \sum_1 \backslash \sum_2$

$(\xi \times \delta)((q, x), \sigma) = (q, x)$ si $\delta(x, \sigma)$! avec $\sigma \in \sum_2 \backslash \sum_1$

- **La composition asynchrone**

Elle est réalisée quand les alphabets des langages associés aux automates considérés n'ont aucun événement en commun.

Soient $L_1 \subseteq \sum_1^*$, $L_2 \subseteq \sum_2^*$, avec $\sum_1 \cap \sum_2 = \emptyset$ et $\sum = \sum_1 \cup \sum_2$ (Pas d'événement commun)

Soient deux automates :

$A_1 = (Q, \sum_1, \xi, q_0, Q_m)$ et $A_2 = (X, \sum_2, \delta, x_0, X_m)$

Le composé est $\mathcal{A}_a$ de $\mathcal{A}_1$ et $\mathcal{A}_2$ noté $\mathcal{A}_a = \mathcal{A}_1 || \mathcal{A}_2$ est défini par le 5-uplet :

$(Q \times X, \sum_1 \cup \sum_2, \xi \times \delta, q_0 \times x_0, Q_m \times X_m)$, avec :

$(\xi \times \delta)((q, x), a) = (q, x)$ si $\xi(q, a) = q'$ pour $a \in \sum_1$

$(\xi \times \delta)((q, x), a) = (q, x')$ si $\delta(x, a) = x'$ pour $a \in \sum_2$

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

Evénement : symbole $\sigma \in \sum$

Trace : séquence w

Comportement du système : langage L. Un langage préfixe- clos 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 séquence w : $q' = \delta(q, w)$ ou q' est l'état atteint quand la séquence w est exécutée depuis l'état q. Cette dernière équation peut être explicitée : d'une manière générale, si w est une chaine et σ un événement :

$$\delta(q, w\sigma) = \delta(\delta(q, w), \sigma).$$

I.5. Synthèse de la commande[1][2]

La théorie de supervision de Ramadge et Wonham a pour objet la synthèse de commande pour les systèmes à événements discrets.

Dans la théorie du contrôle (Ramadge et Wonham 1987-1988) un procédé est supposé évoluer de façon spontanée générant des événements, le principe de supervision est présenté dans la figure I.3.

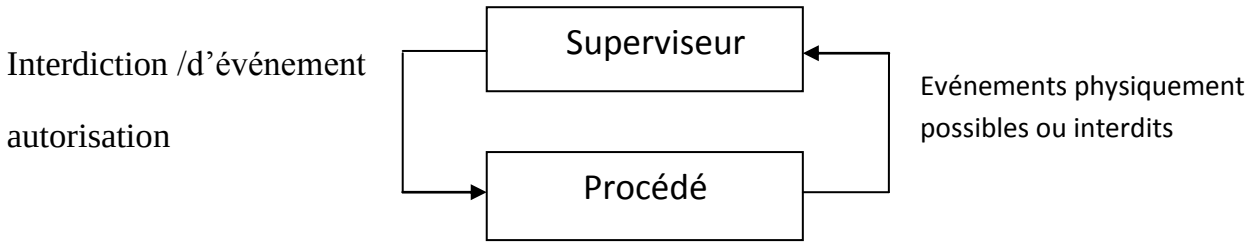


Figure I.3. Procède couplé à un superviseur.

Dans ce schéma le procédé est considéré et perçut comme un SED par le superviseur qui est lui-même un SED. En sortie, le procédé génère des événements qui seront observés comme entrée par le superviseur et vice versa. Etant donné, un ensemble de spécifications, le superviseur couplé au procédé (interdisant ou autorisant l'occurrence de certains événements) modifie le comportement du système tout en respectant les spécifications.

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.

I.5.1. Fonctionnement en boucle fermée

On désigne par le fonctionnement en boucle fermée, le procédé couplé à son superviseur. Dans ce fonctionnement, un événement ne peut-être produit par le procédé que s'il est physiquement possible et non interdit par le superviseur.

La supervision modifie ainsi le fonctionnement du procédé afin de respecter les spécifications. Le langage de fonctionnement en boucle fermée sera inclus dans le langage du procédé par conséquent le superviseur ne peut que restreindre le comportement d'un procédé.

I.5.2. Concept de supervision

Le procédé global (procédé couplé au superviseur) est modélisé par un automate résultant d'un produit asynchrone des modèles automate décrivant les comportements du procédé et du superviseur.

Etant donné des spécifications, le superviseur joue le rôle d'inhibiteur d'événement intervenant sur l'évolution du procédé par l'intermédiaire des lois de contrôle qui définissent l'ensemble des événements autorisés ou interdits.

Lorsque le procédé se trouve dans un état q_i il reçoit de la part du superviseur une liste γ_i d'événements autorisés. La prochaine évolution du procédé se fera sur l'occurrence d'un événement $\sigma_i \in \gamma_i$. A partir du nouvel état q_j atteint par le procédé, une nouvelle liste d'événements autorisés est fournie. Ainsi on constitue ce superviseur d'une manière telle que l'ensemble des événements autorisés dépendent de l'évolution passée du procédé.

I.5.3. Modélisation du SED

Ramadge et Wonham ont opté pour une description qualitative dont le SED considéré est modélisé par un automate, tel que le comportement du procédé est décrit par la structure $G = (Q, \sum, \delta, q_0)$, générant des événements sur l'alphabet $\sum$.

Le procédé génère des événements qui peuvent être interdits par le superviseur que nous appellerons « événement contrôlables » notés $\sum_c$, comme ils peuvent ne pas être interdits par le superviseur dans ce cas nous les appellerons « événements incontrôlables » notés $\sum_{uc}$. Ainsi on partitionne l'alphabet $\sum$ de la manière suivante : $\sum = \sum_c \cup \sum_{uc}$

Définissons la fonction des événements admissibles (possibles) par

$\sum_{\mathbf{a}} : Q \longrightarrow 2^{\Sigma}$, ou $\sum_{\mathbf{a}} (q)$: l'ensemble des événements σ possibles de G à l'état q du procédé c'est-à-dire pour lesquels δ (σ,q) est définie.

Nous associons à l'automate G un langage $L(G)$ sous- ensemble de $\sum^{*}$, $L(G)$ est défini par l'ensemble des séquences s telles que :

$L(G)$= **{**$s \in \sum^{*}$ et δ (σ,q) ∈ G**},** ou : δ : fonction de transition d'état.

$L(G)$: représente l'ensemble de toutes les séquences d'événements pouvant être générés par le procédé. Dans la suite nous considérons tous les langages préfixes-clos.

I.5.4. Définition formelle d'un superviseur

La figure I.3 représente le procédé G couplé à son superviseur S, Formellement, on peut associer a ce superviseur une fonction **:**

$f : L(G) \longrightarrow \Gamma$ tel que $\Gamma = \{\gamma \in 2^{\Sigma} : \sum_{\mathbf{uc}} \subseteq \gamma\}$

-Un élément γ de Γ représente l'ensemble des événements autorisés (non interdits) par le superviseur ce qui implique l'inclusion des événements incontrôlables dans les éléments Γ puisque ces derniers ne peuvent pas être interdits.

Si l'on désigne par **s** une chaine générée par G / ($s \in L(G)$) une liste des événements autorisés $\gamma \in \Gamma$ sera fournie par le superviseur en sortie notés $\gamma = f(s)$.

Le superviseur remet à jour ses sorties à chaque occurrence d'un événement dans le procédé pour chaque séquence $s \in L(G)$:

- δ (s, q_0) =q désigne l'état atteint par s depuis l'état initial (c'est-à-dire l'état courant).

-$\sum_a$ (δ (s, q_0)) ∩ $f(s)$ est l'ensemble des événements σ qui sont physiquement possibles depuis q ($\sigma \in \sum_a$ (δ(s, q_0))) et autorisés par S ($\sigma \in f(s)$). Le procédé ne peut générer d'événements si ces derniers ne sont pas autorisés par le superviseur, ainsi, après une séquence s, seuls les événements de $\sum_a$ (δ(s, q_0)) ∩f(s) (physiquement possible) peuvent se produire dans le fonctionnement en boucle fermée.

Définition

Si on note S/G le système résultant en boucle fermée**,** S/G désigne l'automate accepteur du langage $L(S/G)$ définie de la manière récursive suivante :

- $\varepsilon \in L\,(S/G)$
- $[s\,\sigma \in L(S/G)] \Leftrightarrow [(s \in L(S/G)] \wedge (s\sigma \in L(G) \wedge (\sigma \in f(s))$ ou $\wedge$ désigne l'opérateur de conjonction .
- Le langage préfixe-clos $L(\mathrm{S}/G) \subseteq L(G)$ est ainsi obtenu par l'ensemble des chaines qui sont physiquement possibles et autorisées par le superviseur.

I.5.5. Contrôlabilité et synthèse de la supervision

I.5.5.1. Concept de contrôlabilité et existence d'un superviseur

a) *Contrôlabilité*

Considérons un procédé G et un ensemble de contraintes traduites par une spécification de fonctionnement E. Synthétiser un superviseur S revient à définir ce dernier de façon à ce que le système en boucle fermée respecte les spécifications.

On note par $L(E)$ le langage de spécification et par l'intersection

$K = L(G) \cap L(E)$ le fonctionnement que l'on désir obtenir en boucle fermée .

En général, l'existence d'événements incontrôlables implique que l'on ne peut restreindre par le superviseur le fonctionnement d'un procédé à n'importe quel sous-ensemble de fonctionnement ce qui a conduit Ramadge et Wonham à introduire la notion de langage contrôlable dans le but de synthétiser un superviseur S tel que : $L(S/G) = L(G) \cap L(E)$.

Définition

Un langage $K \subseteq \sum^{*}$ est dit contrôlable par rapport à un langage $L(G)$ si :

$\overline{K}\sum_{\mathrm{uc}} \cap L(G) \subseteq \overline{\mathrm{K}}$

En interprétant K comme le langage de spécification et $L(G)$ comme le langage du procédé, K est contrôlable par rapport à $L(G)$ si pour toute séquence d'événements

($s \in K$), et pour toute occurrence d'événements incontrôlables ($G \in \sum_{uc}$), la séquence $s\,\sigma$ tel que ($s\,\sigma \in L(G)$ implique ($s\,\sigma \in K$).

C'est-à-dire que cette séquence ne mène pas hors du fonctionnement légal spécifié.

L'existence d'un superviseur S tel que $L(S/G)=L(G)\cap L(E)$ réside dans le concept de contrôlabilité.

b) *Existence d'un superviseur et langage suprême contrôlable*

Soient $L(G)$ le langage d'un procédé G, $L(E)$ un langage de spécification préfixe-clos et non vide tel que : $K= L(E) \cap L(G)$. Il existe un superviseur S tel que $L(S/G) = K$ si et seulement si, K est contrôlable par rapport à $L(G)$.

Dans le cas contraire le superviseur n'existera pas ce qui a ramené Ramadge et Wonham à la détermination d'un langage suprême contrôlable noté Supc qui constituera le plus grand sous ensemble de K qui soit contrôlable. Or, Supc (K) est le plus grand élément contrôlable, le fonctionnement en boucle fermée obtenu est le plus large possible : il est dit permissif maximal, ainsi, le fonctionnement est qualifié d'optimal.

I.5.5.2. Synthèse du superviseur

A partir des modèles automates d'un procédé G et d'une spécification de fonctionnement E.des algorithmes permettent de vérifier la contrôlabilité du langage proposés par Ramadge et Wonham, de plus dans le cas ou le langage $L(E)$ n'est pas contrôlables, nous pouvons synthétiser un modèle automate du langage suprême contrôlable supc (K) en utilisant l'algorithme de Kumar.

Intuitivement, la détermination de Supc (K) revient à inhiber par la supervision les événements contrôlables qui précédent, 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.

a) ***Algorithme de calcul du langage suprême contrôlable (Algorithme de Kumar) :***

G= (Q, $\sum$, δ, q_0) et E= (X, $\sum$, ξ, x_0) les modèles automates d'un procédé et d'une spécification, K désigne le langage $L(E)$ .Rappelons que la fonction $f : x \rightarrow$ Q, définit la correspondance entre les états de E et les états de G. L'algorithme proposé est basé sur les pas suivants :

Pas 1 : Construit le composé synchrone D de G et de E associé à un langage $L(D)$ noté L_{D}.

Pas 2 : Détermination de l'ensemble des états défendus pour lesquels : Il existe un événement σ de $\sum_{\mathrm{uc}}$ (non contrôlable) tel que δ (q,σ) est définit dans G et ξ (x, σ) n'est pas définie dans E.

Pas 3 : Détermination de l'ensemble des états faiblement défendus pour lesquels : Il existe un chemin non contrôlable qui conduit à un état de D. Un chemin non contrôlable étant un chemin pour lequel aucun des états visités ne possède d'événements contrôlable admissibles.

Pas 4 : Suppression des états défendus et des états faiblement défendus ainsi que les transitions associés à ces états.

Suppression des états pour lesquels il n'existe pas de chemin accédant a cet état depuis l'état initial. Ainsi obtenu le langage suprême de l'automate (Sup**c**).

- A partir du (pas 2) nous obtenons le modèle automate accepteur pour :

  - il n'y a pas d'événement contrôlable non accessible (pas 1).

  - Les événements incontrôlables qui sont physiquement possibles (c'est-à-dire qui peuvent être générés par le procédé) et qui ne sont pas tolérés par la spécification ne répondent pas au concept de contrôlabilité doivent être supprimés du fonctionnement en boucle fermée.

- On supprime tous les états pouvant joindre les états défendus par des chemins non contrôlables (c'est-à-dire l'ensemble obtenu en (pas 3)).

La suppression des états défendus et faiblement défendus rend certains états non atteignables, ces états aussi seront retirés pour obtenir finalement le modèle automate accepteur du langage suprême contrôlable noté D′.

I.6. Exemple d'application

Considérons un système de communication composé de deux serveurs identiques : S_1 et S_2 et un canal liant ces deux serveurs. Conformément à la figure I.4, les serveurs envoient et reçoivent des messages indépendamment l'un de l'autre.

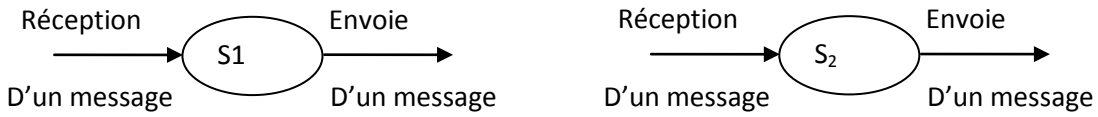
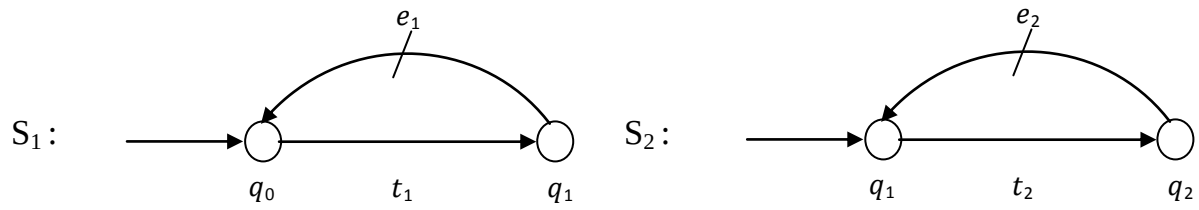


Figure I.4. Exemple de système de communication

I.6.1. Modélisation du procédé

L'évolution du procédé se fait d'une façon spontanée générant des événements, son fonctionnement peut être décrit par un ensemble de séquences d'événements qui constitue un langage formel sur l'alphabet des évènements. On peut modéliser le fonctionnement de chaque serveur par un accepteur (automate sans sortie). La figure I.5 représente le graphe de transitions d'états des accepteurs S_1 et S_2.



(a) (b)

Figure I.5. Modèle accepteur des serveurs.

Prenons la présence d'un message au niveau du serveur S_1 comme étant son état initial q_0, l'occurrence de l'événement e_1 ramène le serveur dans l'état d'envoi q_1 la fin de la transmission (le message est dans le canal) est modélisé par l'occurrence de l'événement t_1. Désignons respectivement par $\sum_1$ et $\sum_2$ les alphabets des serveurs S_1 et S_2, tel que $\sum_1 = \{e_1, t_1\}$ et $\sum_2 = \{e_2, t_2\}$.

On définit le fonctionnement du système de communication sur un alphabet

$\sum = \sum_1 \cup \sum_2$, par convention nous représentons toutes transitions associées à un événement contrôlable par un arc barré. Ainsi, nous supposons que

$\sum_c = \{e_1, e_2\}$, $\sum_u = \{t_1, t_2\}$.

$\sum_c$ (événements contrôlables), $\sum_u$ (événements incontrôlables).

Nous effectuerons un composé asynchrone des modèles S_1 et S_2 pour obtenir un modèle automate de notre système de communication. Le modèle G est représenté par son graphe de transition d'état dans la figure I.6.

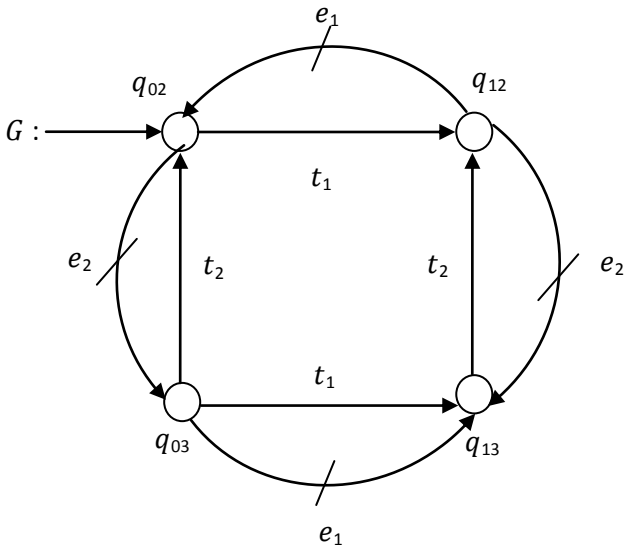


Figure I.6. Modèle accepteur du système de communication

I.6.2. Définition de la spécification

Notre système de communication doit respecter la présence d'un canal entre les deux serveurs de capacité limitée à un seul message à la fois.

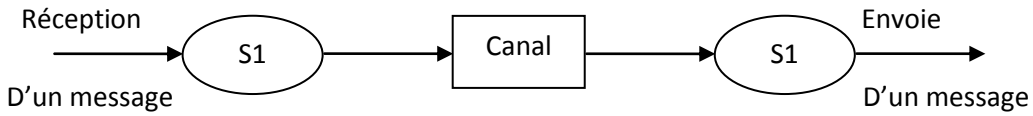


Figure I.7. Système de communication sous la contrainte du canal.

Conformément à la figure I.7 un message doit passer de S_1 vers S_2. La présence du canal entre S_1 et S_2 impose que :- le serveur S_2 ne peut recevoir de message que si le message est traité par le canal, c'est-à-dire, à la présence d'un message dans le canal,- le serveur S_2 ne peut envoyer de message au canal que si ce dernier ne contient pas de message. Le canal est supposé vide dans son état initial.

Le modèle accepteur de la spécification E de la figure I.8 permet de garantir les conditions du fonctionnement. Les états C_0 et C_1 représentent respectivement : « présence d'un message dans le canal », et « absence de message dans le canal ».

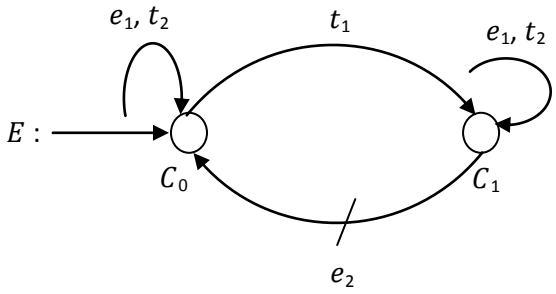


Figure I.8. Accepteur de la spécification.

Lorsque le canal est vide l'occurrence de l'événement e_2 est interdite (début du cycle de S_2). Sur l'occurrence de l'événement t_1 (fin de cycle du serveur S_1 et transmission d'un message au canal), l'accepteur E change d'état et passe dans l'état C_1. Dans cet état, l'occurrence de l'évènement contrôlable e_1 est interdite (début du cycle de S_1).

En supervisant le fonctionnement du procédé nous obtenons le modèle de fonctionnement en boucle fermée noté D, résultant du composé synchrone des modèles G et E : nous avons alors $D=G \parallel E$.

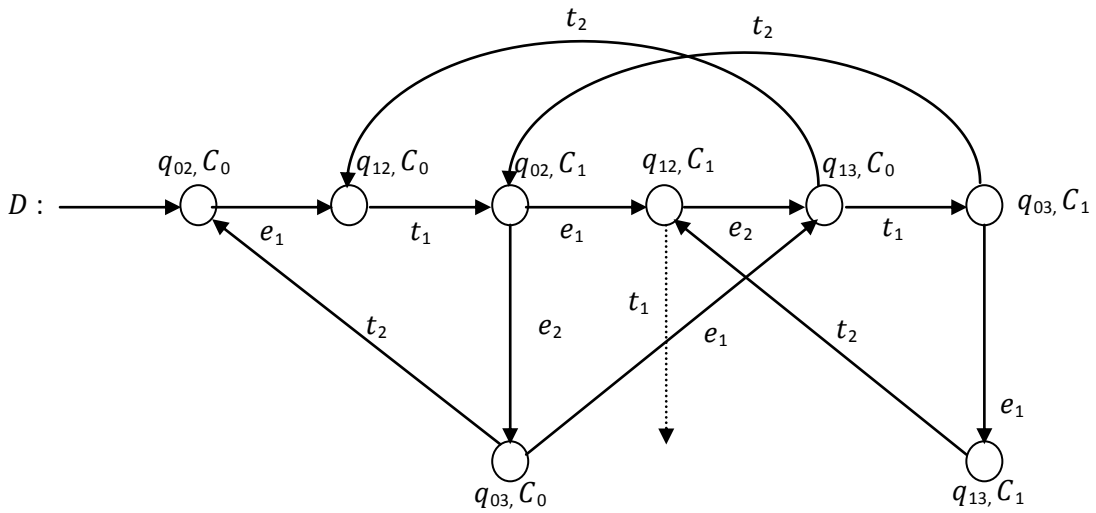


Figure I.9. Modèle du fonctionnement en boucle fermée.

Considérons la séquence d'événements e_1 t_1 e_1 (le serveur S_1 envoie un message, message transmis au canal, puis envoie à nouveau un message). Cette séquence ramène le procédé dans l'état q_{12} qui représente l'action d'envoi du message

par S_1 et le serveur S_2 est en attente de recevoir un message. A partir de cet état t_1 peut être généré spontanément par le procédé. Dans ce cas on peut dire que la séquence d'évènements $e_1\ t_1\ e_1\ t_1$ est possible par le procédé, c'est à dire $e_1\ t_1\ e_1\ t_1 \in L(G)$.

La séquence d'évènements $e_1\ t_1\ e_1$ conduit la spécification dans l'état C_1 (présence d'un message dans le canal). A partir de cet état, il n'existe pas de transitions de sortie associée à t_1. Cela signifie que l'événement t_1 n'est pas toléré par la spécification après la séquence $e_1\ t_1\ e_1$ (présence d'un message dans le canal). Ainsi, la séquence $e_1\ t_1\ e_1\ t_1$ n'est pas un élément du langage de spécification

($e_1\ t_1\ e_1\ t_1 \notin L(E)$).

Cependant l'événement incontrôlable t_1 ne peut être interdit après la séquence

$e_1\ t_1\ e_1$.

Le langage de spécification $L(E)$ sera alors qualifié de non contrôlable par rapport au langage du procédé $L(G)$.

I.6.3. Synthèse du superviseur

Le langage de spécification $L(E)$ de notre exemple de système de communication n'est pas contrôlable par la présence de l'évènement incontrôlable t_1 qui ne peut être interdit après la séquence $e_1\ t_1\ e_1$, mais 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 .

Pour se faire nous allons appliquer les pas de l'algorithme de Kumar qui consiste à supprimer l'ensemble des états défendus, les états faiblement défendus, ainsi que les transitions associées à ces états, et finalement on supprime les états inaccessibles.

Pour obtenir le modèle accepteur D' représenté par figure I.11, on supprime l'état défendu et l'état faiblement défendu.

