Synthèse de contrôleurs discrets par simplification de contraintes et de conditions

Abbas Dideban

Université Joseph Fourier - Grenoble I

N°. attribué par la bibliothèque

|_|_|_|_|_|_|_|_|_|_|

THESE

Pour obtenir le garde de

DOCTEUR DE L'UJF

Spécialité : Automatique - Productique

Préparée au Laboratoire GIPSA – Département Automatique

Dans le carde de l'Ecole doctorale **« Electronique, Electrotechnique, Automatique, Télécommunication et signal »**

Présentée et soutenue par :

Abbas DIDEBAN

Le 31 mai 2007

Titre :

Synthèse de contrôleurs discrets par simplification de contraintes et de conditions

Directeur de thèse **: M. Hassane ALLA**

Jury :

Rapporteur	**Jean-Marc Faure**	**Professeur institut supérieur de mécanique de Paris**
Rapporteur	**Jean-Louis Ferrier**	**Professeur ISTIA, Université d'Angers**
Examinateur	**Eric Niel**	**Professeur INSA DE LYON**
Examinateur	**Piere Ladet**	**Professeur INP Grenoble**
Examinateur	**Hassane Alla**	**Professeur Université Joseph Fourier**

Remerciements

Tout d'abord, je remercie chaleureusement mon directeur de thèse professeur Hassane ALLA qui a été pour moi plus qu'un directeur de thèse. Je le remercie grandement que ce soit pour ces qualités scientifiques ou humaines, faisant toujours preuve de réflexion et de patience.

Je remercie vivement les rapporteurs de ce mémoire de thèse, M. Jean- Marc FAURE professeur à l'institut supérieur de mécanique de Paris, et M. Jean – louis FERRIER professeur à l'université de l'Angers. Merci pour leurs commentaires précis et judicieux qui m'aide beaucoup pour améliorer la qualité de ma thèse. Je remercie à M. Eric NIEL professeur à l'INSA de Lyon et M. Pierre LADET professeur à l'INP Grenoble qui leurs aussi m'ont fait l'honneur d'avoir accepté de porter un jugement sur mon travail de recherche de faire partie de jury de soutenance de ma thèse.

Je tiens à remercie mes amis au laboratoire spécialement Khaled qui m'aide beaucoup pour corriger ma thèse et aussi je remercie mes amis iraniens pour leur soutiens au tout au long de mon séjour en france.

Je n'oublierai pas l'ensemble de membres de l'équipe administrative et technique de département automatique de laboratoire Gipsa, dont je remercie pour leur gentillesse, leur bonne humeur et leur efficacité.

Merci enfin à mon épouse et ma file et mes parents, pour …tout ;

Résumé

Dans ce travail, nous proposons une méthode systématique et facile de mise en œuvre pour la synthèse du contrôle des systèmes à événements discrets. Nous modélisons les systèmes par des modèles RdP saufs. Deux idées distinctes sont utilisées : 1) ajout de places de contrôle pour empêcher l'atteignabilité des états interdits, et 2) ajout de conditions pour les transitions contrôlables. Nous avons alors été confrontés au problème des transitions incontrôlables pour garantir l'optimalité et à la complexité apportée par le nombre des places de contrôles qui peut être très grand.

Dans la première idée, nous avons utilisé le théorème introduit par Guia, qui permet de passer d'un ensemble d'états interdits vers un ensemble de contraintes linéaires. Nous avons proposé des méthodes originales de simplification des contraintes. Il est alors possible de réduire le nombre et la borne des contraintes et ainsi de construire un modèle contrôlé simple. Les méthodes de simplification présentées sont applicables sur les RdP saufs. Nous avons déterminé les conditions nécessaires et suffisantes pour avoir un contrôleur maximal permissif. L'avantage principal de ces méthodes de synthèse de contrôleurs est que le modèle RdP contrôlé est très proche du modèle initial.

La deuxième idée qui a été utilisée pour la synthèse est l'utilisation des conditions pour le franchissement des transitions contrôlables. Les méthodes qui utilisent cette technique, ont en général besoin d'un calcul long en temps réel. En appliquant notre méthode de simplification, nous arrivons à un contrôleur simple.