Les états défendus et faiblement défendus sont représentés dans la figure suivante :

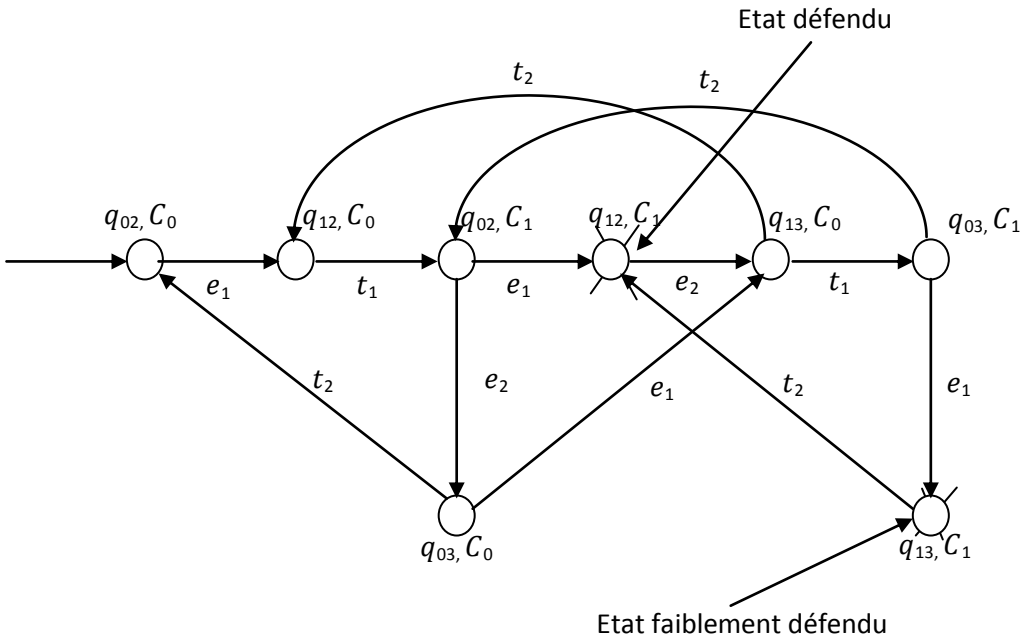


Figure I.10. Etats défendu et faiblement défendu du modèle accepteur de fonctionnement désiré.

Pour obtenir le modèle accepteur D', on supprime les états défendu et faiblement défendu montrés dans la figure I.10. Nous obtenons finalement D′ de la figure1.11.

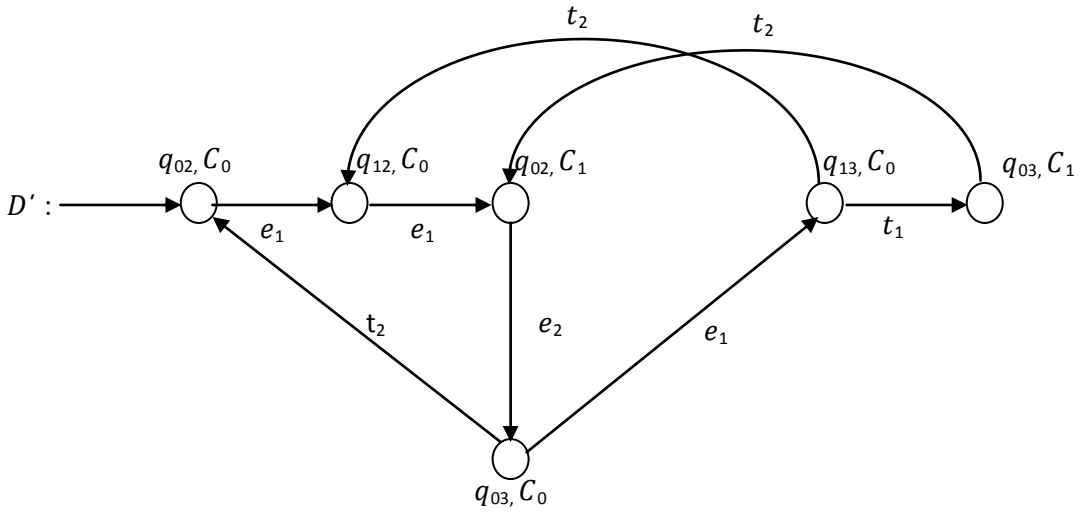


Figure I.11. Superviseur pour la contrainte spécifiée par le canal.

I.7. Conclusion

Nous avons pu présenter dans ce premier chapitre quelques notions sur les SED que nous avons modélisés par un langage régulier. Ce langage, nous a permis de construire un automate qui décrit plus précisément le comportement du SED.

Les automates représentent la méthode classique de modélisation adoptée par Ramadge et Wonham dans leur approche, qui consiste à synthétiser un superviseur par l'utilisation de modèles automates de procédé et de spécification comprenant respectivement n et m états.

La présence des événements incontrôlables ramène le système à généré des événements qui sont toléré par le procédé et qui ne le sont pas par la spécification qualifiant ainsi, le langage de spécification est non contrôlable par rapport au langage du procédé, l'algorithme de Kumar permet d'obtenir un langage contrôlable le plus permissif possible.

Le superviseur obtenu comporte (au plus) n.m états, sa taille est souvent plus grande que celle du procédé, on en résulte que l'explosion combinatoire due à l'utilisation des modèles automates rend impossible la synthèse du superviseur.

La théorie de Ramadge et Wonham par son utilisation des automates à états et un point de vue centralisé à inciter plusieurs travaux à proposer des approches permettant de remédier au problème de taille. L'utilisation d'autres outils par exemple les réseaux de petri et le grafcet sont proposés pour palier à ce problème.

II.1. Introduction

Issue des travaux de Carl Adam Petri, proposés en 1962 dans sa thèse intitulée « communication avec des automates » à l'université de Bonn, les réseaux de Petri sont utilisés dans divers domaines, citons : L'informatique (calcul parallèle, calcul distribué,…), le domaine des communications (protocoles de communication), en entreprise (description organisationnelle, description du système de l'informatique), ainsi que dans le domaine de l'automatique dont ils ont été largement utilisés pour modéliser et simuler beaucoup de genres de systèmes et récemment quelques études dans le but de contrôle ont été menées vers cette direction, citons la contribution de Katerina Yamalidou et John Moody. Les réseaux de petri sont un outil très approprié à l'étude des systèmes à événement discrets.

Dans ce chapitre nous allons palier aux problèmes de l'explosion combinatoire posés dans l'approche de Ramadge et Wonham, par l'utilisation des réseaux de Pétri (Rdp) que nous allons détailler. Par la suite présenter une approche de synthèse de la commande basée sur les invariants de marquage (propriété structurelle des Rdp) qu'on va illustrer par un exemple d'application.

II.2. Définitions et représentations des réseaux de Petri [6][4]

II.2.1. Définition d'un réseau de Petri

Un réseau de petri (Rdp) est composé de places, transitions et arcs :

Une place est représentée par un cercle.

Une transition par un trait.

Un arc relie soit une place à une transition soit une transition à une place.

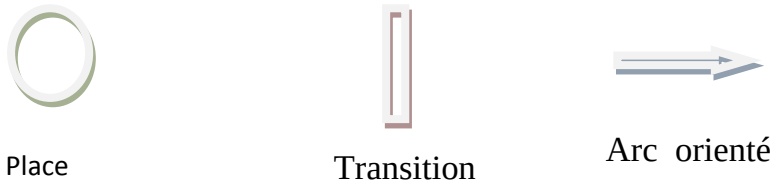


Figure II.12. Représentation graphique

Exemple de Rdp

Les places sont nommées P_1, P_2, P_3, P_4

Les transitions sont T_1, T_2, T_3

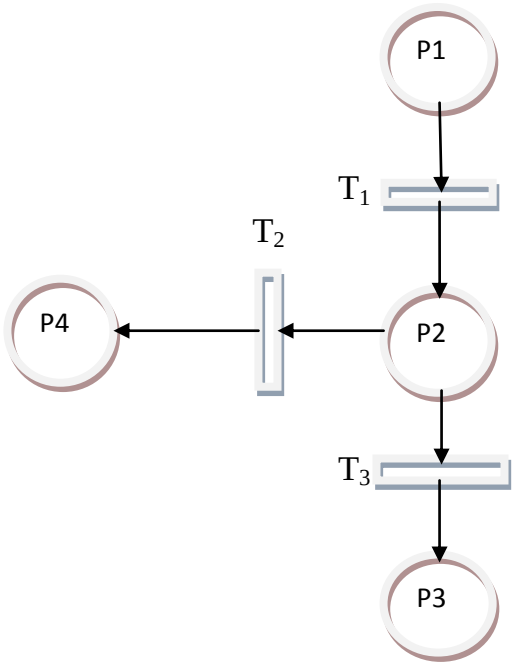


Figure II.13. Exemple de Rdp

II.2.2. Marquage d'un réseau de Petri

Définition1

Chaque place contient un nombre entier positif ou nul de marques ou jetons. Le marquage μ définit l'état du système à un instant donné. C'est un vecteur colonne de dimension égale au nombre de places.

Exemple

Le marquage de la place P_1 est : μ_1=1

Le marquage de la place P_2 est : μ_2=2

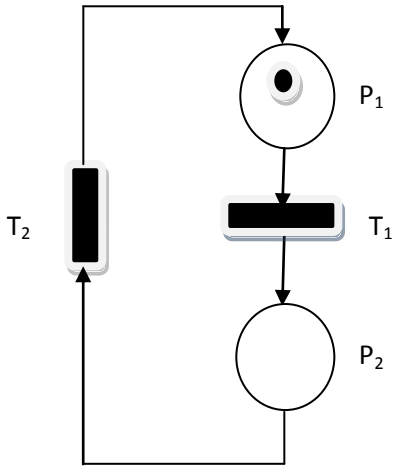


Figure II.14. Exemple de marquage

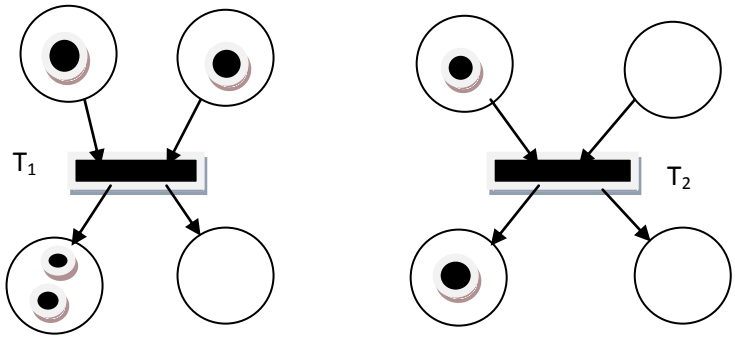
II.2.3. Evolution d'un Rdp

II.2.3.1. Franchissement ou tir d'une transition

Définition

Le franchissement consiste à retirer un jeton de chacune des places d'entrée et à rajouter un jeton à chacune des places de sortie de la même transition.

Exemple de franchissement



Transition franchissable transition non franchissable

Figure II.15. Exemple de franchissement

T_1 est franchissable ce qui n'est pas le cas de T_2 qui ne satisfait pas la condition de franchissement.

- Une transition franchissable n'est pas forcement immédiatement franchie.
- Une transition sans place d'entrée est toujours franchissable : c'est une transition source.
- Une transition sans place de sortie est une transition puits.

II.2.3.2. Séquence de franchissement

Définition

Une séquence de franchissement S est une suite de transitions T_i T_j…..T_k qui peuvent être franchies successivement à partir d'un marquage donné.une seule transition peut être franchie à la fois, notée : μ_i [S→ μ_j ou μ_i [S> μ_j

A partir du marquage μ_i, le franchissement de la séquence S aboutit au marquage μ_j.

Exemple

$\mu_0 T_1 T_2 \rightarrow \mu_1$ où $\mu_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

$\mu_0 T_1 T_3 \rightarrow \mu_1$ où $\mu_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$

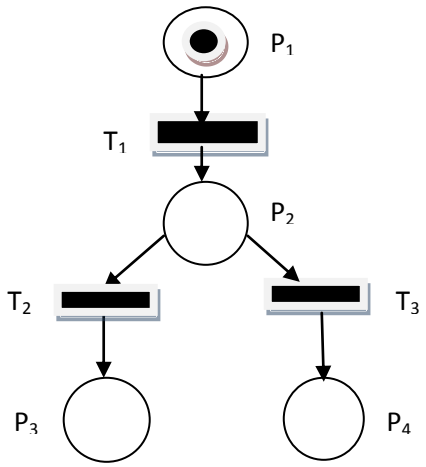


Figure II.16. Séquence de franchissement.

M_0 est le marquage initial, T_1 T_2 et T_1 T_3 sont deux séquences de franchissement.

II.2.3.3. Marquages accessibles et graphe des marquages

Definition1

L'ensemble des marquages accessibles est l'ensemble des marquages μ_i qui peuvent être atteints par le franchissement d'une séquence S à partir du marquage initial μ_0.

$\mu_0 = \{\mu_i / \mu_i\, S \rightarrow \mu_j\}$

Si on revient à l'exemple de la figure II.13 on aura :

$\mu^* = \{\mu_0, \mu_1, \mu_2, \mu_3\}$ est l'ensemble des marquages accessibles ou :

$\mu_0 = [1\ 0\ 0\ 0]^T$, $\mu_1 = [0\ 1\ 0\ 0]^T$, $\mu_2 = [0\ 0\ 1\ 0]^T$, $\mu_3 = [0\ 0\ 0\ 1]^T$

Définition2

On utilise le graphe de marquage quand le nombre de marquages accessibles est fini.

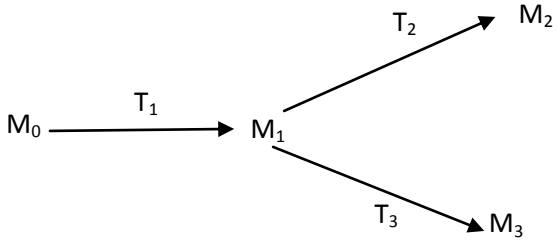


Figure II.17. Graphe de marquage de l'exemple

II.2.4. Les réseaux de petri particuliers

II.2.4.1. Graphe d'état

Un réseau de petri non marqué est un graphe d'état si et seulement si toute transition a exactement une seule place d'entrée et une seule place de sortie.

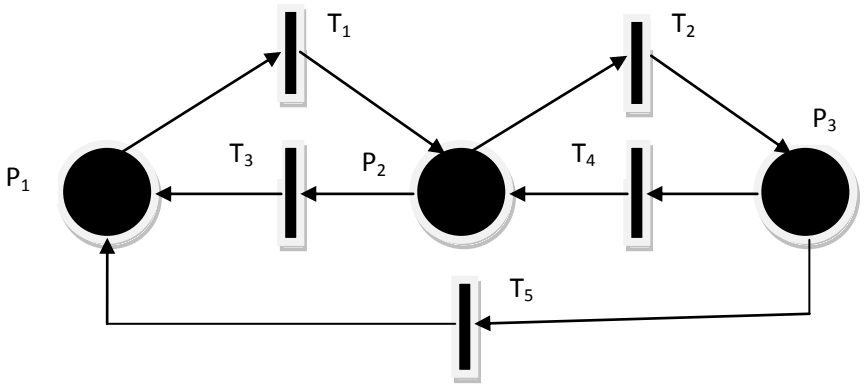


Figure II.18. Graphe d'état.

II.2.4.2. Graphe d'événement

Un Rdp est un graphe d'événement si et seulement si chaque place possède exactement une seule transition d'entrée et une seule transition de sortie.

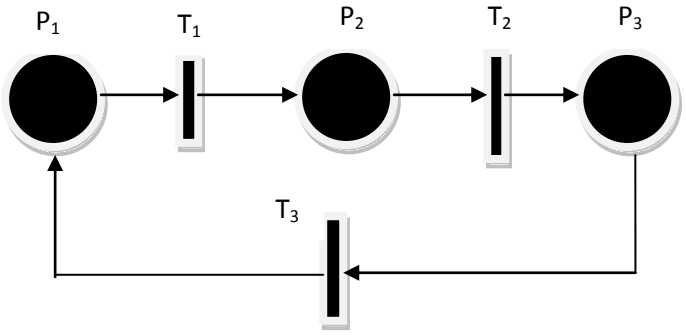


Figure II.19. Graphe d'événement

II.2.4.3. Rdp sans conflits

Un Rdp sans conflit est un réseau dans lequel chaque place a au plus une transition de sortie.

Un Rdp avec conflit est un réseau qui possède donc une place avec au moins deux transitions de sorties

Un conflit est noté : $[P_i, \{T_1, T_2, \ldots, T_n\}]$ avec $T_1, T_2, \ldots, T_n$ les transitions de sorties de la place P_i.

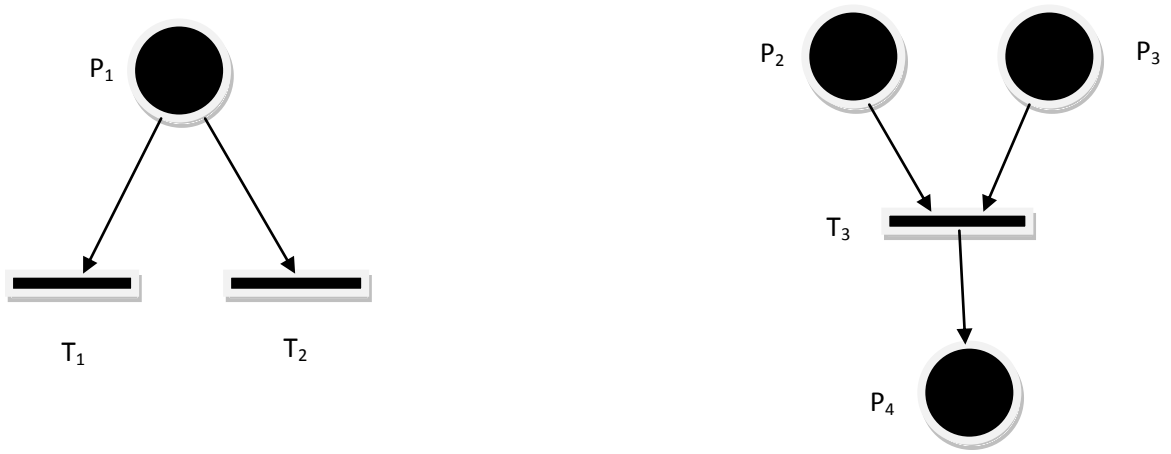


Figure II.20. Rdp avec conflit et Rdp sans conflit.

II.2.4.4. Rdp généralisé

Un Rdp généralisé est un Rdp dans lequel des poids (nombres entiers strictement positifs) associés aux arcs.

Si un arc (P_i, T_j) a un poids k : la transition T_j n'est franchie que si la place P_i possède au moins k jetons. Le franchissement consiste à retirer k jetons de la place P_i.

Si un arc (T_j, P_i) a un poids k : le franchissement de la transition rajoute k jetons à la place P_i.

Lorsque le poids n'est pas signalé, il est égal à un par défaut.

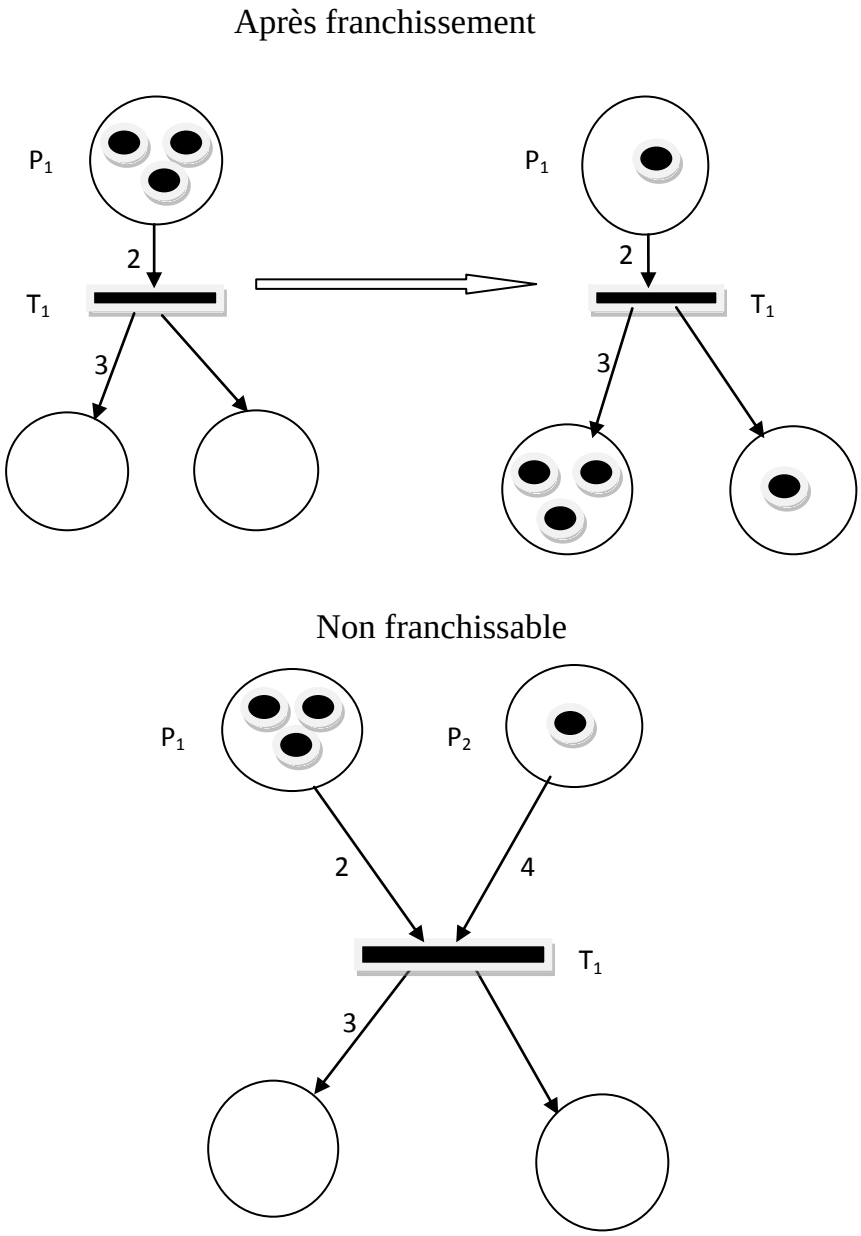


Figure II.21. Transition franchissable et transition non franchissable.

II.2.5. Propriétés des Rdps

Notions

Un marquage μ_1 « couvre » un marquage μ_2 si le marquage de chaque place P_i dans μ_1 est supérieur ou égale à son marquage dans μ_2 : $\mu_1 \geq \mu_2$, $\mu_1(P_i) \geq \mu_2(P_i)$, pour toute place P_i.

Un marquage μ_1 « couvre strictement » un marquage μ_2 si le marquage de chaque place P_i dans μ_1 est supérieur ou égale à son marquage dans μ_2 et avec au moins une place telle que :

$\mu_1 (P_i) > \mu_2(P_i)$:

$\mu_1 > \mu_2$, $\mu_1(P_i) \geq \mu_2(P_i)$, avec au moins une place P_i telle que

$\mu_1 (P_i) > \mu_2 (P_i)$

II.2.5.1. Rdp borné

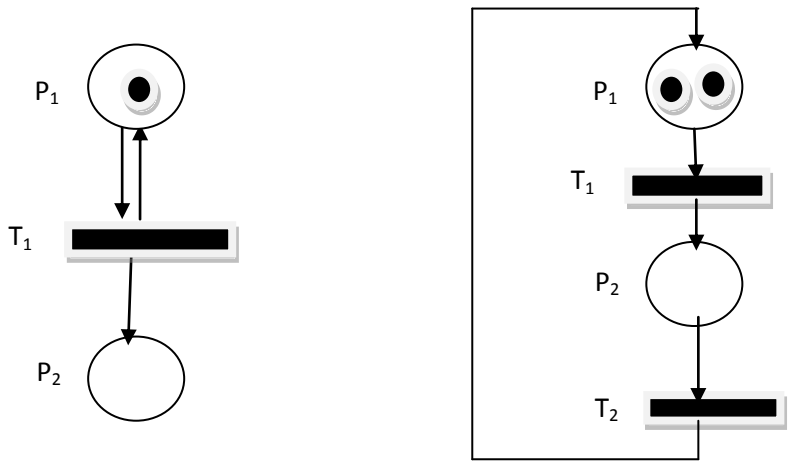
Une place est dite bornée pour un marquage initial μ_0, si pour tout marquage accessible à partir de μ_0 le nombre de marques dans P_i est fini.

Un réseau de petri est borné pour un marquage initial μ_0, si toutes les places sont bornées pour μ_0.

Pour tout marquage $\mu \in \mu^*$, si on $\mu(P_i) \leq k$, k étant un entier naturel. On dit que P_i est k-borné.

Un Rdp 1-borné est dit sauf(ou binaire).

Exemple de Rdp borné et non borné



Non borné borné

Figure II.22. Rdp borné et Rdp non borné

II.2.5.2. Vivacité d'un Rdp

Une transition T_j est vivante pour un marquage initial μ_0, si pour tout marquage accessible $\mu_i \in \mu^*$, il existe une séquence de franchissement S qui contient la transition T_j à partir de μ_i.

Un réseau de petri est vivant si pour un marquage initial μ_0, toutes ses transitions sont vivantes.

II.2.5.3. Quasi- vivacité

Une transition T_j est quasi-vivante pour un marquage initial μ_0, s'il existe une séquence de franchissement S qui contient la transition T_j à partir de μ_0.

Un Rdp est quasi-vivant si toutes ses transitions sont quasi-vivantes.

II.2.5.4. Blocage d'un Rdp

Un blocage (ou état puits) est un marquage tel qu'aucune transition n'est validée.

Un réseau de pétri est sans blocage pour un marquage initial μ_0, si aucun état accessible n'est un blocage.

Si une transition T_j est quasi-vivante pour un marquage initial μ_0, elle est quasi-vivante pour un marquage $\mu'_0 \geq \mu_0$.

Si une transition T_j est vivante pour un marquage initial μ_0, elle n'est pas nécessairement vivante pour un marquage $\mu'_0 \geq \mu_0$.

Si un Rdp est sans blocage pour un marquage initia $\mu 0$, il n'st pas nécessairement sans blocage pour un marquage $\mu'_0 \geq \mu_0$.

II.2.5.5. Rdp réinitialisable

Un Rdp a un état d'accueil μ_a pour un marquage initial μ_0 si pour tout marquage accessible $\mu_i \in \mu^*$ il existe Si tel que $\mu_i (S_i \rightarrow \mu_a$.

Un Rdp est réinitialisable pour un marquage initial μ_0 si μ_0 est un état d'accueil.

II.2.6. Notations et définitions

« Pré (P_i, T_j) » est le poids « k » de l'arc reliant une place à une transition.

$$\text{Pré } (P_i, T_j) = \begin{cases} k & \text{si l'arc } (P_i, T_j) \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

« Post (P_i, T_j) » est le poids « k » de l'arc reliant une transition à une place.

$$\text{Post } (P_i, T_j) = \begin{cases} k & \text{si l'arc } (T_j, P_i) \text{ existe} \\ 0 & \text{sinon} \end{cases}$$

On appelle « matrice d'incidence avant » :

$D^- = [\text{pré } (P_i, T_j)]$

On appelle « matrice d'incidence arrière » :

$D^+ = [\text{post } (P_i, T_j)]$

On appelle « matrice d'incidence » :

$D = D^+ - D^-$

II.2.7. Équation fondamentale ou équation d'état

-Soit S une séquence de franchissement réalisable à partir d'un marquage

$$\mu i : \mu_i[S > \mu_k$$

-soit $\underline{S}$ le vecteur caractéristique de la séquence S : c'est un vecteur de dimension m égale au nombre de transitions dans le réseau. Sa composante numéro j correspond au nombre de fois ou la transition T_j est franchie dans la séquence S.

Exemple

S=T_2 T_4 T_1 T_4 T_2 T_4 alors $\underline{S}$ = [1, 2, 0, 3] ^{T}

Si la séquence de franchissement S est tel que μ_i [S> μ_k alors l'équation fondamentale correspondante s'écrit :

$\mu_k=\mu_i+D*\underline{S}$

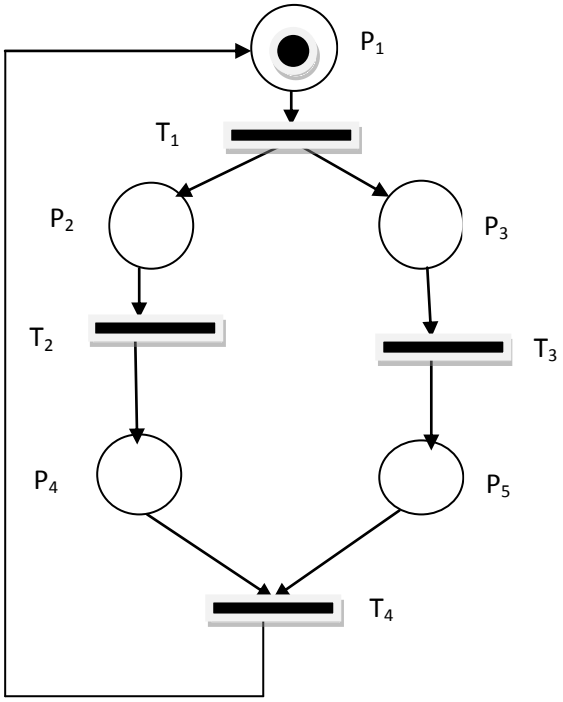


Figure II.23.

Soit S= T_1 T_2 donc $\underline{S}$ = [1, 1, 0, 0] ^{T}

$$D^- = \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0011 \\ 0001 \end{pmatrix} \quad D^+ = \begin{pmatrix} 0001 \\ 1000 \\ 1000 \\ 0100 \\ 0010 \end{pmatrix} \quad D = \begin{pmatrix} -1001 \\ 1-100 \\ 10-10 \\ 010-1 \\ 001-1 \end{pmatrix}$$

L'équation fondamentale correspondante à cette séquence est :

$\mu_2 = \mu_0 + D * \underline{S}$

$$\mu_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

II.3. Synthèse de la commande basée sur les invariants de marquages [9][10][11]

Cette approche est née des travaux de Yamalidou et Moody en formulant le problème de contrôle de processus chimiques d'événement discrets sous forme de problème d'optimisation linéaire basée sur les réseaux de petri.

Le contrôleur consiste en un ensemble de places qui sont reliées aux transitions du Rdp du procédé afin de garantir que le système ne tombe dans un état défendu. Le Rdp combiné du procédé/contrôleur possède des invariants de marquages nécessaires pour s'assurer que l'ensemble des contraintes ne sera pas violé.

Dans cette approche le contrôleur se compose de places et d'arcs calculés à l'aide du concept d'invariant de marquage, les spécifications sont données par des contraintes linéaires entre les marquages de places du Rdp.

II.3.1. Invariant de marquage

L'invariant de marquage est l'une des propriétés structurelle des Rdps, c'est-à-dire elle dépend seulement de la structure topologique du Rdp et pas du marquage initial du réseau.

Il représente un ensemble de places dont le compte symbolique reste toujours constant, il est défini par un vecteur X de n colonnes, où n : est le nombre de places du Rdp, le vecteur X satisfait l'équation suivante :

$$\mu^{T}.X=\mu^{T}_{0}\ .X \qquad \text{(II.1)}$$

μ_{0} : le marquage initial μ: représente tout marquage accessible.

L'équation (II.1) explique la somme pondérée des jetons des places de l'invariant de marquage qui demeure constante à tout marquage, cette somme est déterminée par le marquage initial.

L'invariant de marquage est défini par tout vecteur entier satisfaisant l'équation suivante :

$$X^{T}\ .D=0 \quad \text{(II.2)}$$

D : matrice d'incidence du Rdp de dimension $n \times m$, n est nombre de places et m : est nombre de transitions.

II.3.2. Description de la méthode

La méthode que nous décrierons exige les modèles Rdps que se soit pour le procédé à contrôler et que se soit pour les constructions du contrôleur lié au réseau du procédé. Le procédé doit satisfaire des spécifications décrites par des contraintes linéaires (inégalités et égalités).

Supposons que le système à contrôler est modélisé par un Rdp avec n places et m transitions et doit satisfaire la contrainte suivante :

$$\mu i + \mu j \leq 1 \qquad \text{(II.3)}$$

Où μi , μj sont les marquages des places P_i, P_j du Rdp, il est évident qu'au plus une des deux places soit marquées ; en d'autre termes, les deux places ne peuvent pas être marquées en même temps.

Cette inégalité peut être transformée en une égalité en introduisant une variable entière positive μc ; la contrainte devient alors :

$$\mu i + \mu j + \mu c = 1 \quad \text{(II.4)}$$

La variable μc dans ce cas représente la nouvelle place qui reçoit l'excès de jetons, donc s'assurer que la somme des jetons dans l'ensemble de places vérifie

$\mu i + \mu j \leq 1$, cette place appartient au réseau du contrôleur. L'introduction de la variable introduit un invariant de marquage pour tout système défini par l'équation (II.5). C'est évident qu'il y aura autant de places de contrôle que de contraintes de type (II.4), donc la taille du contrôleur est proportionnelle au nombre de contraintes de types (II.4).

En ajoutant une place de contrôle P_c au Rdp, nous obtenons une matrice augmentée D_c du système contrôlé en ajoutant une ligne correspondante à la place introduite par la variable positive, cette nouvelle ligne appartient au Rdp du procédé couplé au contrôleur.

Les arcs connectant les places du contrôleur au réseau original du système seront calculés par l'invariant de marquage de l'équation (II.2) ou les inconnus sont les éléments de la nouvelle ligne de la matrice D tel que le vecteur X_i d'invariant de marquage est défini par l'équation (II.4).

On général le problème peut être déclaré comme suit :

Toutes les contraintes de type (II.3) peuvent être regroupées en une matrice comme suit :

$$L.\mu p \leq b \quad \text{(II.5)}$$

Où μp : est le vecteur de marquage du Rdp du procédé, L est une matrice de dimension $nc \times n$ et b est un vecteur de $nc \times 1$, nc est le nombre de contraintes de types (II.3).

De la même façon, toutes les équations d'invariant de marquage de type (II.4), généré après l'introduction de la variable positive, peuvent être regroupées dans la matrice qui suit :

$$L.\mu p + \mu c = b \qquad \text{(II.6)}$$

Où : μc est un vecteur de $nc \times 1$ represantant les marquages des places de contrôle.

L'invariant de marquage défini par l'équation (II.4) doit satisfaire l'invariant de marquage de l'équation (II.1)

L'équation matrice suivante est l'équation invariant de marquage pour tous les invariants définis par l'équation (II.6)

$$X^{T}.D = \begin{bmatrix} L & I \end{bmatrix} \begin{pmatrix} Dp \\ Dc \end{pmatrix} = 0$$

$\Leftrightarrow L.Dp + Dc = 0$

$\Leftrightarrow Dc = -L.Dp$ (II.7)

Où I: matrice identité de dimension nc×nc. La matrice Dc contient les arcs qui permettent de connecter les places du contrôleur aux transitions du procédé. Etant donné le réseau du procédé (Dp) et les contraintes que doit satisfaire le procédé définies par (nc, L et b), le réseau de petri du contrôleur (Dc) est défini par l'équation (8)

Le marquage initial du Rdp du contrôleur devrait être aussi calculé. Le marquage initial du contrôleur μc_0 doit être tel que l'invariant de marquage défini dans l'équation (II.6) est satisfait et dépend du marquage initial des places du procédé qui participent dans les invariant de marquage.

De l'équation (II.1) et de l'équation (II.6) on peut écrire le vecteur de marquage initial :

$$L.\mu p_0 + \mu c_0 = b \Leftrightarrow \mu c_0 = b - L.\mu p_0 \quad \text{(II.8)}$$

II.3.3. Problèmes des transitions incontrôlables

Soit D_u la sous matrice représentant la partie incontrôlable. Elle contient les colonnes de D_p qui correspondent aux transitions incontrôlables. $D_u \in \mathbb{Z}^{m \times n_u}$, n_u est le nombre des transitions incontrôlables. Soit L D_u la sous-matrice de L D_p qui correspond aux transitions incontrôlables du procédé.

Et soit un ensemble de contraintes : $L\ \mu_p \leq b$. Le Rdp du contrôleur synthétisé par l'approche développée ici est donné par : $D_c = -L.D_p$

Le contrôleur viole les contraintes si L D_u contient au moins un élément strictement positif. En revanche, s'il existe un arc liant une place à une transition incontrôlable, on conclut que L D_u contient des valeurs positives, et les contraintes ne sont pas satisfaites. Pour résoudre ce problème Yamalidou a proposé une méthode intuitive qui consiste à remonter les branches jusqu'à trouver une transition contrôlable qui soit en aval de la place de contrôle. L'inconvénient majeur de cette méthode c'est qu'il n'y a pas d'algorithme qui permet le calcul systématique du contrôleur optimal.

Exemple :

L'exemple de la figure ci-dessous illustre l'utilisation de la commande. Supposons que l'on désir imposer au Rdp la contrainte suivante :$\mu(P_2)+\mu(P_3)\leq 1$. Supposons aussi que la transition T_2 est incontrôlable.

Le système contrôlé est donné dans la figure II.24, une place de contrôle a été ajoutée pour respecter la contrainte. Dans ce cas une solution consiste à remonter la branche jusqu'à la transition T_1 que l'on suppose contrôlable, ainsi on obtient le système final qui correspond au superviseur optimal.

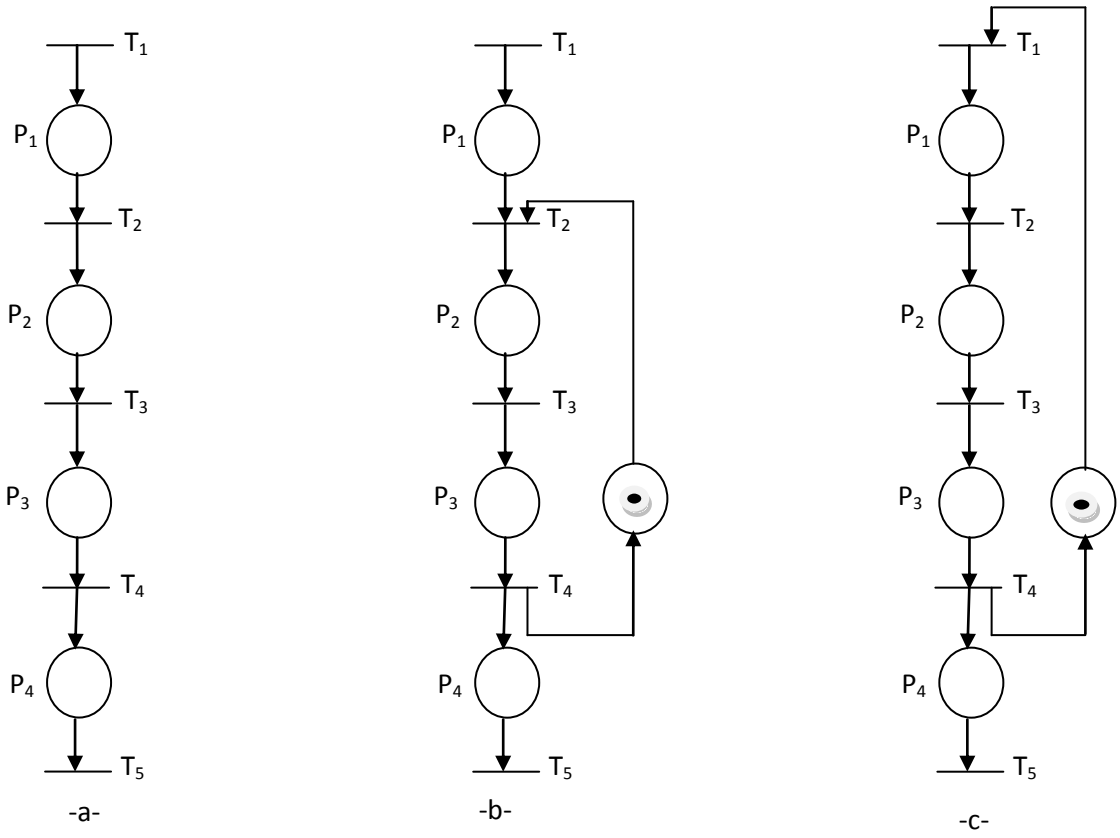


Figure II.24. a)système à contrôler b) système avec place de contrôle c)système contrôlé final.

II.4. Exemple d'application

Considérons l'exemple de chat et de la souris introduit par Ramadge et Wonham.

Le problème implique un labyrinthe de cinq pièces ou un chat et une souris peuvent circuler. Les pièces sont connectées par des portes permettant au chat et à la souris de passer d'une pièce à l'autre comme le montre la figure suivante :

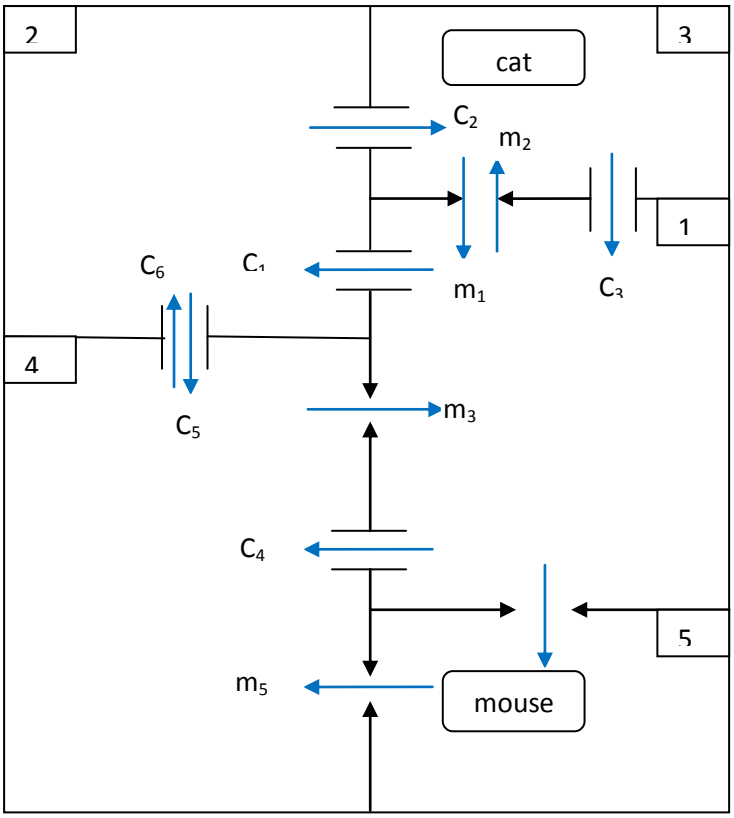


Figure II.25. Exemple de labyrinthe ou circule le chat et la souris.

Le problème est de contrôler les portes afin que le chat et la souris ne puissent se retrouver dans la même pièce au même temps. Sachant que le chat ne peut passer dans la pièce 5 et que la souris ne peut passer dans la pièce 2.Le contrôleur doit accorder une liberté maximale des mouvements dus à la fois au chat et à la souris en respectant les spécifications. Le modèle réseau de petri du problème est représenté dans la figure.

Les parties gauche et droite concernent respectivement le mouvement du chat et celui de la souris.

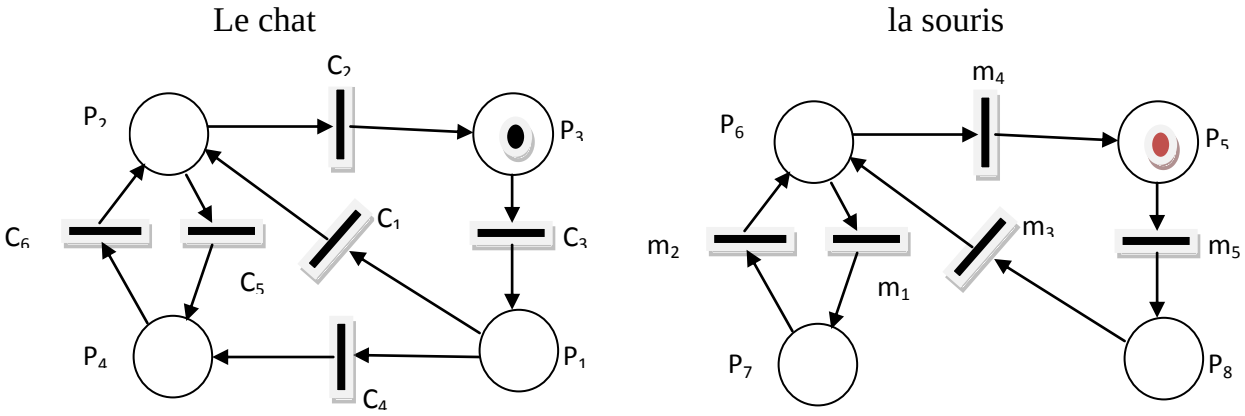


Figure II.26. Le RDP décrivant les mouvements du chat et la souris.

La matrice d'incidence du modèle réseau de patri est :

$$Dp = \begin{pmatrix} -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

Le marquage initial : $\mu p_0 = [\mu 1 \quad \mu 2 \quad \dots \quad \mu 8j]^T$

$\mu p_0 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]^T$

Supposons que toutes les portes sont contrôlables.

L'objectif du contrôleur est de s'assurer que le chat et la souris ne se retrouvent jamais dans la même pièce. Cela signifie que chaque paire de places des deux parties

du réseau de petri décrivant les mouvements du chat et de la souris appartenant à la même pièce ne doivent jamais contenir un jeton au même temps.

Cette figure traduite par les trois contraintes sous formes d'inégalité que nous transformerons en égalités en ajoutant des variables d'écart.

$$\mu 4 + \mu 8 \leq 1 \Longrightarrow \mu 4 + \mu 8 + \mu c1 = 1$$

$$\mu 3 + \mu 7 \leq 1 \Longrightarrow \mu 3 + \mu 7 + \mu c2 = 1$$

$$\mu 1 + \mu 6 \leq 1 \Longrightarrow \mu 1 + \mu 6 + \mu c3 = 1$$

Soit L la matrice de contrainte :

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

La matrice d'incidence du contrôleur est:

$Dc = -LDp$

$$Dc = \begin{bmatrix} 1 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}$$

Le marquage initial des places de contrôle est donné par :

$$\mu c0 = 1 - L\mu p0$$

$$\mu c0 = [1 \quad 1 \quad 0]^T$$

Les mouvements du chat et de la souris sous les contraintes et représenté dans la figure suivante :

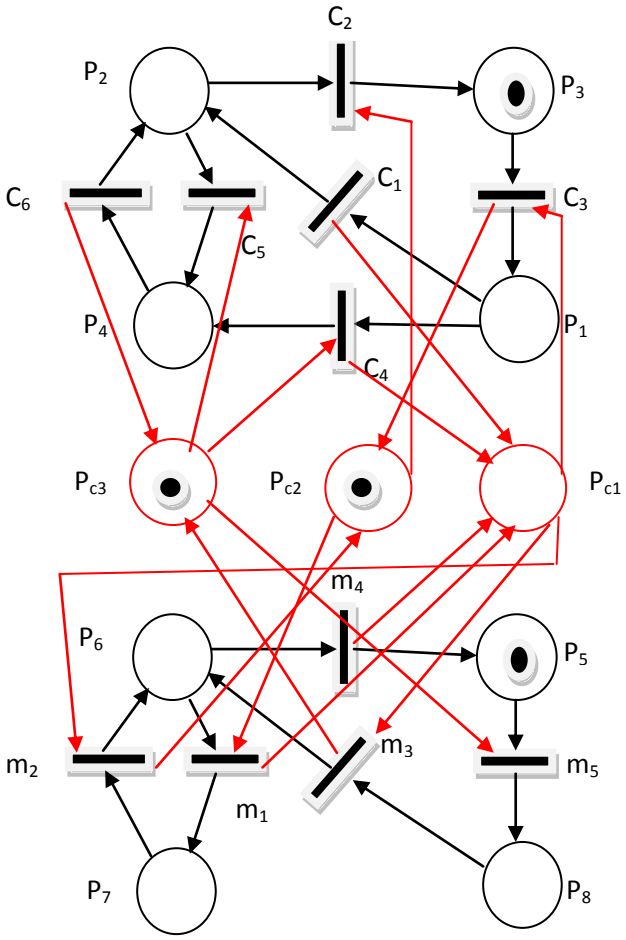


Figure II.27. Le réseau de petri des mouvements du chat et la souris sous contrôleur.