Mots clés : *Systèmes à événements discrets, Synthèse du contrôleur, Réseaux de Petri, Contraintes linéaires, Simplification*.

Synthesis of discrete controllers by simplification of constraints and conditions

Abstract:

In this work, we propose a systematic method for controller synthesis in discrete events systems. We model the process and the specification by safe Petri Nets (PN). Two distinct ideas are used: 1) adding the control places to prevent the reachability of forbidden states, and 2) adding conditions with the controllable transitions. The uncontrollability asks the problem of optimatlity and the large number of control places the problem of complexity.

In the first idea, we use the theorem introduced by Guia, which makes it possible to pass from the set of forbidden states to the set of linear constraints. We propose original methods of simplification of the constraints. It is then possible to reduce the number and the bound of the constraints and thus to build a simple controller model. The methods of simplification presented are applicable on safe PN. We determine the necessary and sufficient conditions to have a maximal permissive controller. The principal advantage of these methods is that the controlled PN model is very close to the PN initial model.

The second idea for controller synthesis is the using of conditions for controllable transitions. The methods which use this technique generally need a long calculation in real time. While applying our method of simplification, we arrive to a simple controller.

Key words: *Discrete events systems, Controller synthesis, Petri Nets, Linear constraints, Simplification.*

Table des matières

Table des figures

Introduction

Le contrôle de systèmes industriels à cause de l'automatisation et la réduction de nombre des opérateurs devient un enjeu crucial. De tels systèmes intègrent de nombreux composants matériels et logiciels et interagissent avec des environnements complexes. Ainsi la synthèse de la commande de ces systèmes devient difficile, et elle ne peut être basée sur des essais et corrections successives, coûteux et approximatifs. L'expérience et la perspicacité de l'ingénieur qui conçoit ces systèmes deviennent insuffisantes pour imposer le respect du cahier des charges. Par conséquent, il est nécessaire de disposer d'outils aidant à la conception de systèmes de commande et permettant de vérifier si le système commandé répond bien au cahier des charges.

L'objectif de cette thèse est de proposer une méthode de synthèse d'un contrôleur simple de mise en œuvre qui impose au système le respect des spécifications logiques de fonctionnement.

Pour atteindre cet objectif il faut d'abord modéliser le système (l'objet physique) que l'on désire contrôler, et ensuite exprimer les objectifs de commande (spécifications) et finalement utiliser des techniques mathématiques classiques qui permettent d'élaborer les contrôleurs correspondants. Les systèmes à événements discrets (SED) recouvre des systèmes également dynamiques, mais dont la dynamique est du type événementiel. En réalité, au lieu de s'intéresser au déroulement continu des phénomènes, on ne se soucie que des ''débuts'' et des ''fins'' de ces phénomènes (les événements discrets) et de leur enchaînement dynamique, logique ou temporel. Les modèles SED sont utilisés dans le domaine de la production manufacturière, la robotique, les trafics des véhicules, la logistique, les réseaux de communications, etc.

L'étude des systèmes à événements discrets peut être menée avec différents outils tels que la simulation sur ordinateur, les réseaux de files d'attente, les langages de programmation parallèle/temps réel, des modèles dynamiques algébriques, comme l'algèbre ''Max Plus'', et finalement les réseaux de Petri (RdP), les automates et les langages qui reposent sur l'ordre exact d'occurrence des événements.

Ce sont Ramadge et Wonham (Ramadge et al. 1989, Wonham 2005) qui à l'origine ont introduit la théorie de la synthèse de contrôleurs sur les systèmes à événement discrets. Le système, ainsi que l'ensemble des comportements valides (*Spécification*) sont donnés chacun par un langage. A partir de ces deux langages, le but de la synthèse consiste alors à déterminer un sous ensemble des comportements du système qui appartienne aussi à la spécification. Ce sous ensemble représente les comportements du système subissant l'action du contrôleur et caractérise ce dernier.