II.5. Conclusion

Nous avons présenté dans ce chapitre, les réseaux de petri, leurs propriétés et quelques classes particulières de réseaux. Nous avons présenté également, la méthode de synthèse de superviseur basée sur les invariants de marquage. Cette méthode apporte une réponse aux problèmes de l'explosion combinatoire présente dans la modélisation par les automates. Malgré le fait qu'elle ne garantie pas l'optimalité, l'approche basée sur le concept des invariants de marquage demeure la plus prometteuse. Pour résoudre le problème d'incontrôlabilité des transitions Yamalidou a proposé une méthode intuitive qui n'est pas toujours général.

III.1. Introduction

La complexité croissante des processus industriels à suscité le besoin de créer un outil graphique d'analyse et de synthèses des automatismes industriels. Ainsi, le Grafcet (GRAphe Fonctionnel de Commande Etape /Transition) a été défini en 1975-1977 par les travaux de l'AFCET (Association Française pour la Cybernétique Economique et Technique), repris et soutenus par l'association pour le développement de la production automatisée normalisée en France en 1990 et normalisée sur le plan européen ainsi que sur le plan international en 1992. Il fournit selon Arnaud Hubert [16] : un outil de représentation et d'analyse de système à événements discrets dont les évolutions peuvent s'exprimer séquentiellement :

- Le Grafcet est indépendant de la matérialisation technologique du système de contrôle, que celui-ci soit câblé (électromagnétique, pneumatique ou électronique : désormais utilisé uniquement pour les systèmes à couts très réduits et /ou de très grandes séries) ou programmé (API, microcontrôleur ou ordinateur)
- il traduit de façon cohérente le cahier des charges de la commande séquentielle en obligeant même parfois celle-ci à être précisée.
- il peut prendre en compte des évolutions simultanées ou des choix entre plusieurs séquences.
- il est bien adapté au cas des systèmes faisant intervenir un grand nombre de variable d'entrée.

Nous allons présenter dans ce chapitre la synthèse d'un superviseur basée sur l'outil Grafcet, proposer la technique de passage d'un modèle automate à un grafcet en appliquant la théorie de Ramadge et Wonham dans le cas des spécifications incontrôlables afin de présenter le problème lié à cette approche.

III.2. Définition et représentation[18]

Le grafcet est un graphe orienté bipolaire articulant deux types de sommets :

III.2.1. Les étapes

Elles représentent les phases stables du cycle de fonctionnement d'un système, ayant un état actif ou non. A un instant donné, la situation du SED, c'est-à-dire l'état de ce système est entièrement défini par l'ensemble des étapes actives.

Les étapes sont représentées par des carrés repérés par des chiffres placés au centre. Les étapes initialement actives au début du fonctionnement sont représentées par un double carré.

Une étape est associée à une variable logique xi qui vaut 1 lorsque cette étape est active et 0 sinon.

Une étape peut valider plusieurs transitions qui peuvent déclencher, simultanément, lançant plusieurs processus.

III.2.2. Les transitions

Elles représentent les phases transitoires du cycle de fonctionnement d'un système de durée nulle.

Une transition sera validée si toutes les étapes qui la précédent sont actives.

Une transition sera franchissable si sa condition propre (ou réceptivité) est vraie.

La transition devenue valide et franchissable se déclenche, désactivant les étapes immédiatement en amont et activant celle en aval.

Une transition est représentée par un trait plein rectiligne généralement horizontal.

Le schéma de la figure1 illustre ces représentations :

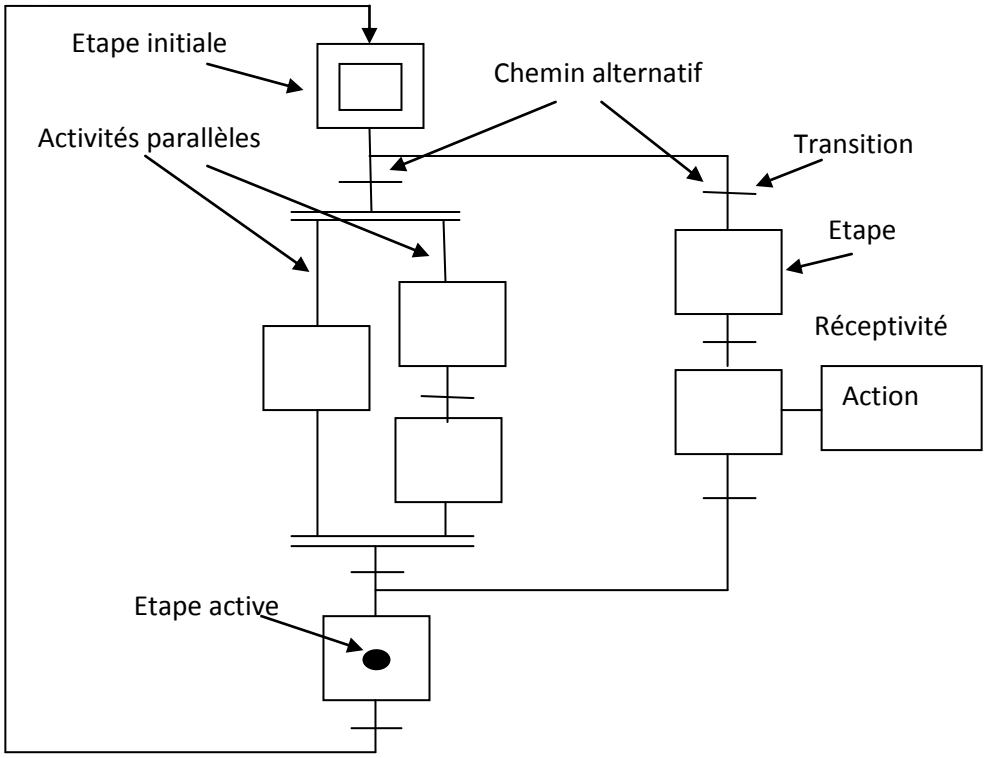


Figure III.1. Concepts de base du Grafcet[16].

III.2.3. Interprétation graphique

[16]Lorsqu'on associe l'action à une étape et une réceptivité à une transition on définit les taches du système ainsi que les conditions de leurs exécutions.

- Les actions : l'action spécifie la tache du système lors de l'activation de l'étape lui associée. Une action peut être interne (capteur) ou externe (sortie de l'automate)
- Les réceptivités : la réceptivité représente la condition propre du franchissement de la transition pouvant prendre les valeurs vraies ou fausses, elle dépend des entrées et/ ou aux états internes et/ou du temps. La réceptivité est active soit sur un niveau soit sur un front.

III.2.4. Niveau de description et de spécification fonctionnelle[16][7]

Nous pouvons décrire le fonctionnement d'un système modélisé par un grafcet par deux étapes, la première étape de description est appelée l'étape de spécification fonctionnelle et conduit à la définition d'un grafcet fonctionnel, la seconde étape de description est appelée l'étape de spécification opérationnelle et conduit à la définition d'un grafcet opérationnel.

- spécification fonctionnelle

A ce niveau de description, les spécifications doivent préciser :

-le dispositif à automatiser sous forme d'un dessin simplifié ou d'un schéma fonctionnel.

- la fonction assurée par chaque partie ou chaque constituant du dispositif.

- les sécurités éventuellement envisagées.

A ce stade, le grafcet doit respecter seulement le fonctionnement logique du système adaptant un langage proche du langage courant et indépendant du choix technologique. Les actions sont souvent décrites par des verbes à l'infinitif, alors que les réceptivités sont décrites par des verbes au participe passé.

- Spécification opérationnelle

C'est à ce niveau de spécifications que sont prises les options relatives aux choix :

-des actionneurs, de leur près-actionneurs et de leur mode de contrôle (monocontrôlable, bistable, continu….)

- des capteurs (modes et type de fonctionnement, type d'informations retournées par les conditionneurs de signaux).

Le choix des actionneurs et des capteurs est souvent restreint par les conditions d'environnement (température, humidité, poussière, chocs,…) et d'alimentation (électrique, pneumatique, hydraulique).

-Dans un grafcet opérationnel, les actions et les réceptivités sont des variables symboliques (logiques ou numériques).

III.2.5. Règle d'évolution d'un grafcet

[14]Les conditions d'évolutions du grafcet seront spécifiées en respectant les 5 règles résumées ci-dessous :

Regle1 : *l'initialisation* :

Elle précise l'étape ou les étapes actives au début du fonctionnement. Cette initialisation spécifie l'état initial du SED.

L'étape ou les étapes initiales étant actives inconditionnellement en début de cycle.

Regle2 : *validation et franchissement d'une transition :*

Une transition est validée lorsque toutes les étapes immédiatement précédentes sont actives.

La transition ne peut être franchie que si :

-elle est validée.

-la réceptivité associée à la transition est vraie.

Règle 3 : *activation et désactivation des étapes :*

Le franchissement d'une transition entraine simultanément l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

Regle4 : *évolution simultanée :*

Plusieurs transitions franchissables sont simultanée franchissables sont simultanément franchies.

Regle5 : *activation et désactivation d'une étape :*

Si au cours du fonctionnement, une même étape doit être activée et désactivée simultanément elle reste activée.

III.2.6. Validation du Grafcet

Dans de nombreuses méthodes de spécification ou de conception utilisant le Grafcet, il est recommandé d'utiliser quelques techniques pour la validation des modèles. La technique la plus importante est la validation par génération du graphe des situations accessibles Les approches proposées par Blanchard et Roussel 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 qui 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 Leparc.

Ces approches sont donc basées sur l'exploitation du graphe des situations accessibles, ce qui permet de définir la notion de situation atteinte ou 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 [3].

III.3. Synthèse d'un superviseur basée sur le grafcet

Ramadge et Wonham ont décris le comportement du système à évènement discrets par un procédé générant des évènements spontanés, instantanés et asynchrones couplé à un superviseur qui a pour rôle l'autorisation et l'interdiction des évènements contrôlables. De

plus le procédé supervisé, ainsi constitué est considéré comme étant isolé de son environnement.

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. 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 [1].

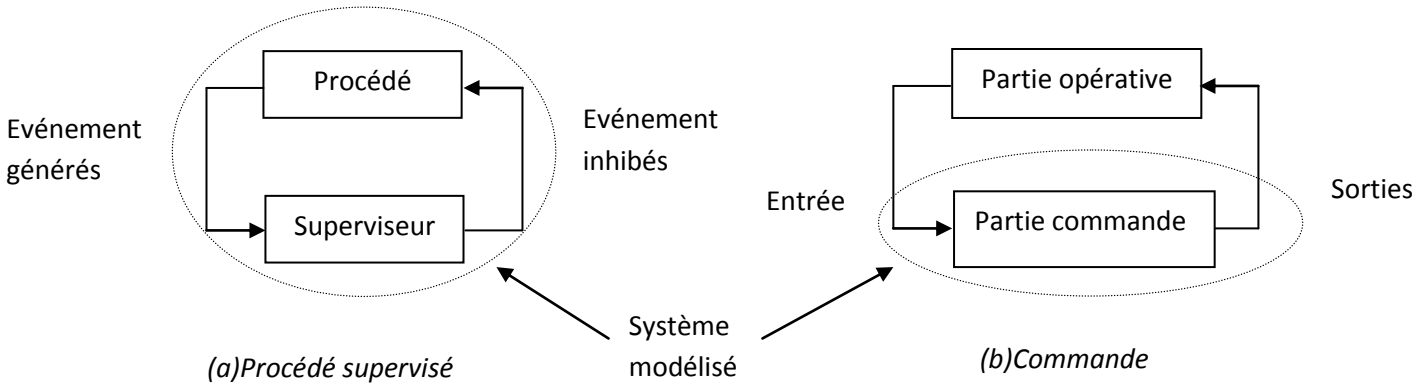


Figure III.2. Les systèmes modélisés.

L'écart sémantique des deux descriptions introduites par Kattan [1] empêche l'utilisation d'un modèle de procédé supervisé sous forme d'automate par exemple, comme étant un modèle d'une commande opérationnelle.

A partir de la supervision, la conception de la commande nécessite la construction de deux modèles : un modèle du procédé supervisé (sous la forme d'un automate à états par exemple), et un modèle de la commande opérationnelle(en utilisant le grafcet).

[1]Afin de réduire l'écart sémantique des deux modèles construits, de nombreuses approches sont proposées :

- L'approche de Brandi et Balemi basée sur le concept d'évènement forcé : dans cette approche, le superviseur ne joue plus le rôle d'inhibiteur d'événements générés par le procédé mais plutôt force certains évènements à se produire dans le procédé. De plus dans le cas d'un superviseur modulaire, pour qu'un évènement contrôlable soit forcé il suffit qu'il le soit par un des superviseurs ce qui n'est pas le cas dans l'approche R&W tel que un évènement contrôlable peut être généré par le procédé si et seulement si il est autorisé par l'ensemble des superviseurs.

- La commande supervisée proposée par Charbonnier et qui revient à considérer que le procède est constitué d'un procédé à commander et un système de commande. Ainsi, on constitue le procédé étendu générateur d'événement en interaction avec le superviseur.

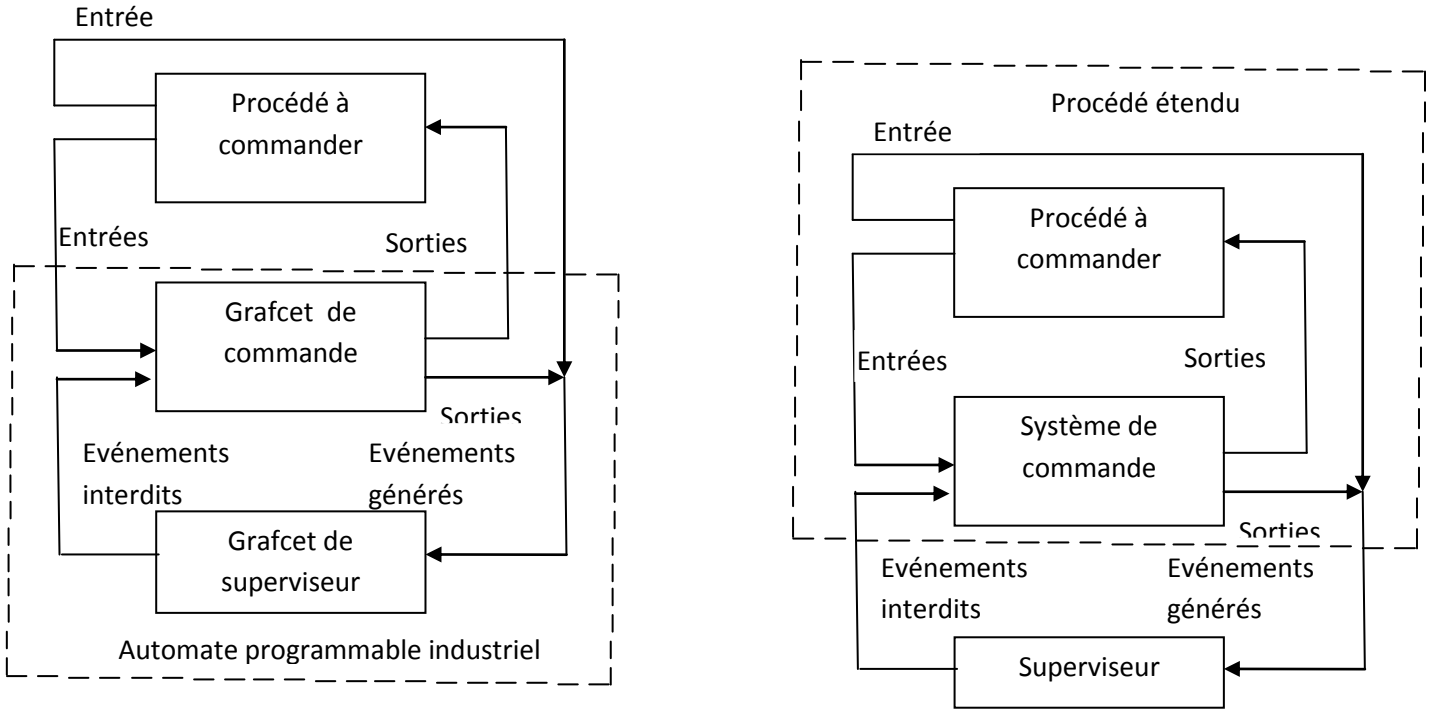
- Approche sureté de fonctionnement : la spécification du comportement nominal du système, la conception d'un système automatisé de production nécessite de prévoir des modes de fonctionnement d'exceptions. Ces modes doivent assurer la sécurité du système, son environnements dont l'operateur, tout en en augmentant la durée de disponibilité du système : c'est l'approche sureté de fonctionnement.

Ces approches montre que non seulement la théorie de R&W n'est pas directement implantable sous 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 Grafcet. Ceci n'enlève rien à la théorie de R&W mais montre simplement que la distance sémantique entre cette théorie et la commande opérationnelle ne peut être franchie par des approximations ou des simplifications.

Dans la suite, nous allons détailler l'approche de charbonnier, dans son travail il a pu modéliser le superviseur par le grafcet dans le cas ou les spécifications sont contrôlables, par la suite Kattan a conçu le modèle grafcet du superviseur dans le cas des spécifications non contrôlables.

III.3.1. La commande supervisée

[1]Dans cette approche, le système de commande peut forcer des évènements à ce produire dans le procédé et le superviseur garde sa fonction originelle définie dans la théorie de R&W. Comme nous le constaterons cette approche permet de clarifier les notions d'entrées/sorties et de systématiser le passage de la synthèse à l'implantation de la commande.



a) Pratique b) conceptuelle

Figure III.3. La commande supervisée.

[1]Dans cette approche, on définit le procédé étendu comme étant le procédé couplé à son système de commande. Pour un observateur extérieur, le procédé étendu est perçu comme un SED qui évolue spontanément en générant des évènements. Si l'on souhaite imposer un ensemble de spécifications logiques de fonctionnement au procédé étendu, celui-ci peut être couplé à un superviseur. Le superviseur observe de façon asynchrone l'ensemble des évènements qui sont générés dans le procédé étendu, c'est-à-dire l'alphabet $\sum$. A l'instant i, le superviseur fournit au procédé étendu une liste d'évènements interdits $\phi^{(i)}$, cela signifie que le procédé étendu peut uniquement évoluer en générant des évènements de $\sum / \phi^{(i)}$.

Nous désignons respectivement par $\sum_{Pr}$ et $\sum_{Co}$ les sorties du procédé et celles du système de commande. De plus, nous noterons l'alphabet : $\sum=\sum_{Pr} \cup \sum_{Co}$.

[1]Le schéma de commande supervisé est représenté dans la figure III.4.dans ce schéma le procédé est perçu par le système de commande comme un SED qui génère des évènements de $\sum_{Pr.}$ de même, le système de commande est considéré comme un SED qui produit des évènements de $\sum_{Co}$ .le superviseur perçois le procédé étendu comme étant un SED qui génère spontanément des événements de $\sum=\sum_{Pr} \cup \sum_{Co}$.

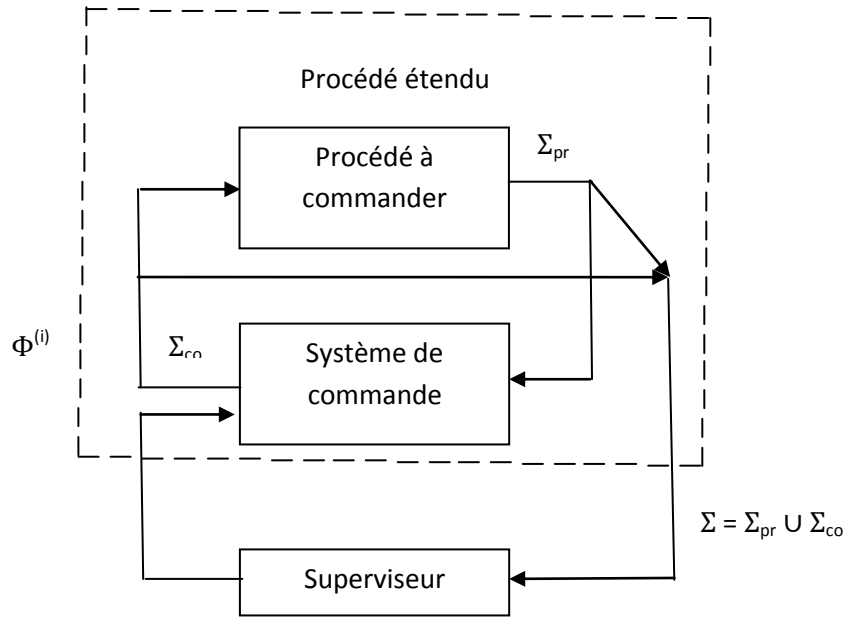


Figure III.4. Schéma de commande supervisée

Remarques

Dans la figure III.4, le superviseur interdit l'occurrence de certains évènements gardant ainsi, sa fonction originelle définie dans la théorie de R&W.

La commande et le superviseur sont clairement séparés. C'est-à-dire que la tache de la commande est indépendante de celle du superviseur. Pour cela, si l'on modifie les spécifications de la supervision, le système de commande ne nécessite pas d'être modifié.

Les sorties du procédé ne peuvent pas être interdites par le superviseur, elles seront donc considérées comme des évènements incontrôlables, c'est-à-dire $\sum_{Pr} \subseteq \sum_{u}$.

Par contre le superviseur peut toujours interdire au système de commande de produire un évènement de $\sum_{Co}$, ainsi, l'ensemble des sorties $\sum_{Co}$ est considéré comme un ensemble d'évènement potentiellement contrôlables.

Pour un problème donné, il ne sera pas nécessaire d'interdire l'occurrence de tous les évènements de $\sum_{Co}$. De façon générale, l'ensemble des évènements contrôlables $\sum_c$ sera donc défini comme un sous ensemble de $\sum_{Co}$, c'est-à-dire $\sum_c \subseteq \sum_{Co}$. L'ensemble des évènements incontrôlables sera alors défini par $\sum_{u=}\sum/\sum_c$ (ou $\sum/\sum_c$ représente l'ensemble des évènements qui appartiennent à $\sum$ et qui n'appartiennent pas à $\sum_c$.

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[1].

La synthèse d'une commande supervisée peut être basée sur les deux étapes suivantes[1] :

1) La synthèse de commande (s'il n'existe pas déjà) qui définit la spécification de la commande.
2) La synthèse du superviseur qui définit spécification du superviseur.

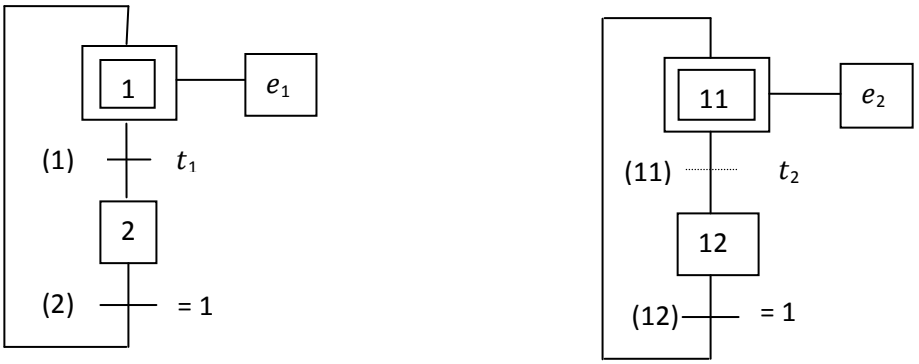
Afin de bien représenter clairement les sorties et les entrées de notre système, Charbonnier ainsi que Kattan ont procuré l'outil grafcet par son puissant pouvoir de décrire des systèmes logiques séquentiels et son habilité de l'utiliser comme langage dans les automates programmables, pour la synthèse de la commande supervisée et également son implantation.

Dans notre travail la spécification de commande ainsi que celles de la supervision seront modélisées par l'outil Grafcet. Par contre la preuve de contrôlabilité sera basée sur des modèles automates. Pour illustrer la démarche de la synthèse d'un superviseur on prendra le même exemple des deux serveurs traité dans le premier chapitre.

III.4. Exemple d'application

Considérons notre système de communication du premier chapitre dans la figure I.4. Le système comprend deux serveurs S_1 et S_2 pouvant envoyer et recevoir des messages indépendamment l'un de l'autre.

III.4.1. Spécification de commande



(a) (b)

Grafcet de commande de S_1 Grafcet de commande de S_2

Figure III.4. Spécification de commande.

La figure III.4 modélise les spécifications de commande des deux serveurs.

A l'instant initial, l'étape 1 est activée et l'action e_1 est exécutée (S_1 envoie un message). L'occurrence de l'événement t_1 (transmission du message au canal) conduit le grafcet dans l'étape 11. La transition (2) qui est franchie immédiatement ramène le grafcet dans son état initial. Le serveur S_2 a le même fonctionnement.

Au franchissement des transitions (1) et (11) on pourra considérer la modélisation est complète ce qui nous amène à considérer que les étapes (2) et (12) et les transitions (2) et (12) sont inutiles, ainsi on obtient un grafcet à une seule étape et une seule transition. Ce qui nous amène à conclure que dans le cas de notre système qu'à l'occurrence de t_1, l'étape 1 est activée et désactivée et qu'elle demeurera active et l'action impulsionnelle e_1 ne serait jamais exécutée, et finalement la modélisation représentera mal le cahier des charges.

L'ensemble produit par le procédé est $\sum_{Pr}= \{t_1, t_2\}$ et l'ensemble des évènements produit par la commande est $\sum_{Co}= \{e_1, e_2\}$.

Remarque

Le temps de franchissement d'une transition d'un grafcet est supposé petit, mais non nul. Cependant pour un observateur extérieur (c'est-à-dire, pour le superviseur), nous considérons que les évènements t_1 et e_1 se produisent simultanément. Rappelons que les observations du procédé par le système de commande et par le superviseur (figure III.4) sont asynchrones. Ainsi, dans notre exemple, le système de commande et le superviseur observaient de façon simultanée l'occurrence des évènements t_1 et e_1. Si le superviseur n'est pas un système plus rapide que le système de commande, alors il ne peut pas interdire l'occurrence de l'évènement de e_1 (sortie de la commande) après l'observation de t_1. Du point de vue du superviseur, l'occurrence de t_1 apparait comme étant indissociable de celle de e_1.

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

III.4.2. Spécification de supervision

Notre système de communication doit respecter la présence d'un canal entre les deux serveurs de capacité limitée à un la réception d'un seul message à la fois.

Le serveur S_1 qu'on désignera d'expéditeur doit envoyer un message vers le serveur S_2 qui sera ainsi le destinataire passant par le canal. Cela signifie que la présence d'un message au niveau du canal permet au S_2 de recevoir le message (donc débuter son cycle), et que S_1 ne peut envoyer de message que si le canal est vide. Le grafcet de la figure III.5 permet de modéliser cette spécification.

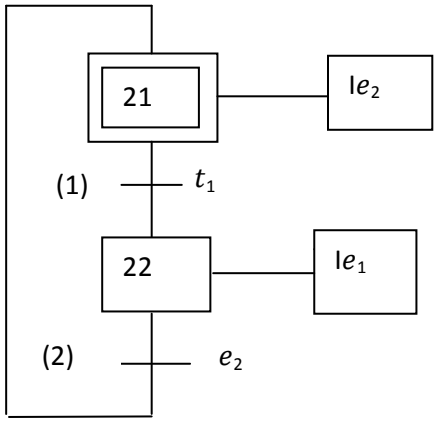


Figure III.5. Grafcet de spécification de supervision

Dans son état initial, l'étape 3 est active. L'occurrence de l'événement t_1 (transmission du message dans le canal) conduit le grafcet dans l'étape 33. Dans cette étape l'occurrence de e_2 (début de travail de S_2) ramène le grafcet dans son état initial. Nous pouvons remarquer que les états {3} et {33} correspondent respectivement aux états du système : « présence d'un message au niveau du canal » et « absence de message dans le canal ».

Le grafcet comprend aussi deux sorties booléennes : Ie_2 et It_1 dans l'état {3}, l'action à niveau Ie_2 associée à l'étape 3 est exécutée, c'est-à-dire Ie_2=1. Cela signifie que dans cet état le canal ne contient pas de message donc l'occurrence de l'évènement e_2 n'est pas tolérée. De façon similaire, on constate qu'à It_1=1 l'occurrence de l'évènement t_1 n'est pas toléré.

Les sorties du procédé étendu sont des évènements de l'alphabet $\sum=\sum_{Pr} \cup \sum_{Co.}$

Ainsi les entrées du grafcet de la figure III.5 sont, soit des évènements de $\sum_{Pr}$ (par exemple l'évènement t_1), soit des évènements de $\sum_{Co}$ (par exemple l'événement t_1). Les sorties du grafcet sont des grandeurs booléennes Ie_2 et It_1 égale à la valeur à 1 signifient que les évènements e_2 et t_1 sont pas tolérés par la supervision.

Remarque

Nous pouvons remarquer que le superviseur peut interdire en sortie que des évènements de $\sum_{Co}$. dans notre système les sortie du grafcet sont It_1 et Ie_2, ou l'évènement e_2 est potentiellement contrôlable donc peut être interdit par le superviseur contrairement à l'évènement t_1 qui est incontrôlable donc ne peut être interdit par la supervision. Lorsque la grandeur booléenne It_1 =1 cela signifie seulement que l'occurrence de l'évènement t_1 n'est pas tolérée.

Formellement, on peut définir le grafcet de spécification de supervision comme suit :

Définition

Un grafcet de supervision est un grafcet tel que ses entrées sont des évènements de $\sum$ et tel que ses sorties sont des grandeurs booléennes Iσ ou $\sigma\ est$ un evenement de $\sum$. Pour tout état du grafcet, l'événement σ est toléré si la grandeur booléenne Iσ a la valeur 0[1].

III.4.3. Synthèse de la commande supervisée

Les spécifications de supervision modélisées par le grafcet de la figure.4 interdisent l'occurrence de l'évènement E_2 lors de la présence d'un message au niveau du canal ainsi que l'occurrence de l'évènement T_1 lors de l'absence de message dans le canal. Il sera nécessaire de modifier le grafcet de la figure.3 en introduisant les spécifications de la supervision ainsi conditionnant les sorties que l'on souhaite pouvoir interdire. Un tel système appelé procédé étendu sous supervision.

Remarque

L'événement incontrôlable t_1 ne peut être interdit par le superviseur, pour cela on va interdire l'occurrence de l'évènement contrôlable e_1 lors de présence de message dans le canal de manière à pouvoir garantir que l'évènement t_1 ne puisse être généré par le procédé.

III.4.3.1. Procédé étendu sous supervision

A partir du grafcet de commande, nous pouvons remarquer que les sorties e_1 et e_2 ne sont pas conditionnées, or le superviseur doit interdire à tout instant l'occurrence de l'un des évènements contrôlables e_1 et e_2. Pour ce faire le grafcet de la commande devra être transformé conformément à la figure III.6.

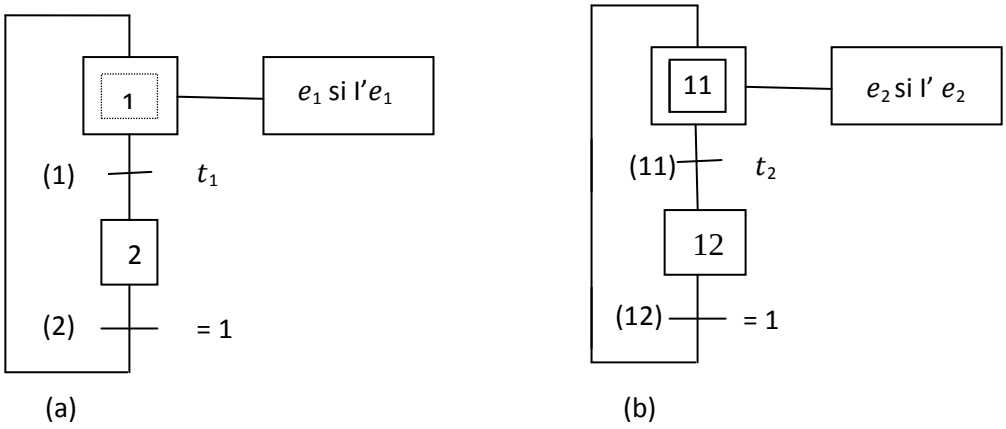


Figure III.6. Procédé étendu sous supervision, obtenu à partir de la figure III.4.

Considérons le grafcet de la commande de S_1 figure III.4 (a) l'action impulsionnelle e_1 associée à l'étape 1 du grafcet de commande de S_1 a été remplacé par l'action impulsionnelle conditionnelle : e_1 et I$'e_1$ dans la figure III.6 (a) avec I$'e_1$ est le complémentaire de Ie_1. Dans la figure III.6 (a), l'action e_1 ne peut être exécutée que si seulement si la condition associée est vraie, c'est-à-dire I'e_1=1 (ou de façon équivalente Ie_1=0, signifiant que l'évènement e_1 n'est pas interdit). De même pour l'action e_2 associée à l'étape 2 du grafcet de la commande de S_2 a été remplacé par l'action e_2 si I'e_2 dans la figure III.6.(b).

Définition

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 III.7[1].

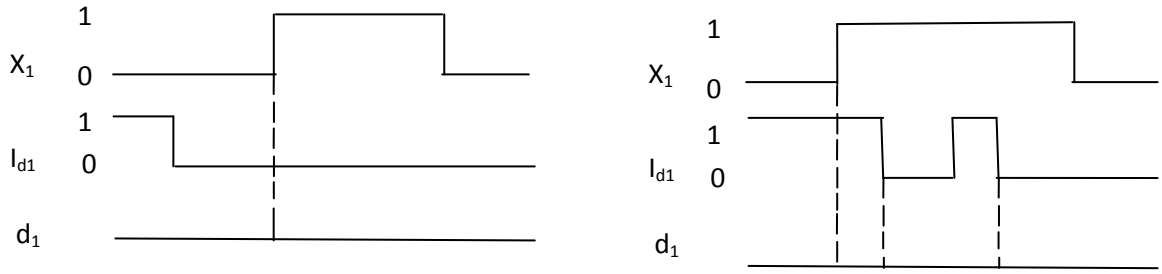


Figure III.7. Exécution de l'action impulsionnelle d_1 si I'd_1

Pour obtenir un modèle automate du procédé étendu sous supervision à partir du grafcet, Charbonnier a proposé une règle de construction de grafcet en ajoutant une étape en aval de l'étape conditionnée liée par une transition qui est la condition elle-même en maintenant le même comportement entrées/sorties du grafcet du procédé étendu sous supervision.

Pour illustrer cette règle de constructions, nous considérons le grafcet du serveur S_1. On ajoute une transition (1′) et une étape 1′ en aval de l'étape 1 et on associe l'action impulsionnelle (non conditionnée) e_1 à l'étape 1′, nous obtenons le grafcet de la figure III.7.

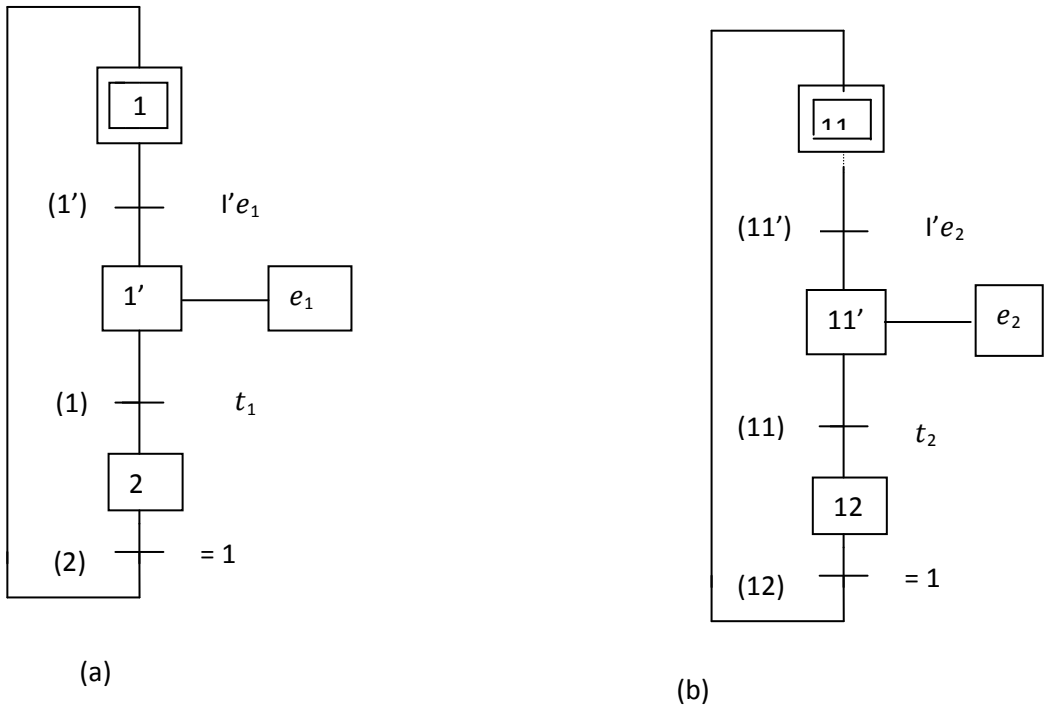


Figure III.8. Deux grafcet équivalents à ceux de la figure III.4.

Dans cette figure III.8, on peut supprimer maintenant les étapes 11 et 22 ainsi que les transitions (11) et (22) parce qu'elles n'ont aucune influence sur le fonctionnement du grafcet. Le grafcet de la figure.8 peut être transformé conformément à la figure III.9.

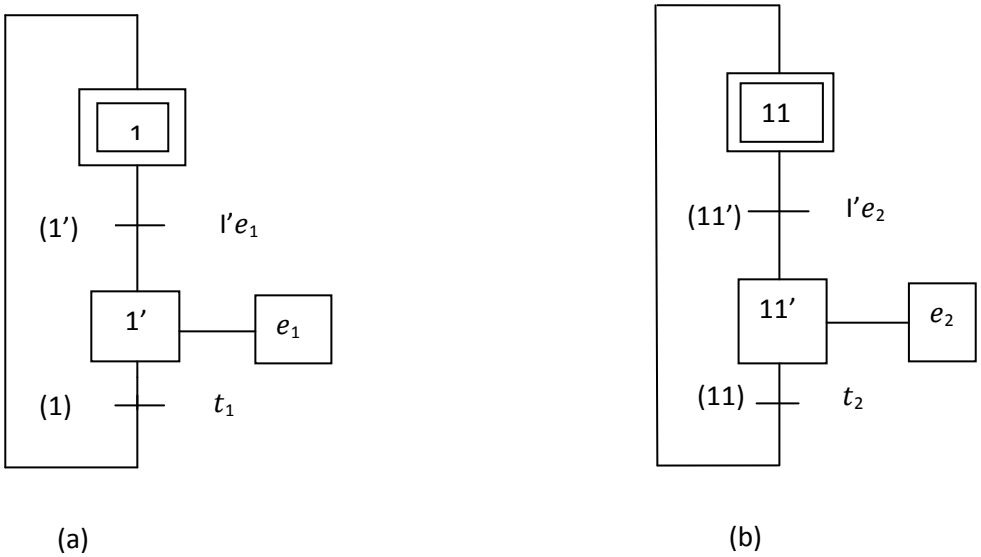


Figure III.9. Deux Grafcet équivalents à ceux de la figure III.8

III.4.3.2. Synthèse du superviseur

a) Modèle automate du procédé étendu sous supervision

Soit G_1de la figure III.10 le modèle automate du procédé étendu sous supervision pour S_1obtenu à partir du Grafcet de la figure III.9.(a).

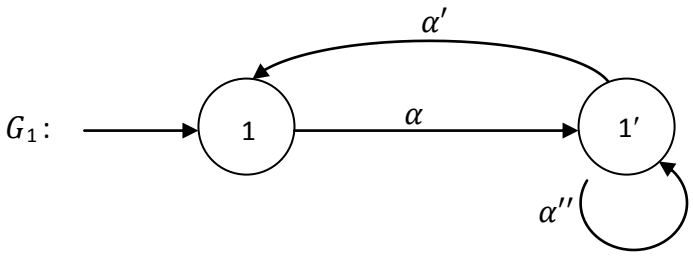


Figure.III.10. Automate du procédé étendu sous supervision pour S_1.

$\alpha = \{e_1\}.\ I'e_1 \qquad \alpha' = \{t_1\}.\ Ie_1 \qquad \alpha'' = \{e_1, t_1\}.\ I'e_1$

A l'état initial, l'étape 1 du grafcet de la figure III.9 (a) est active et le serveur S_1 n'envoie plus de message, quand $I'e_1$ =1 (Ie_1=0) l'évènement e_1 est autorisé alors, l'étape $1'$ est immédiatement activée et l'action impulsionnelle e_1 est exécutée, l'étape $1'$ est activée signifie que S_1 est en travail.la figure III.10 modélise ce changement d'état par la transition $\{1\}\rightarrow\{1'\}$ étiquetée par l'évènement $\alpha=\{t_1\}.I'e_1$. L'évènement α représente l'occurrence de e_1 lorsque $I'e_1$=1. Quand l'évènement t_1 se produit, a cet instant, si $I'e_1$=0 (Ie_1=1) alors l'événement de t_1 ramène le grafcet dans son état initial. l'évènement $\alpha' = \{t_1\}.Ie_1$ associée à la transition $\{1'\} \rightarrow \{1\}$ dans la figure III.10 modélise l'occurrence de t_1 lorsque Ie_1=1. Dans le cas contraire, si Ie_1=0, l'étape $1'$ est alors activée de nouveau et l'action e_1 est exécutée. L'événement $\alpha'' = \{t_1, e_1\}.I'e_1$ associée à la boucle de l'état $\{1'\}$ modélise l'occurrence simultanée de t_1 et e_1 lorsque $I'e_1$=1. Nous obtenons de façon similaire le modèle automate du serveur S_2 de

La figure III.10 qui est construit sur l'ensemble des évènements $\{\beta, \beta', \beta''\}$.

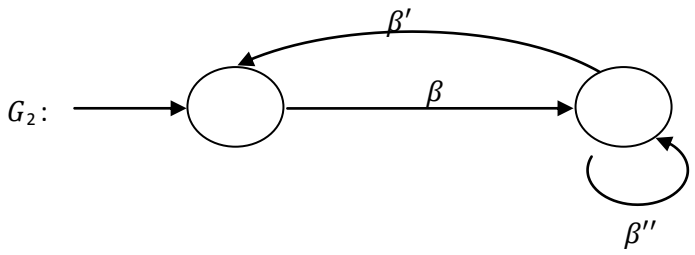


Figure.III.11. Automate du procédé sous supervision pour S_2

$\beta = \{e_2\}.\ I'e_2 \qquad \beta' = \{t_1\}.\ Ie_2 \qquad \beta'' = \{e_2, t_2\}.\ I'e_2$

Remarque

Le produit d'une grandeur booléenne par un évènement est un évènement.

En effectuant le produit asynchrone des deux automates G_1 et G_2 on obtient le modèle global du procédé étendu sous supervision.la figure III.12 représente ce modèle.

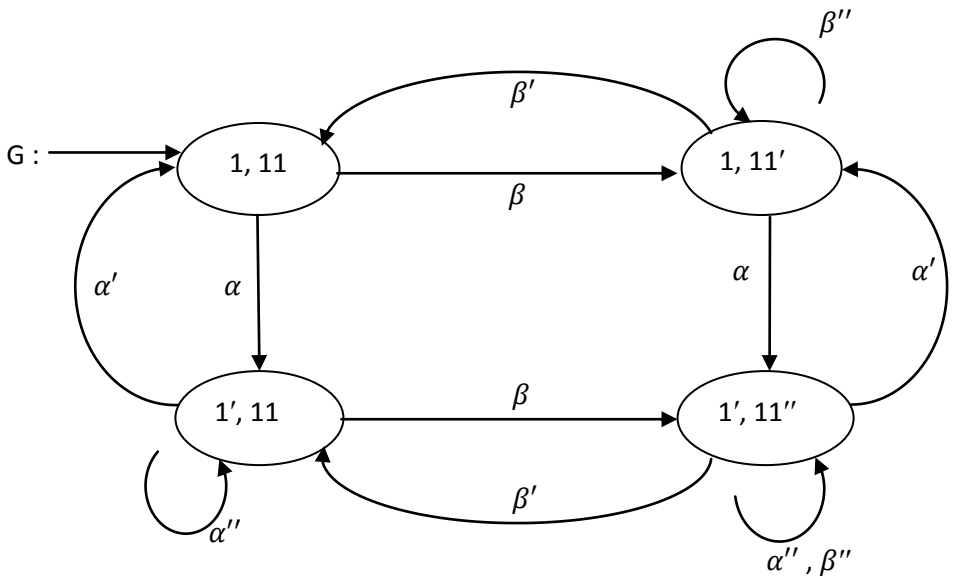


Figure III.12. Modèle accepteur global G du procédé étendu sous supervision.

On notera l'alphabet $\sum'= \{ \alpha, \alpha', \alpha'', \beta, \beta', \beta''\}$

Notons que chaque évènement de $\sum'$se produit simultanément avec l'évènement de $\sum$ éventuellement conditionné par les grandeurs booléennes Ie_1 et Ie_2 ou leur complémentaires, nous avons la propriété suivante pour les évènements de $\sum'$: tout évènement de $\sum'$ qui est conditionné est contrôlable.

A partir de l'état (1′, 11), les évènements 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 au même temps). Si par exemple α'' est contrôlable, on ne peut alors commander α'.

a) Modèle automate des spécifications de superviseur

Pour appliquer le principe de contrôlabilité défini dans la théorie de Ramadge et Wonham sur le procédé étendu sous supervision on a besoin des modèles accepteurs du procédé étendu sous supervision lui-même et celui de la spécification de la supervision.

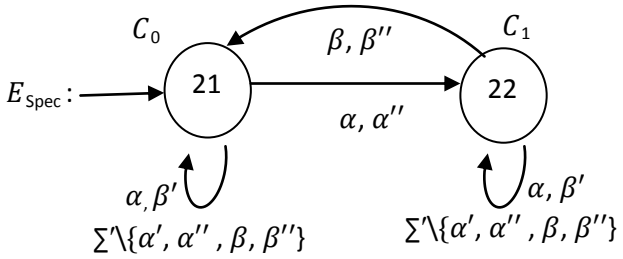


Figure III.13. Modèle des spécifications de supervision construit sur l'alphabet $\sum'$.

A partir de grafcet des spécifications de supervision on peut obtenir un modèle automate. Ceci est illustré dans la figure.13, pour se faire nous allons remplacer les étapes {21} et {22} du grafcet par les états 21 et 22 de l'automate. L'évènement t_1 qui conduit le grafcet de l'étape {21} à l'étape {22} est modélisé par la transition qui mène de l'état 21 à l'état 22. Cependant si un évènement autre que t_1 se produit dans l'étape {21}, le grafcet demeure dans la même étape. Les évènements e_1, t_2 sont alors associée à la boucle de l'état 21. De façon similaire, la transition 22 →21 est étiqueté par e_2 et les évènements e_1 et t_2 sont associée à la boucle de l'état 22 .la sortie Ie_2 associée à l'étape 21 du grafcet signifie que l'évènement e_2 n'est pas toléré dans l'état 21, de même pour l'étape 22 on constate que l'évènement t_1 n'est pas toléré dans l'état 22.

On peut remplacer la transition 21 →22 étiquetée par l'évènement $t_1$dans $\sum$ par la transition étiquetée par l'événement α', α'' de l'alphabet $\sum'$ qui contiennent l'évènement t_1. Pour tout autre évènement que t_1 on obtient le modèle automate des spécifications de supervision représenté dans la figure III.13.

De façon similaire, le passage de l'état 21 à l'état 22 est assuré par les évènements α', α'',dans un premier temps, on associe à la boucle de l'état 21 la liste $\sum'/\{\alpha', \alpha''\}$, par la suite l'événement contrôlable e_2 est interdit dans l'état 21 par la condition Ie_2=1. Ainsi tous les évènements de $\sum'$ conditionnés par I'e_2 (β, β'') seront supprimés de la liste associée à la boucle de l'état 21dans la figure III.12, et nous obtenons finalement la liste

$\sum'/\{\alpha', \alpha'', \beta, \beta''\}$. On obtient de façon similaire la liste des évènements associés à la boucle de l'état 22.

A partir des modèles accepteurs du procédé sous supervision, ainsi que le modèle de supervision, on construit le composé synchrone de G et E_{spec} de la figure III.14.