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

autre distinction essentielle entre le procédé et le superviseur est que le procédé est donné, il correspond à un système existant qui ne peut être changé, tandis que le superviseur est un système à construire, donc plus souple.

L'approche de R&W est basée sur la modélisation des systèmes par des automates à états finis et des langages formels. Cependant, le grand nombre d'états à considérer pour représenter le comportement du système ainsi que le manque de structure dans les modèles limite la possibilité de développer des algorithmes de calcul efficaces pour l'analyse et la synthèse des systèmes réels. En outre l'utilisation de cette approche a montré une difficulté dans l'implémentation du système supervisé. Pour pallier ces difficultés, différentes approches peuvent être envisagées. (Holloway et al. 1990, Li et al. 1993, Boel et al. 1995, Charbonnier 1996, Kattan 2004,…). Ces approches sont basées soit sur les Réseaux de Petri (RdP) soit sur le Grafcet.

Le problème principal de la théorie de supervision est l'existence des événements incontrôlables. Lors de synchronisation d'un événement incontrôlable de modèle procédé avec celui de la spécification, le modèle final ne peut pas respecter cette synchronisation. Dans ce cas il apparaîtra un ensemble d'états qui s'appelle « *états interdits* ». Pour les approches basées sur l'automate on peut le résoudre par l'algorithme proposé par Kumar (Kumar 1991). L'inconvénient principal est que le contrôleur est donné sous forme d'automate dont la taille peut être inexploitable.

Deux méthodes efficaces pour résoudre le problème des états interdits ont été proposées par Giua (Giua et al.1992) et Yamalidou (Yamalidou et al. 1996). Giua a proposé une méthode représentant les états interdits par un ensemble des contraintes linéaires. Cette méthode est applicable pour les RdP sauf et conservatifs. Yamalidou a proposé une méthode pour construire un contrôleur à partir des contraintes linéaires. Par l'utilisation de ces méthodes, on peut construire un contrôleur efficace en modèle RdP. Les problèmes de l'approche proposée par Giua est qu'en général le nombre d'états interdit est grand et donc nous avons un grand nombre des contraintes linéaires. A chaque contrainte il faut associer une place de contrôle augmentant ainsi la complexité du contrôleur. Notre contribution va consister à proposer deux méthodes de réduction du nombre des contraintes, ce qui permettra de diminuer la complexité de système. Ces méthodes sont basées sur l'exploitation de l'espace d'état pour réduire au maximum le nombre et la borne des contraintes. Notre objectif final est d'avoir un modèle contrôlé par un RdP ou un Grafcet, les plus simples possibles et implémentables dans des automates programmables industriels.

D'autres méthodes synthétisent le contrôle par la détermination de conditions associées aux transitions (Holloway et al. 1990, Boel et al. 1995). L'inconvénient de ces approches est le temps de calcul long en temps réel. Pour résoudre ce problème, nous construisons le graphe de marquage hors ligne et calculons les conditions pour empêcher l'atteignabilité des états interdits. Nos méthodes de simplification permettront d'avoir un contrôleur final aisément implémentable.

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

Le chapitre deux présente dans un premier temps les principales méthodes existant dans la littérature sur la commande des systèmes à événements discrets. La théorie de la supervision des SED proposée par Ramadge & Wonham est d'abord présentée en détail. Elle est à l'origine de la commande par supervision. Puis les approches de la synthèse de commande basées sur le Réseaux de Petri sont décrites.

Nous présentons dans le troisième chapitre, la première méthode de simplification des contraintes qui est applicable sur les RdP saufs et conservatifs. Cette méthode de simplification est basée sur l'invariant de marquage des places. Les relations déduites des invariants et également les inégalités déduites des invariants partiels, nous aident pour la simplification. Nous montrons qu'il est possible d'avoir un contrôleur maximal permissif pour ce type de RdP. A la fin de ce chapitre pour montrer que cette méthode est applicable sur les systèmes industriels, nous décrivons la transformation d'un RdP contrôlé vers le Grafcet.