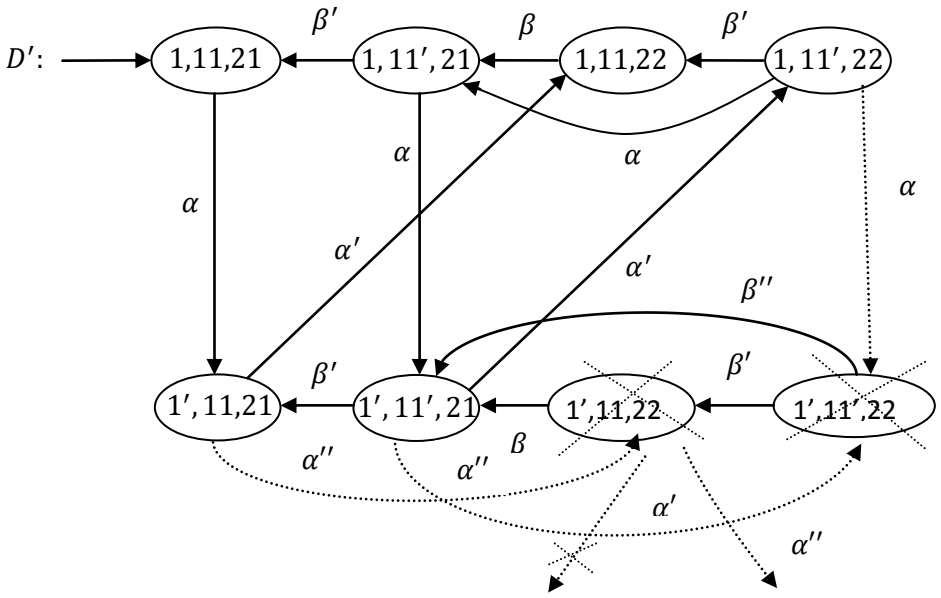


Figure III.14. Modèle accepteur du fonctionnement en boucle fermée du système manufacturier.

III.5. Cas ou les spécifications ne sont pas contrôlables

III.5.1. la contrôlabilité

Considérons la séquence d'évènements α,α', α dans l'exemple de notre système de communication (S_1 envoie un message, le message est transféré dans le canal, puis S_1 envoie un autre message). Après cette séquence, le procédé G est dans l'état (1', 11), à cet état le serveur S_1 envoie un message, 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, alors la séquence $\alpha\alpha'\alpha\alpha'$ est possible dans le procédé, c'est-à-dire $\alpha\ \alpha'\alpha\alpha' \in L(G)$. La séquence d'événements $\alpha\alpha'\alpha$, conduit la spécification dans l'état 22 (présence d'un message dans le canal), 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 αα'α (l'envoi d'un message dans le canal plein n'est pas toléré).cela signifie que

αα'α α'∉ $L(E_{spec})$, ce qui implique que le langage de spécification est non contrôlable par rapport au langage du procédé.

III.5.2. synthèse d'un modèle automate contrôlable à partir d'algorithme de Kumar

On va appliquer l'algorithme de Kumar que nous avons défini dans le premier chapitre en supprimant les états défendus ainsi que les transitions associées à ces arcs, les états défendu de notre système sont (1′, 11, 22) et (1′, 11′, 22)

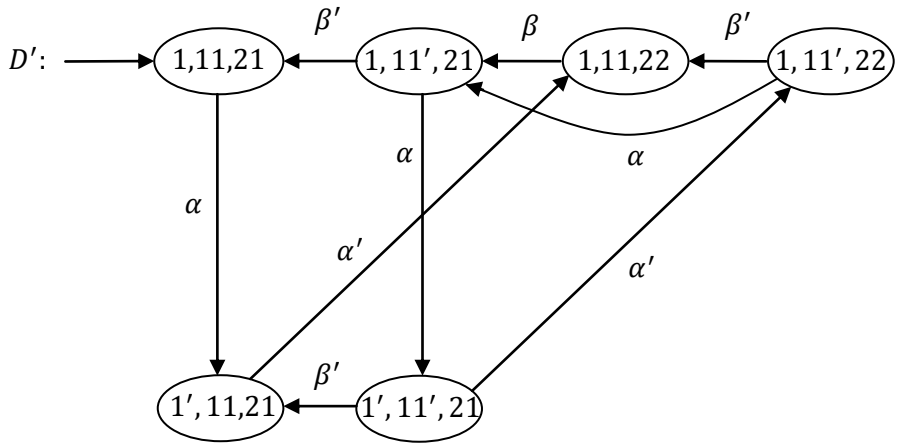


Figure III.15. Modèle accepteur D', (automate de supervision).

III.5.3. Passage des automates au grafcet

Obtention d'un modèle grafcet superviseur à partir de l'automate superviseur de manière systématique qui sera de plus structurelle.

III.5.3.1. Structures élémentaires

Transition autorisée

Considérons la transition entre l'état i et l'état i+1 de la figure III.16. Elle a eu lieu sur l'occurrence de l'événement σ_k s'il est autorisé ($I'\sigma_k = 1$). Ceci se traduit par le Grafcet superviseur de la figure celui- ci autorise l'évènement et attend de voir son occurrence réalisée par le système de commande pour avancer (passage à l'étape i+1)[1].

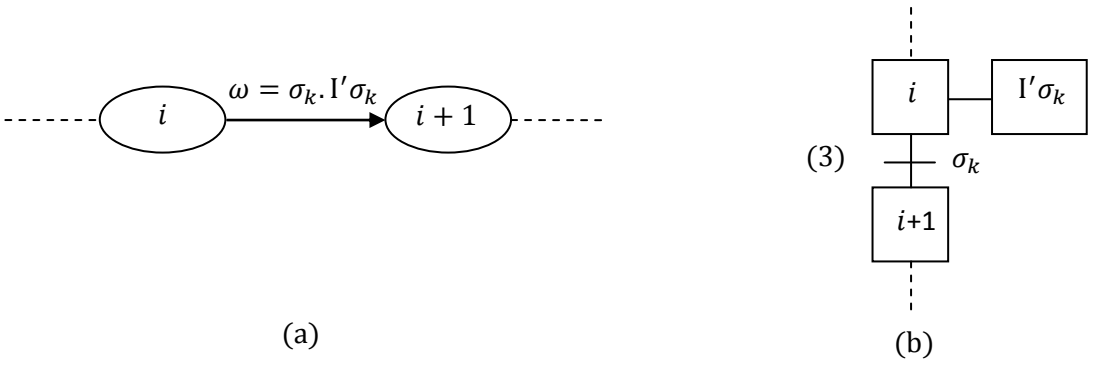


Figure III.16. Passage de l'automate superviseur vers le Grafcet du superviseur pour un évènement autorisé du type $\omega = \sigma_k . \mathrm{I}' \sigma_k$

Dans le cas ou la commutation se fait sur l'occurrence d'évènement successif, tel que α à partir de l'état i à l'état i+1, on ajoute une transition et une étape supplémentaire. Ceci se traduit par le grafcet de superviseur de la figure, le principe de commutation est similaire : le grafcet du superviseur observe σkl, produit I′σk et observe ensuite σk[1].

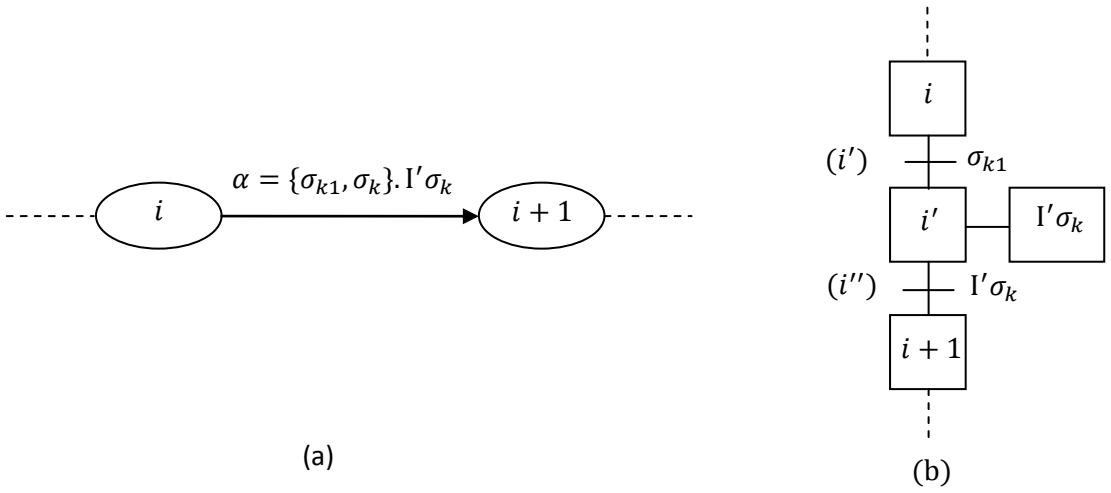


Figure III.17. Passage de l'automate superviseur vers le grafcet du superviseur pour un évènement autorisé du type α = { σkl, σk}. I′σk.

Transition interdite

Considérons la transition ω qui sort de l'état i de la figure. On doit interdire l'occurrence de l'évènement contrôlable σk, pour ne pas tomber dans un étatsinterdit, ceci se traduit par le grafcet de superviseur de la figure III.18[1].

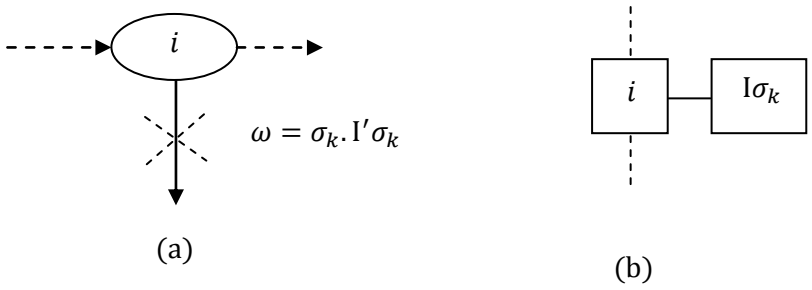


Figure III.18. Passage de l'automate superviseur vers le Grafcet du superviseur pour un événement interdit du type ω= σk.I′σk

Transition interdite et transition autorisée

Ceci revient à faire la somme des deux structures : on autorise ω, i,e. I'σk=1 et on interdit ω', i.e, I σkl=1. Conformément à la figure III.19[1].

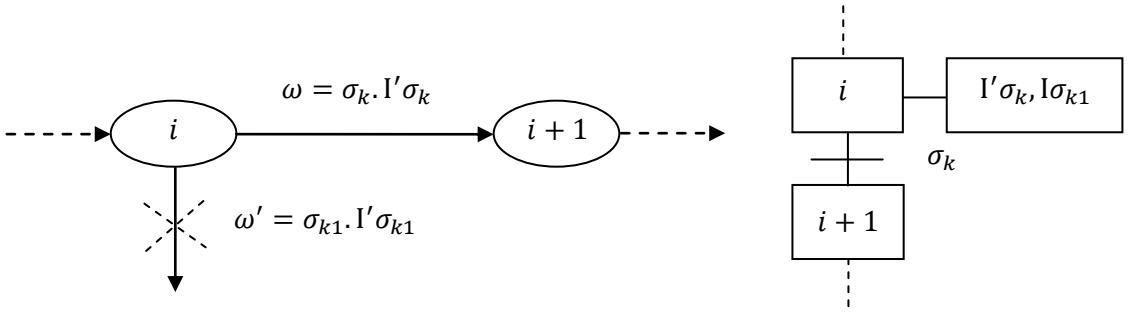


Figure III.19. Passage de l'automate vers le Grafcet du superviseur pour un événement autorisé et un événement interdit.

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 événement de $\sum'$ figure III.20[1].

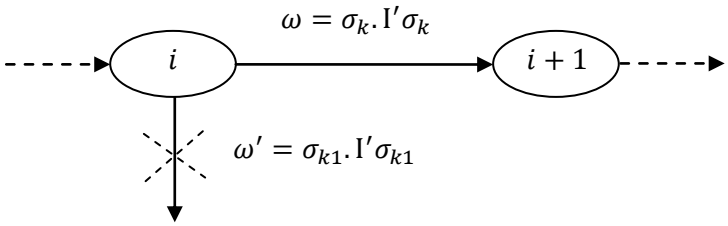


Figure III.20

i. Etape 1

À 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[1].

ii. Etape 2

pour réaliser la commutation sur l'occurrence de ω' qui conduit à un état interdit, on ajoute à l'étape i du grafcet précédent, l'événement I σkl qui garantit de ne pas tomber dans un état interdit[1].

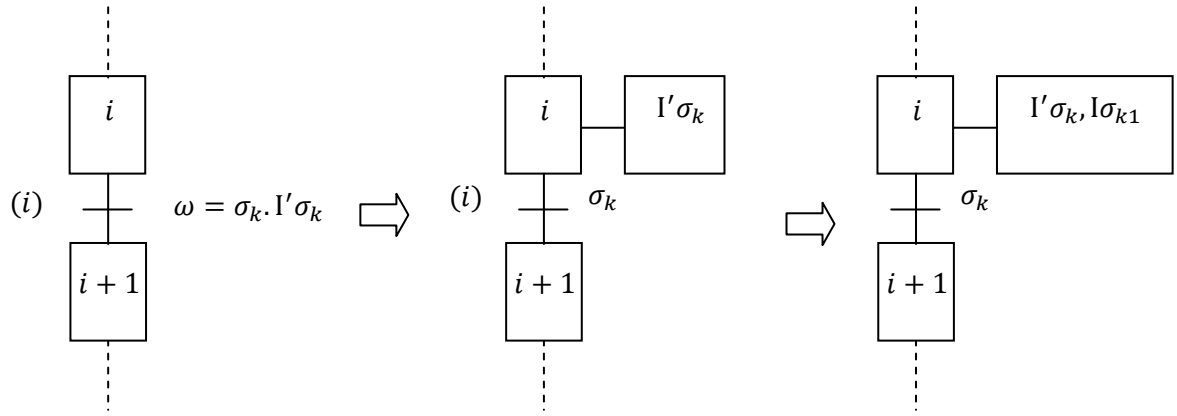


Figure III.20. Les étapes de passage de l'automate superviseur vers le Grafcet superviseur dans le cas général.

III.5.3.2 grafcet de supervision final

On applique la première étape, on obtient le grafcet présenté dans la figure III.21.

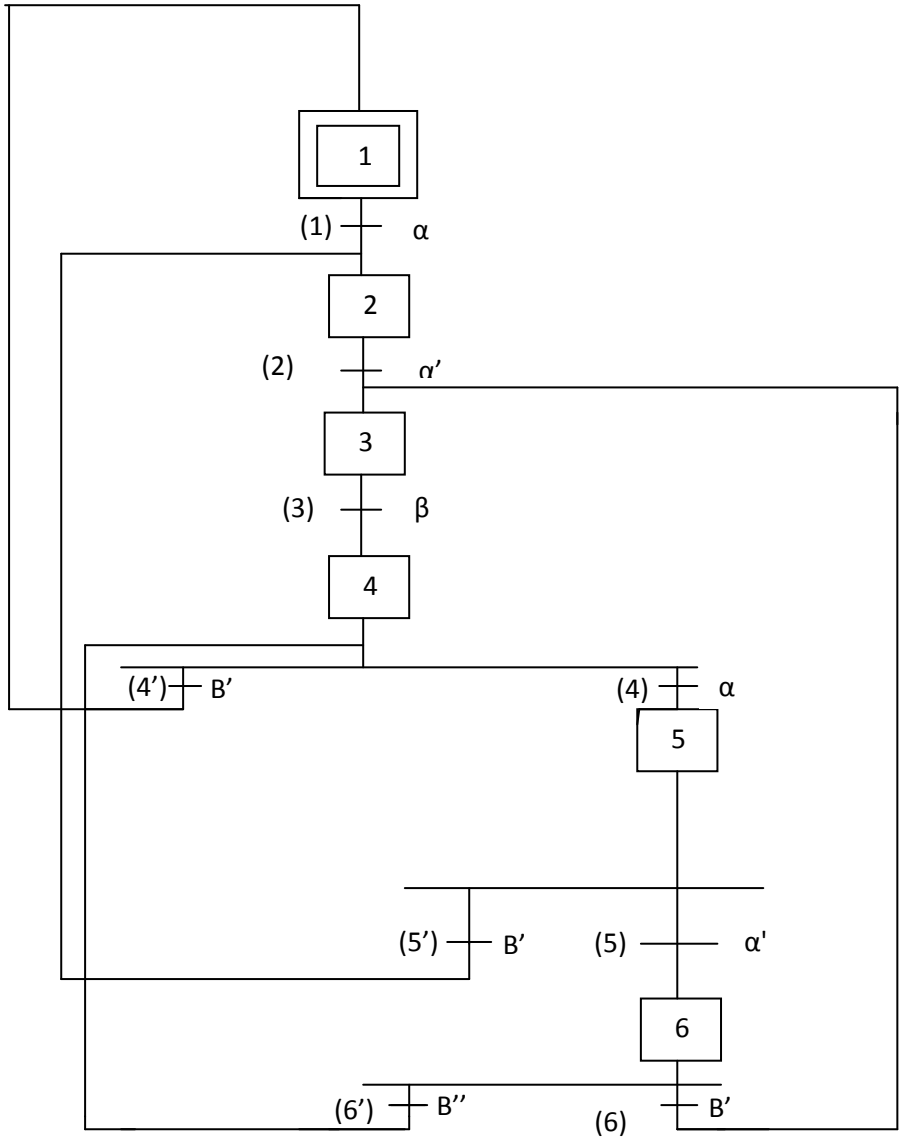


Figure III.21. Création des étapes du grafcet correspondant aux états de l'automate D′.

Nous remarquons qu'à partir de l'étape 6 dans la figure III.21, nous avant deux transirons 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 nous obtenons une évolution non déterministe figure III.22.

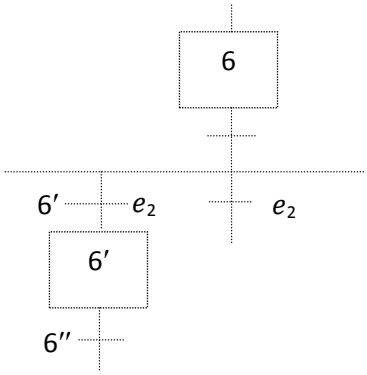


Figure III.22[1]

Donc il faut faire un choix. Si on choisit de contrôler β′ on supprime la transition β″ et on obtient la figure III.23. Si on choisit de contrôler β″ on supprime la transition β′ et on obtient.

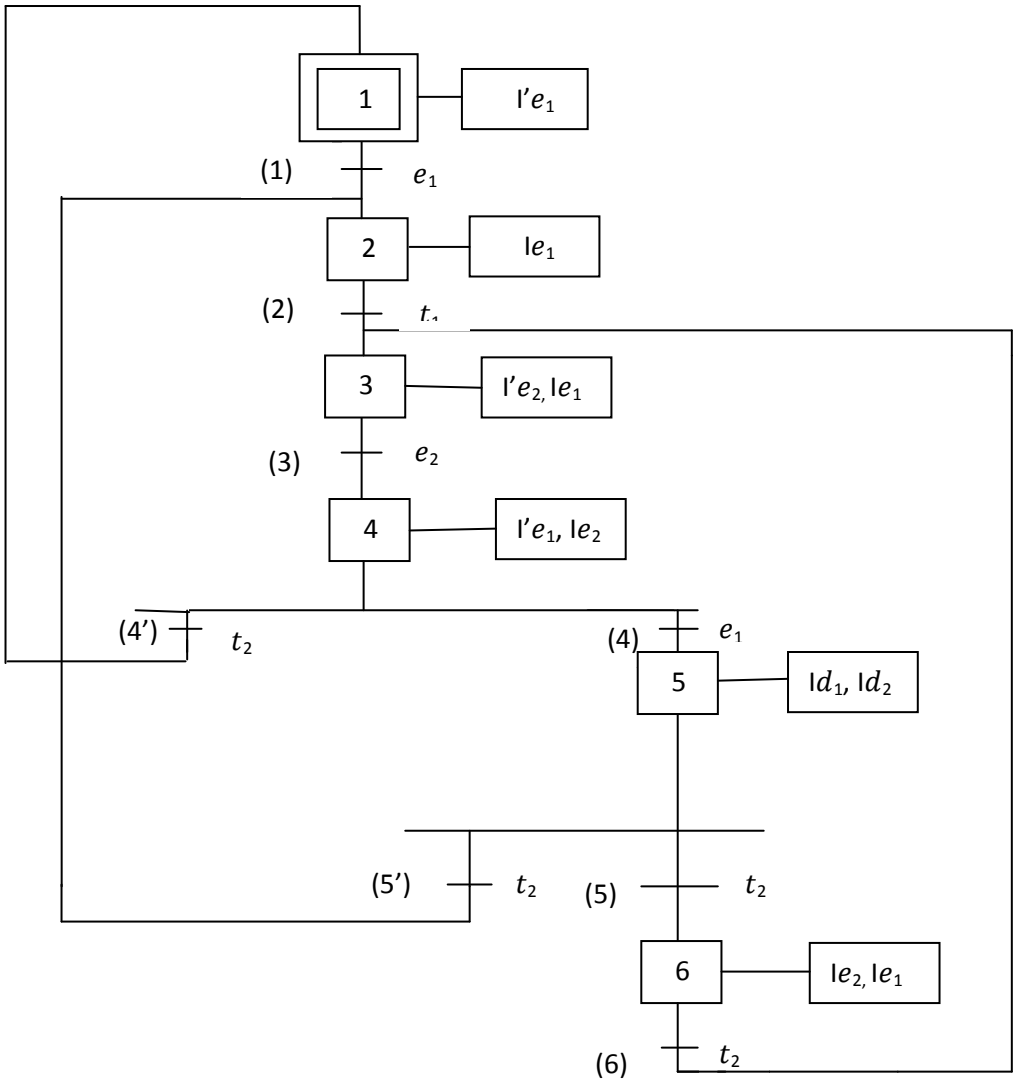


Figure III.23. Grafcet du superviseur.

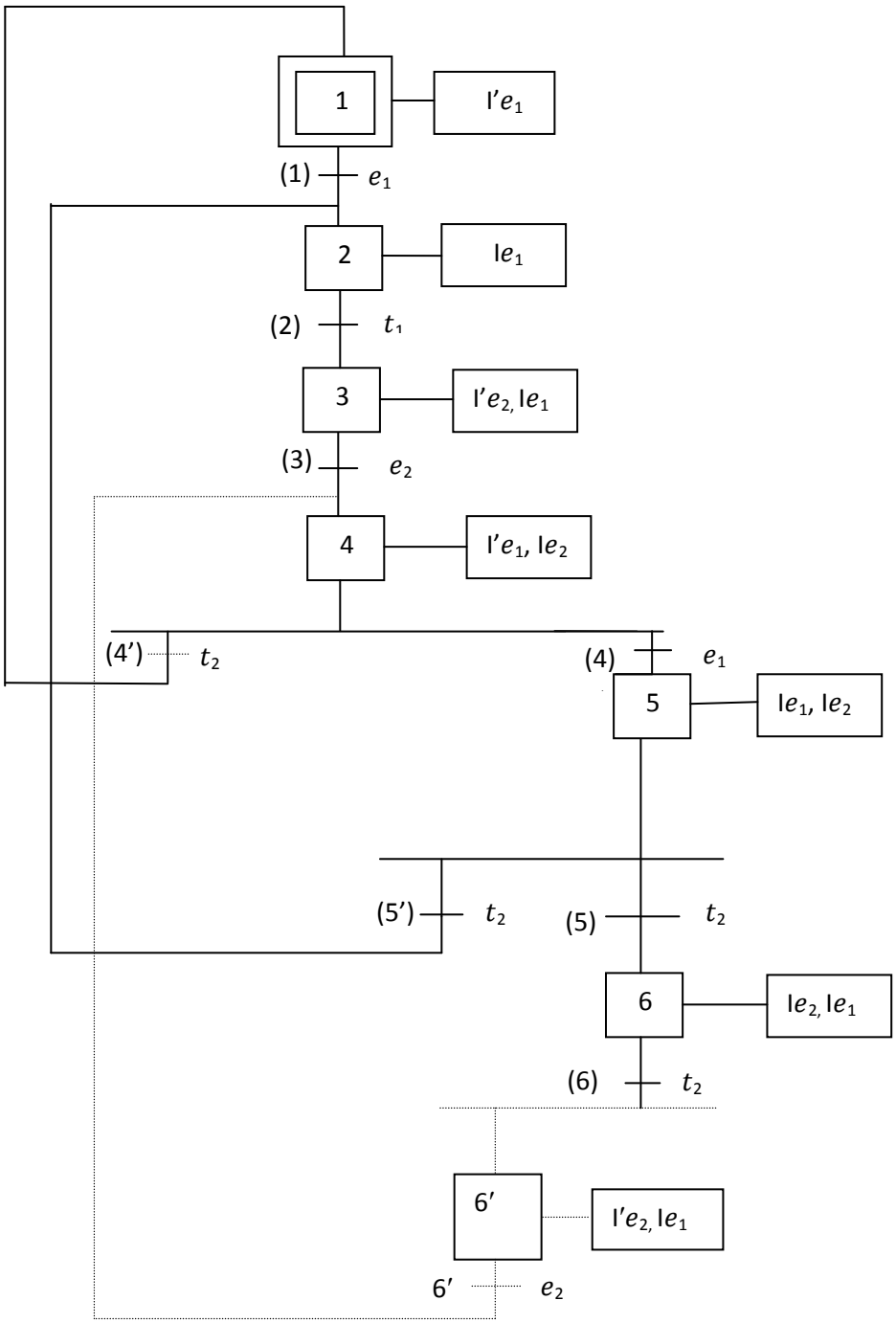


Figure III.24

Les grafcet de la figure III.23 et la figure III.24 sont équivalents et ils ont le même comportement ou 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 III.25 et de la figure III.26. Nous remarquons que dans

l'automate de la figure III.26 l'état 3 et 6′ vont interdire les mêmes évènements, donc ils représentent un seul état qui est l'état 3 dans la figure III.25.

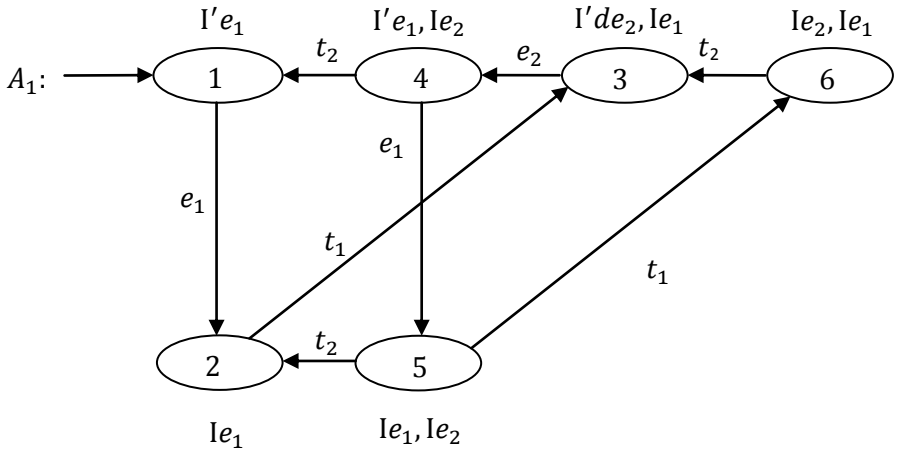


Figure III.25 : Automate correspondant au Grafcet de la figure III.23.

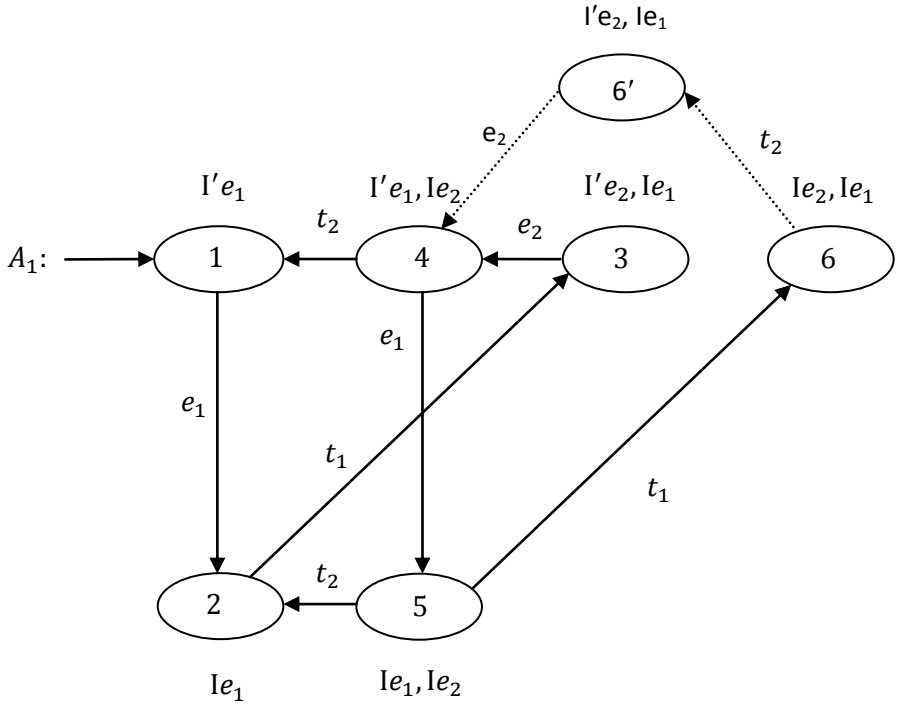


Figure III.26 : Automate correspondant au Grafcet de la figure III.24.

Donc les événements β″ et β′ sont mutuellement exclusifs, et ils ne se produisent pas ensemble.

III.6.Conclusion

L'approche de la commande supervisée en modélisant le système de communication par Grafcet nous a permis d'implanter directement le grafcet de commande ainsi que le grafcet de superviseur dans un automate programmable.

Dans cette approche nous avons aussi étudié la contrôlabilité du langage des spécifications de supervision qui sont non contrôlables en passant d'un modèle Grafcet à un modèle automate pour reconstruire finalement le modèle Grafcet superviseur optimal. Le grafcet final de superviseur obtenu par cette approche contient un nombre important d'étapes qui est identique à l'automate synthétisé. Cette complexité rend l'application du grafcet difficile ou même impossible dans le ca d'application réelle.

IV.1 Introduction

Dans le chapitre précédent nous avons présenté une méthode de synthèse de superviseur calculé directement à partir d'un automate contrôlable. Nous avons vu comment passer de l'automate superviseur vers un grafcet. Le modèle grafcet obtenu est aussi compliqué que celui du modèle automate ce qui nous ramène au problème de la commande posé dans la théorie de Ramadge et Wonham. L'inconvénient majeur de cette synthèse est l'explosion combinatoire du nombre des états quand il s'agit d'étudier des systèmes de taille réelle. Les Rdp par leur puissance de représentation graphique permettent de pallier à cet inconvénient.

Dans ce chapitre nous présentons une méthode de synthèse de contrôleur basée sur les Rdp en appliquant l'approche des invariant de marquages. De plus, nous sommes intéressés par des outils de modélisation facilement implantables dans un automate programmable industriel (API). Ainsi la méthode L basée sur les Rdp est adaptée ensuite au Grafcet.

Dans la suite nous allons utiliser le terme « contrôleur » aux places de supervision car le superviseur défini dans les chapitres précédents constitue une partie séparée du système de commande et dans la suite le superviseur sera confondu avec la commande et sera appelé contrôleur.

IV.2 Modélisation par Rdp et application de la méthode des invariants de marquages

Notre objectif est de synthétiser un contrôleur modélisé par un Rdp en appliquant les invariants de marquages et finalement passer à un modèle grafcet. Les étapes suivies dans cette synthèse sont illustrées par la figure suivante :

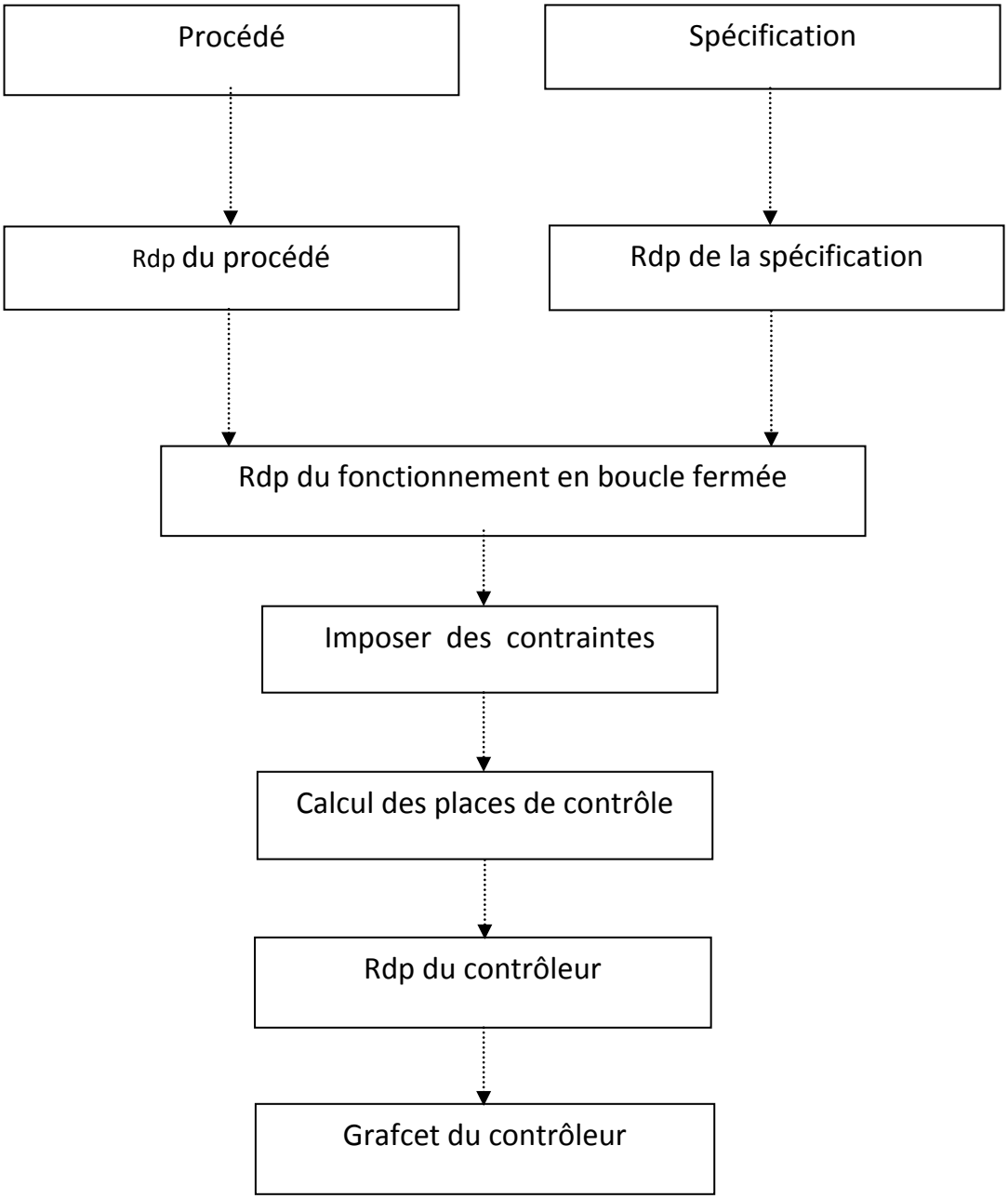


Figure VI.1. Les étapes de la synthèse du contrôleur proposée.

Le point de départ consiste à modéliser le procédé, la spécification par un Rdp. Le comportement désiré en boucle fermée est obtenu par la synchronisation des deux modèles. Imposer des contraintes par les spécifications pour calculer les places de contrôle en appliquant la méthode des invariants de marquages. Introduire les places de contrôle dans le Rdp du fonctionnement en boucle fermée, et finalement passer d'un modèle Rdp à un modèle grafcet.

IV.2.1 Modélisation par Rdp

Définition

On appelle procédé étendu le procédé couplé à son système de commande. Le modèle Rdp du procédé étendu sera noté $R_{G.}$

Exemple d'application

Pour illustrer les idées que nous développons dans ce chapitre, nous allons considérer l'exemple d'un système industriel constitué de convoyeurs et de postes robotisés. Notre objectif est de déterminer un contrôleur pour la boucle extérieur du système, composée d'un convoyeur et d'une station d'assemblage, sachant que les palettes viennent de la boucle centrale, cheminent sur le convoyeur jusqu'à la station d'assemblage. Quand l'assemblage est terminé la palette revient sur le convoyeur, puis elle est remise sur la boucle centrale pour subir d'autres opérations figure VI.2 :

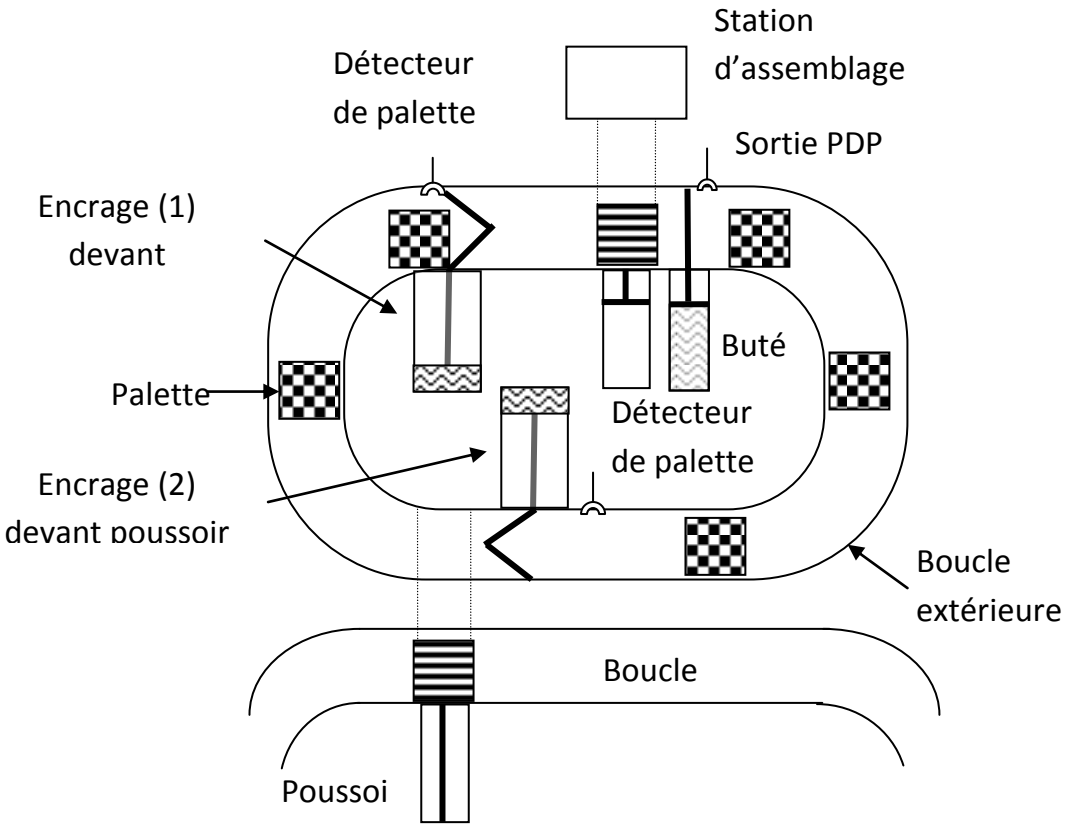


Figure IV.2. Système d'assemblage.

Les éléments associés à chaque transition sont explicités dans le tableau ci-après :

Evénement	Action	Evénement	Action	Evénement	Action
DPA1	Détection palette devant Ancrage 1	OA1	Ouverture ancrage 1	APA1	Absence palette devant Ancrage1
FA1	Fermeture ancrage 1	DPB	Détection palette devant butée	SRT	Début lecture étiquette
FEE	Fin écriture étiquette	OB	Ouverture butée	SPDP	Sortie palette du PDP
FB	Fermeture butée	OA2	Ouverture Ancrage 2		

Tableau IV.1. Événements associés aux transitions.

Les événements OA1, OA2 et OB sont des événements contrôlables.

- Les Spécifications

Nous avons choisis les spécifications suivantes :

1) Le nombre de palettes entre l'ancrage devant station et la butée est limité à 1.
2) Le nombre de palettes entre la sortie du poste de présentation (SPDP) et l'ancrage du poussoir est limité à 2.

IV.2.1.1. Modélisation

Nous avons modélisé le système par un Rdp, sachant que l'ensemble des places $\{P_0, P_1, P_2, P_3\}$ représente les situations de la palette près de l'ancrage et l'ensemble des places $\{P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}\}$ celle de la butée. La place P_{15}

représente la situation de la palette entre l'ancrage et la butée. Chaque place est représentée dans le tableau suivant :

Nom de la place	description	Nom de la place	Description
P_0	Fermeture de l'ancrage 1 L'absence de palette	P_8	Deux palettes entre la butée et l'actuateur
P_1	Présence de palette devant l'ancrage	P_9	La Butée est fermée
P_2	Ouverture de l'ancrage	P_{10}	Présence de palette devant la butée
P_3	Absence de palette lorsque l'ancrage est ouvert	P_{11}	Opération de lecture, assemblage et écriture
P_4	Absence de palette entre l'ancrage et la butée	P_{12}	Fin d'assemblage
P_5	Une palette ente l'ancrage et la butée	P_{13}	La butée est ouverte
P_6	Zéro palette entre la butée et l'actuateur	P_{14}	Absence de palette après la butée
P_7	Une palette entre la butée et l'actuateur	P_{15}	Présence de palette entre l'ancrage et la butée

Tableau IV.2. Description de chaque place.

Le Rdp du procédé étendu du système industriel est donné par la figure suivante :

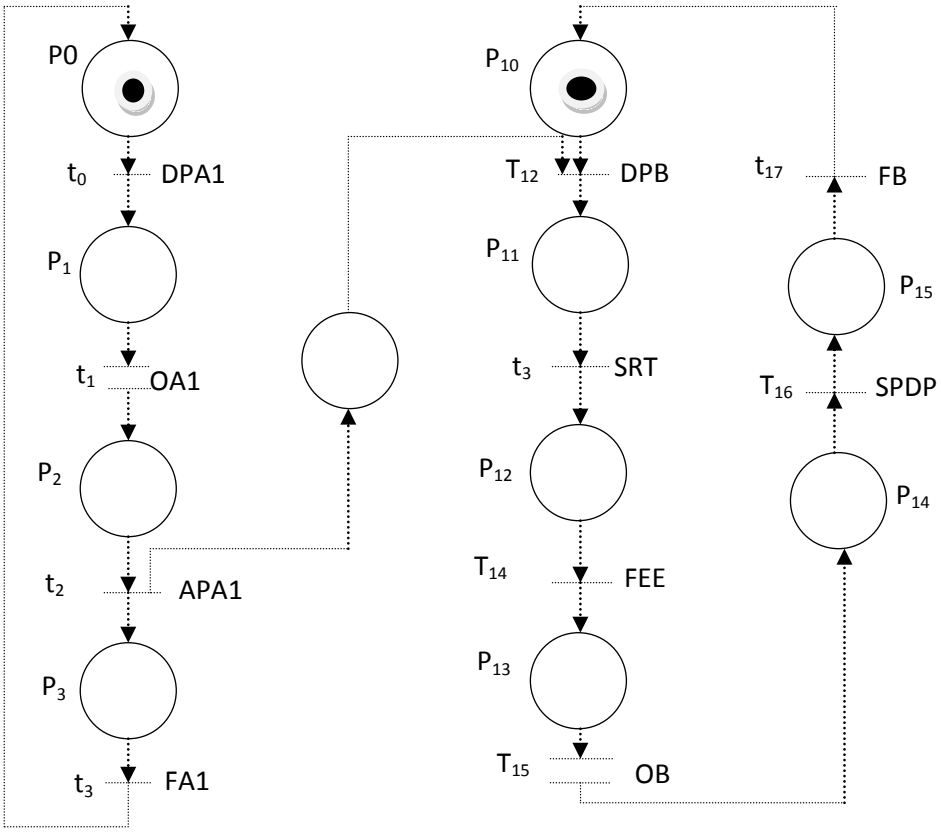


Figure IV.3. Rdp du système industriel

IV.2.1.2. Rdp de la spécification

Les spécifications sur l'état sont souvent données comme ensemble de situations interdites ou autorisées par le procédé à contrôler. Notre procédé étendu est soumis à des spécifications que nous modéliserons par un Rdp.

Définition

On appelle modèle Rdp de spécification le modèle Rdp qui définit les spécifications notées R_E

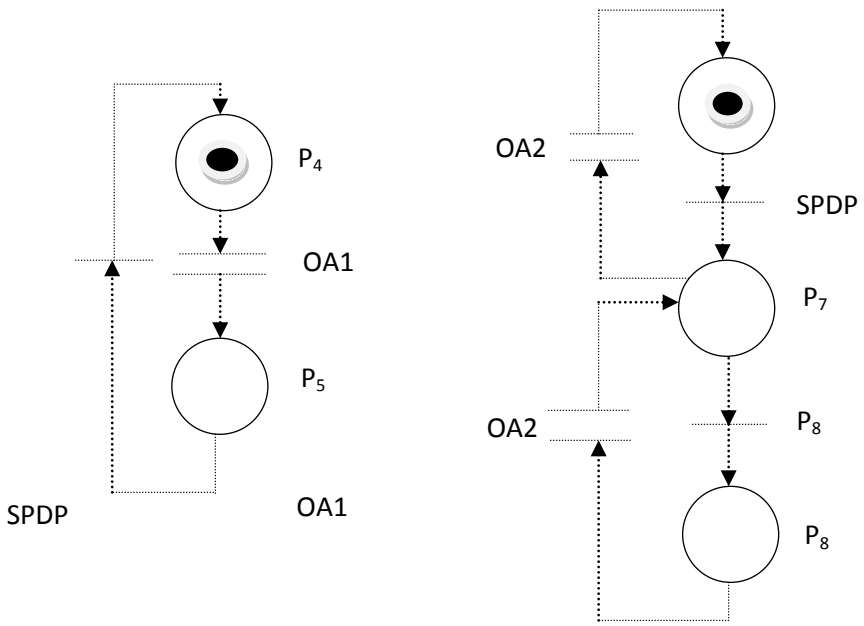


Figure IV.4. Modèle des spécifications.

IV.2.1.3 Réseau de Petri du fonctionnement en boucle fermée

a) Synchronisation des deux Rdp

Définition

Le fonctionnement en boucle fermée correspond au comportement du procédé sous l'influence du superviseur. Il est obtenu par la synchronisation des Rdps (R_G et R_E) figure IV.5.

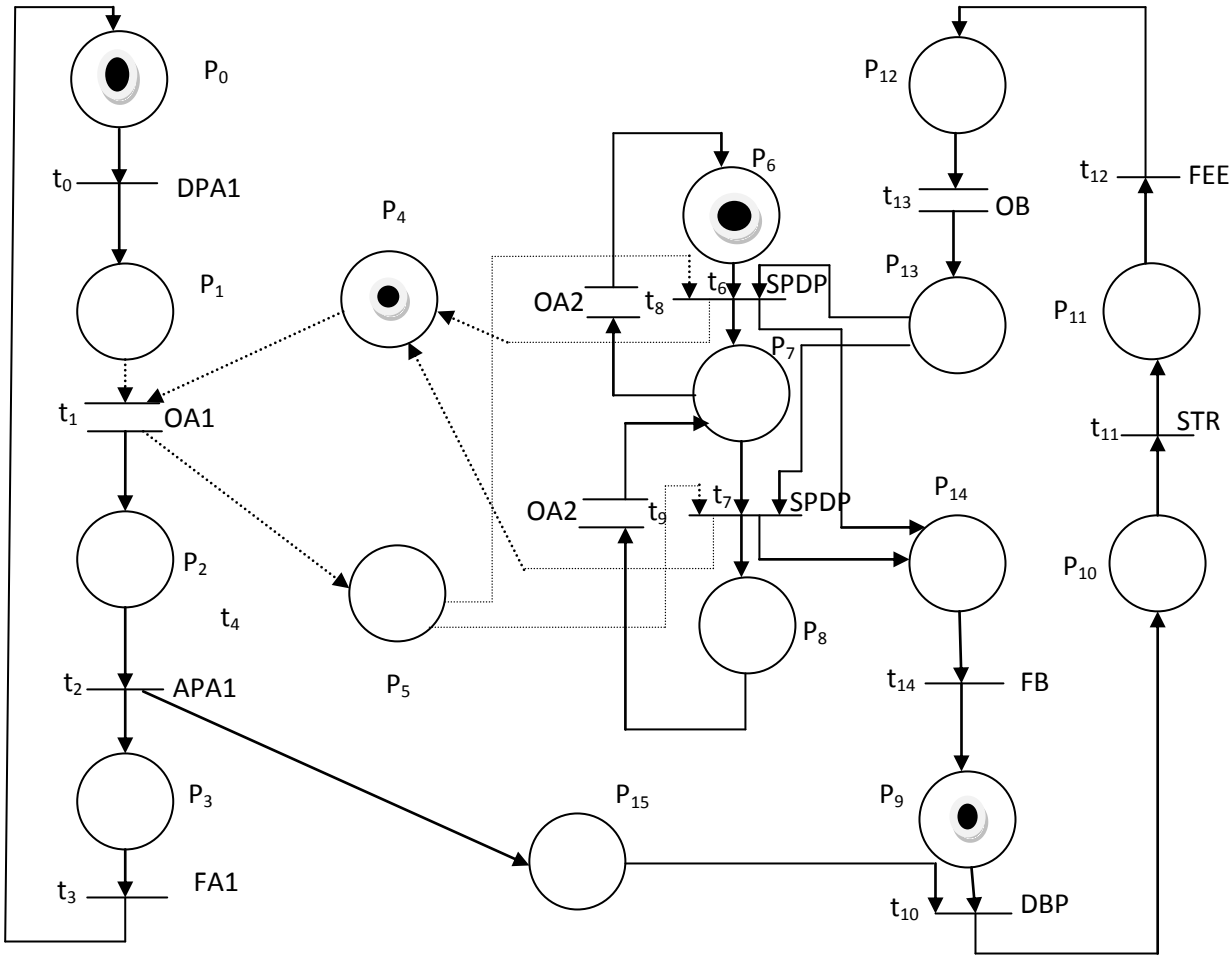


Figure IV.5. Modèle Rdp de système manufacturier en boucle fermée.

b) Détermination des contraintes

Revenons au fonctionnement physique du système et à la spécification.