Dans le quatrième chapitre nous introduisons une autre méthode de simplification qui est applicable au RdP sauf dans le cas général. Cette méthode est basée sur la notion de *sur-état*. En utilisant cette notion, nous présentons les propriétés qui peuvent nous aider pour la simplification. Le fait de ne pas avoir un RdP conservatif fait que le contrôleur synthétisé n'est pas toujours maximal permissif. Nous élaborons alors des propriétés qui donnent les conditions nécessaires et suffisantes pour avoir un contrôleur maximal permissif.

Le cinquième chapitre propose une nouvelle méthode de synthèse de contrôleurs basée sur l'idée principale de retour d'état. A partir des états interdits nous calculons l'ensemble des états critiques et des états sains et nous interdisons le franchissement des transitions contrôlables par un contrôle basé sur ces états, et qui sera simplifié par la suite. Cette méthode est applicable sur les RdP saufs mais comme nous verrons dans les perspectives, il est possible développer cette méthode pour les RdP non saufs.

Enfin, nous terminerons notre mémoire par une conclusion et les perspectives de ce travail.

Chapitre 1

Systèmes à Evénements Discrets

Dans ce chapitre nous allons présenter les notions de base sur les systèmes à événements discrets (SED). Nous commencerons par une introduction puis nous présenterons brièvement les deux outils de base d'étude des SED que sont les automates et les Réseaux de Petri (RdP). Nous mettrons l'accent sur les RdP dans la mesure où c'est le modèle de base utilisé dans cette thèse, on donnera les principales propriétés. La mise en œuvre des modèles RdP sera faite par le Grafcet qui est un langage de spécifications qui a conduit au langae SFC (Sequantial Fonction Chart), un outil couramment utilisé dans les automates programmables industriels. Une présentation de cet outil est faite à la fin de ce chapitre.

Chapitre 1

Systèmes à Evénements Discrets

1.1 Introduction

Un système à événements discrets (SED) est un système dynamique dans lequel l'espace d'état est discret. L'évolution de ces systèmes est déterminée par l'occurrence instantanée des événements. Ainsi, tant qu'il n'y a pas d'occurrence d'un événement, l'état du système reste inchangé. Ces systèmes se trouvent dans des nombreux domaines d'application tels que les systèmes de production, l'informatique, les réseaux de communication, etc.

Considérons l'exemple classique d'une machine qui peut être dans trois états : *arrêt*, *marche* et *panne*. On suppose qu'il peut y avoir l'occurrence de quatre événements : *début de travail*, *fin de travail*, *panne* et *réparation*. Ces événements sont étiquetés par les symboles d, f, t et r respectivement. L'ensemble des états de ce système est présenté par l'ensemble $\mathcal{E} = \{a, m, p\}$. Le changement des états par l'occurrence des événements est présenté dans la figure 1.1.

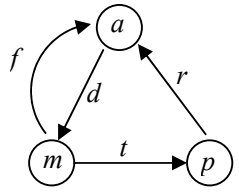


Fig. 1.1 : Changement d'état par occurrence des événements dans un SED

En général pour construire un contrôleur, les concepteurs construisent directement le modèle supervisé du système. Mais pour les grands systèmes, on n'a aucune garantie que les spécifications seront respectées. Il faut alors passer par la simulation avec les inconvénients que l'on connaît. Il existe des méthodes formelles dont nous parlerons dans la suite de ce mémoire. Ces méthodes reposent sur l'utilisation d'outils tels que les automates et les réseaux de Petri que nous allons présenter en détail dans ce chapitre.

1.2 Langage

1.2.1 Définition de langage

L'évolution d'un SED peut être décrite par un ensemble de couples (e, t) où « e » représente un événement et « t » représente l'instant d'occurrence de cet événement. Dans notre exemple,

l'évolution de l'état de la machine peut être définie par les couples : (d, t_1), (f, t_2), (d, t_3), (t, t_4), (r, t_5),…etc. Cet ensemble ordonné de couples constitue ce que l'on appelle une séquence. Une telle description se place à un niveau temporel dans le sens où l'instant d'occurrence des événements est une information considérée comme pertinente. En revanche, si l'on considère un modèle logique pour décrire le SED, seul l'ordre d'occurrence des événements importe. Dans ce contexte, le fonctionnement de la machine est décrit par la séquence des événements : *dfdtr*….etc. L'évolution d'un SED sera en général décrite par un ensemble de séquences d'événements. Cet ensemble de séquences constitue un *Langage* sur l'ensemble des événements possibles dans notre système.