La première spécification limite le nombre de palettes entre l'ancrage devant station et la butée à 1, ce qui veut dire qu'on doit interdire le passage de palette par l'ancrage 1 donc interdire l'ouverture de l'ancrage 1 (associé à l'événement OA1) s'il ya déjà une1 palettes. Pouvoir contrôler OA1 (événement contrôlable) nous permet de garantir la première spécification en interdisant l'occurrence de l'événement OA1 de se produire.

La deuxième spécification limite le nombre de palette entre la sortie du poste de présentation (SPDP) et l'ancrage du poussoir à deux palettes ce qui veut dire qu'on doit interdire le passage de palettes par le SPDP après le passage de la 2^{eme} palette.

La présence de 2 palettes entre le SPDP et l'ancrage du poussoir marque la place P_8 (P_8=1). D'un autre coté, si la place P_5 est marquée (présence de palette entre l'ancrage 1 et la butée) et que la butée est ouverte P_{13}=1, il n'y a aucun moyen d'interdire le passage d'une palette par le SPDP (capteur) qui est un événement incontrôlable, alors on ne peut pas garantir la 2^{eme} spécification qui veut qu'il y ait 2 palettes entre le SPDP et l'ancrage du poussoir.

Il faut donc ajouter une autre contrainte qui permet de garantir le respect de cette spécification. Il s'agit de garantir que :

$$\mu(P_5) + \mu(P_8) + \mu(P_{13}) \leq 2 \Rightarrow \mu(P_5) + \mu(P_8) \leq 1 \text{ ou } \mu(P_{13}) + \mu(P_8) \leq 1$$

1) La contrainte $\mu(P_5) + \mu(P_8) \leq 1$ implique qu'on interdit la présence d'une palette entre l'ancrage 1 et la butée quand la place P_8=1 (présence d'une palette entre le SPDP et l'ancrage du poussoir). Ce qui restreint le fonctionnement du système donc cette spécification ne sera pas retenue (c'est-à-dire qu'on interdit quelque chose qui est autorisée par le cahier des charges, à savoir la présence d'une palette au plus entre l'ancrage 1 et la butée (spécification 1) .

2) La contraint $\mu(P_{13}) + \mu(P_8) \leq 1$ autorise la première spécification mais exige la fermeture de la butée ce qui nous amène à conclure que le respect de cette spécification est garantie par la contrainte $\mu(P_{13}) + \mu(P_8) \leq 1$

IV.2.2 Synthèse d'un superviseur

L'outil de modélisation de notre système nous permet d'appliquer l'approche des invariants de marquages présentée au II^{eme} chapitre.

- *calcul des places de contrôle*

Soit D_G la matrice d'incidence du Rdp correspondant au fonctionnement en boucle fermée, l'approche de synthèse consiste à compléter cette matrice par la matrice D_E correspondant aux places de contrôle. Soit D la matrice d'incidence du système contrôlé, chaque place de contrôle va ajouter une ligne à la matrice d'incidence du procédé D_G pour avoir la matrice du procédé contrôlé D . Cette matrice sera composée de la matrice du procédé et la matrice d'incidence du contrôleur D_C :

$$D = \begin{pmatrix} D \\ G \end{pmatrix}$$

Dans le cas général, l'ensemble des contraintes peut s'écrire sous la forme matricielle :

$L \cdot \mu_G \leq b$ où :

L : Matrice des contraintes,

b : Vecteur dont les éléments sont des entiers.

Le principe de ce contrôle est de créer des invariants en ajoutant des places :

$L \cdot M_G + M_C = b$

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

$D_C = - L \cdot D_G$ et $\mu_{C0} = b - L \cdot \mu_{G0}$

Le contrôleur est connu par la détermination de sa matrice d'incidence D_C et son marquage initial μ_{C0}.

Nous avons vu dans le 2eme chapitre le problème des transitions incontrôlables qui est défini par la présence des arcs partant de la place de contrôle vers une transition incontrôlable et pour résoudre ce problème, on se réfère à la méthode proposée par Yamalidou qui consiste à remonter les branches jusqu'à trouver une transition contrôlable qui soit en aval de la place de contrôle. Initialement la place de contrôle P_C est marquée.

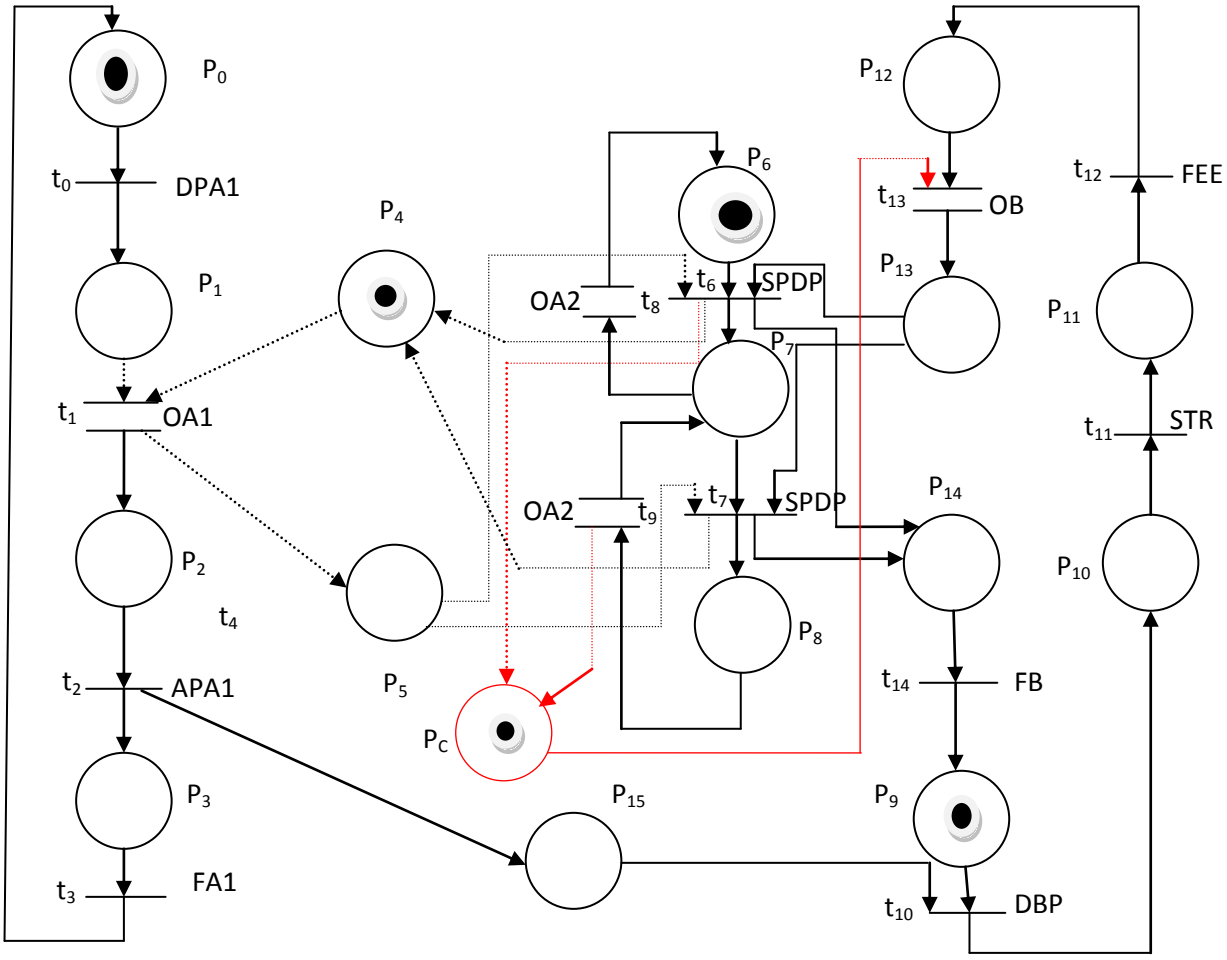


Figure IV.6. Modèle Rdp final du système sous contraintes.

Remarque

Dans notre exemple la place de contrôle obtenue ne contrôle que des transitions contrôlables, donc les spécifications sont satisfaites (on n'a pas besoin de remonter les branches jusqu'à trouver une transition contrôlable). Ce qui nous permet de qualifier le contrôleur d'optimal.

IV.3 Le passage d'un réseau de Petri à un Grafcet

Le modèle Rdp contrôlé obtenu n'est pas directement implantable dans les API, pour pouvoir réaliser la commande, le Rdp doit être traduit en un langage approprié par exemple le Grafcet.

Pour passer d'un Rdp à un Grafcet, David et Alla[12] ont imposé au Rdp dans leur définition de posséder les quarte caractéristiques suivantes :

1) Il est non temporisé
2) Il est sauf
3) Il est déterministe
4) Ses possibilités d'entrées et sorties sont étendues pour être homogène avec celle du Grafcet.

Pour trouver le réseau de Petri sous-jacent à un grafcet, il faut enlever tout ce qui concerne la spécification des réceptivités, celles des actions ainsi que tout ce qui concerne le temps, il reste donc un graphe comprenant des étapes et des transitions que l'on peut voir comme les places et les transitions d'un réseau de Petri sous-jacent, mais il s'agit bien d'un réseau de Petri particulier.

Une étape peut être active ou inactive, ce qui correspond au fait qu'une place d'un réseau de Petri peut être vide ou contenir un jeton.

Mais dans le cas général, une transition d'un réseau de Petri peut contenir plus d'un jeton. La classe particulière de réseau de Petri correspondant au Grafcet est celle des réseaux de Petri saufs (une place ne peut pas recevoir plus d'un jeton).

Le Rdp sauf est proche du Grafcet, cependant il y a des différences importantes entres ces deux outils :

1-L'évolution simultanée : dans le Grafcet, lorsque une étape valide deux ou plusieurs étapes, il y a alors franchissement de toutes les transitions. Dans un Rdp sauf ce n'est pas possible ce qui nous amène à un conflit. La résolution de ce problème consiste à choisir une stratégie pour lever ce conflit (priorité, aléatoire, alternatif…).

2-Réactivation : dans un Grafcet, lorsqu'une étape active est activée à nouveau, elle reste active, mais dans le Rdp le nombre de marque augmente .Dans un Rdp sauf ce problème n'existe pas. Cependant, une place de contrôle synthétisée ait un marquage binaire rend le modèle global non sauf.

3-Notion de conflit : les différences de comportement du Grafcet et des réseaux de Petri en cas de conflit découlent directement de la nature synchrone de l'un et asynchrone de l'autre. Dans un Rdp il y a conflit lorsqu'un marquage sensibilise plus d'une transition, mais le nombre de marques contenues dans la place est insuffisant pour que toutes ces transitions de sortie soient effectivement franchies.

Définition1

Un modèle de système (système à événement discrets) est dit synchrone lorsqu'il comporte une horloge de base et que les événements ne peuvent se produire (ou bien qu'ils ne peuvent être pris en compte) qu'aux instants définis par les impulsions de cette horloge de base. Il peut arriver que plusieurs événements se produisent (ou soient pris en compte) lors de la même impulsion. Ils sont alors vus comme simultanés.

Le Grafcet est un modèle synchrone. On suppose que le temps est échantillonné ce qui est réaliste car les valeurs des capteurs ne sont pas lues par l'automate programmable que lors de certaines impulsions de l'horloge de base. On suppose en général que les valeurs des capteurs sont mises à jour en début de cycle. Plus d'un

capteur peut changer de valeur à un instant donné et donc plus d'une transition peut être validée. Les transitions validées sont simultanément franchies.

Définition2

Un modèle d'un système (système à événements discrets) est asynchrone lorsqu'il ne fait aucune référence à une horloge de base. Le temps est considéré comme une variable continue (dense, c'est-à-dire représentée par un nombre réel, ce qui implique qu'entre deux instants, il est possible de définir une infinité d'instant intermédiaires). Les événements peuvent se produire à n'importe quel instant et comme ils sont de durée nulle, il n'est pas possible que deux événements se produisent au même instant (même s'ils sont infiniment proches).

Le réseau de Petri est un modèle asynchrone. Il n'est fait référence à aucune horloge, à aucune notion d'impulsion. Lors de la définition du comportement dynamique, on suppose toujours qu'une seule transition est franchie à chaque instant. Quand on construit le graphe des marquages accessibles, les arcs sont toujours étiquetés par le nom d'une seule transition. On ne présente pas des changements d'état qui correspondraient au franchissement simultané de plusieurs transitions.

- *Représentation graphique et vocabulaire*

1) place (Rdp)→ étape(Grafcet)

0 ou 1 jeton → inactivation ou activation

Marquage → situation

2) transition

Validée (sensibilisée) dans une situation.

Conditions logiques pour changer de pas

3) arcs

Pas de flèches si orienté du haut vers le bas.

- *Modèle grafcet du modèle Rdp du contrôleur*

Le réseau de Petri du contrôleur obtenu possède toutes les caractéristiques définies par David et Alla à savoir : non temporisé, déterministe, il est sauf et ses entrées et sorties sont étendues pour être homogène avec celle du Grafcet (manipulation des événements).

Absence de conflit même en ajoutant la place de contrôle. . Les opérations A, B, C, D, E correspondent, respectivement, aux opérations dans les places P_0,P_2 ,P_9,P_{11}, P_{13} selon la description de ces places dans le tableau IV.2.

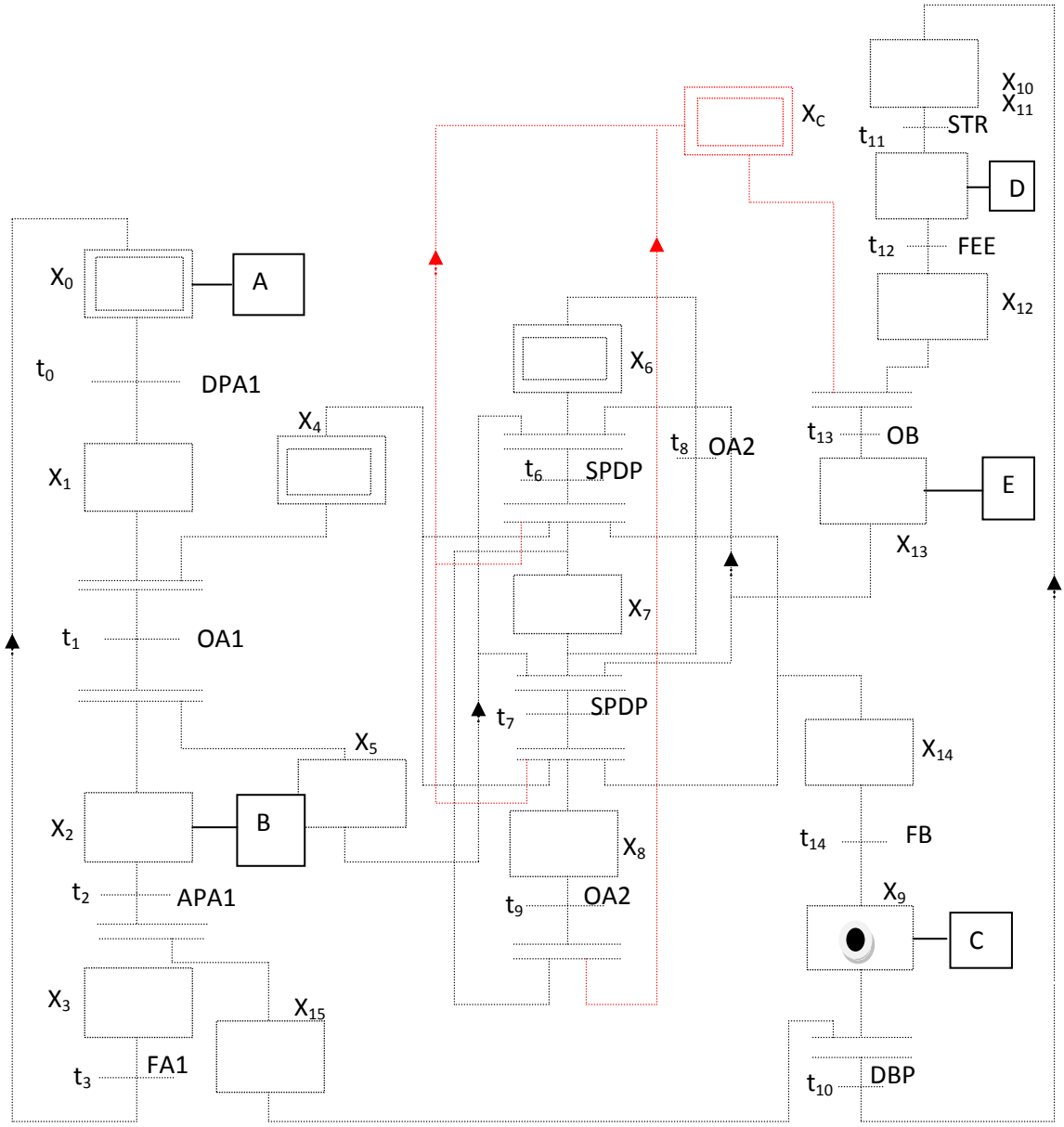


Figure IV.7. Modèle grafcet du contrôleur.

IV.4. Conclusion

Nous avons présenté dans ce chapitre une application de la synthèse de contrôleur pour système manufacturier modélisé par des réseaux de Petri. On a appliqué l'approche des invariants de marquages après avoir bien défini les spécifications. La place de contrôle obtenu ne contient pas de sorties vers des transitions incontrôlables, donc on n'a pas eu à utiliser la technique proposée par Yamalidou qui consiste à remonter les branches jusqu'à atteindre une transition contrôlable, ce qui nous permet de qualifier le contrôleur d'optimal. Nous avons situé les différences entre un Rdp et un Grafcet pour passer finalement d'un Rdp à un Grafcet donc vers une implémentation.

Conclusion générale

La synthèse de superviseur/contrôleurs pour des systèmes à événements discrets a été abondamment étudiée dans la littérature à partir des outils automates et réseaux de Petri. Cette théorie a été initiée par les travaux de Ramadge et Wonham qui ont utilisé les automates comme outil de modélisation. Comme nous l'avons constaté, malgré l'optimalité du superviseur obtenu par cette méthode, l'explosion du nombre d'états pour les systèmes complexes demeure le principal inconvénient de cette approche.

Nous avons présenté, par la suite, l'approche des invariants de marquages basée sur les Rdp qui consiste à calculer des places de contrôle. En dépit de la puissance de représentation graphique des réseaux de Petri, on ne peut pas toujours aboutir à un contrôleur optimal, lorsqu'au moins une place de contrôle conditionne une transition non contrôlable. La technique proposée par Yamalidou consiste a remonté les branches jusqu'à atteindre une transition contrôlable n'est pas générale, il s'agit d'une méthode plutôt intuitive.

Finalement nous avons présenté deux approches qui mènent directement à la solution implantable. La première est l'approche de la commande supervisée proposée par François Charbonnier dans laquelle il utilise le Grafcet comme outil de modélisation. La deuxième approche est celle de Kattan dans laquelle le concept de contrôlabilité de la théorie de R&W est utilisé pour construire le contrôleur maximalement permissif pour le graphe des situations accessibles d'un grafcet donné. Il reconstruit à partir du modèle automate obtenu un grafcet implantable sur automates programmables industriels (API).

La deuxième méthode de synthèse est basée sur l'utilisation des modèles Rdp du système à contrôler et des spécifications auxquelles des invariant de marquage sont associés. Cette méthode consiste à construire un contrôleur par le calcul de places de contrôle, puis passer d'une classe particulière de Rdp appelle Rdp sauf ou binaire au modèle grafcet du système contrôlé. L'existence de cas où les places de contrôles ont

des sorties vers des transitions incontrôlables remet en cause l'optimalité du contrôleur, mais ne conduit pas à un mauvais fonctionnement.

Ces approches constituent une importante démarche vers de nouveaux horizons dans la synthèse et la mise en œuvre de commandes capables de respecter un cahier des charges malgré la complexité des systèmes à étudier.

Enfin, cette étude nous a permis d'approfondir nos connaissances sur des outils de modélisation comme les automates à états finis, les langages formelles, les réseaux de Petri ainsi que le passage de l'un de ces modèles à un modèle Grafcet.

Bibliographie

[1] **Bassam KATTAN**, *" synthése structurelle d'un controleur basée sur le Grafcet"*, doctorat de l'université JOSEPH FOURIER-GRENOBLE I, 2004.

[2] **Abbas DIDEBAN**, *"synthese de controleur discrets par simplification de contraintes et de conditions"*, Doctorat de l'université de JOSEPH FOURIER-GRENOBLE I, 2007.

[3] **Mustapha NOURELFATH**, *"un nouvel algorithme pour la synthese de la commande ds systemes à événements discrets",* université du QUEBEC en Abitibi-Témiscamingue(UQAT), Nº069.

[4] **A.BOUFADEN, L.PIETRAC et S. GABOUJ**, *"l'usage des réseaux de Petri dans la théorie de contrôle par supervision"*, INSA Lyon et INSAT Tunisie, Nº02-V8.

[5] **Philippe QUEINNEC**, *"automates à états finis"*, 2011.

[6] **Claude KAISER**, *"les réseaux de Petri "*, 2001.

[7] **Jean-Louis FERRIER**, *"commande par supervision : outils et synthese",* université d'ANGERS.

[8] **Olivier BOURNEZ**, *" expressions régulieres. Automates finis"*, cours de LIX, Ecole polytechnique.

[9] **Pierre CASTAGNA**, *"contribution à la modelisation, la simulation et la commande des systemes de production et de transitique"*, Doctorat de l'université de NANTES , 2004.

[10] **K.YAMALIDOU, J.MOODY, M.LEMMON et PANAOS**, *"feedback of Petri nets based on place invariants"*, 1994.

[11] **John O. MOODY**, *"Petri net supervisors for discret event systems"*, doctorat de l'université de Notre dame, Indiana, 1998.

[12] **R.DAVID et H.ALLA**, *"du grafcetaux réseaux de Petri"*, editions Hermes, Paris, 1997 ;

[13] **R.KARA**, *"productique"*, cours Master II Académique, université de Mouloud MAMMERI, TIZI-ouzou, 2011.

[14] **Richard GOURDEAU**, *"le grafcet "*, cours d'école polytechnique de Montréal, 2008.

[15] **C.CASSANDRAS et S. LAFORTUNE**, *"introduction to discret event system second edition"*, spring science+ business, LCC, USA, 2008.

[16] **Arnaud HBERT**, *"commande des systemes : introduction à la modélisation et au contrôle des systemes automatiques"*, didactiques, 2008.

[17] **Abdelouahed TAJER**, *"genie informatique, automatique et traitement du signal"*, doctorat de l'université de REIMS CHAMPAGNE ARDENNE, 2005.

[18] **Robert VALETTE**, *"le grafcet une introduction"*

http://WWW.laas.fr/~robert