Pour formaliser des systèmes à événements discrets sous forme de langage, on représente les événements par des symboles. L'ensemble de tous ces symboles (σ_1 , …, σ_n) est fini et constitue un alphabet noté Σ. Toutes les séquences finies d'événements ou traces, peuvent alors être représentées par une séquence de symboles $s = \sigma_1. \sigma_2 \ldots$ appelée chaîne (ou mot) sur l'alphabet Σ. On appelle alphabet (ou vocabulaire), un ensemble fini de symboles noté Σ. Dans le cas d'un SED, l'alphabet pourra représenter l'ensemble des événements possibles dans le système. Cet ensemble est composé de tous les événements qui font évoluer le SED. Une *chaîne* (ou *mot* ou *séquence*) définie sur un alphabet Σ est une suite finie d'éléments de Σ noté s. La longueur d'une chaîne s, notée $|s|$, représente le nombre de symboles de s (par exemple, $|dfdtr| = 5$).

Etant donné un alphabet Σ, on notera Σ^n où n est un entier naturel, l'ensemble des chaînes sur Σ de longueur n. Par exemple, pour l'alphabet $\Sigma=\{d, f\}$, nous obtenons : $\Sigma^0 = \{ \varepsilon \}$ où ε représente le mot vide (qui ne comporte aucune symbole) ; $\Sigma^1 = \{d, f\}$et $\Sigma^2 = \{ dd, df, fd, ff \}$, ….etc. Par extension, on définira par Σ^* l'ensemble des chaînes d'une longueur n quelconque que l'on peut construire sur Σ. Ainsi Σ^* peut être défini par :

$$\sum^* = \bigcup_{i \geq 0} \sum^i$$

Où ∪ représente le symbole d'union. Etant donné l'alphabet $\Sigma=\{d, f\}$, nous obtenons alors :

$\Sigma^* = \{\varepsilon, d, f, dd, df, fd, ff, ddd, \ldots\}$

Définition 1.1 : On appelle un langage défini sur un alphabet Σ, tout sous-ensemble de Σ^*.

❑

Par exemple l'ensemble des séquences telles que « d » apparaît en premier et que « d » et « f » apparaissent alternativement est décrit par le langage : L = {ε, d, df, dfd, $dfdf$, $dfdfd$, … }. On notera par Φ, le langage vide.

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

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

1.2.2 Langages réguliers

Pour définir un *langage régulier* on définit d'abord l'expression régulière.

Une expression régulière sur un alphabet ∑ est une expression dont les opérandes sont des symboles de ∑, et dont les opérateurs sont pris dans l'ensemble {+, · , *}. Par exemple l'expression : $b+a \cdot b^*$ est une expression régulière.

Une expression régulière définit un langage sur ∑. Chaque symbole « a » de ∑ correspond au mot a. Les opérateurs +, · et * sont interprétés respectivement comme les opérateurs d'union, de concaténation et de fermeteure itérative sur des langages. Par exemple, l'expression régulière : $b + a \cdot b^*$ représente le langage: $\{b\} \cup \{a\} \cdot \{b\}^*$

On appelle *langage régulier*, tout langage qui peut être défini par une expression régulière. Par exemple, considérons le langage L sur l'alphabet $\sum = \{a,b\}$qui comprend les chaînes telles que a et b

apparaissent alternativement et telles que *a* apparaît en premier. Nous obtenons : $L = \{\varepsilon, a, ab, aba, abab, ababa, \ldots\}$. Ce langage est régulier puisqu'il peut être décrit par l'expression régulière : $(a.b)^* \cdot (\varepsilon + a)$.

1.2.3 Opérations sur les langages

Soit L_1 et L_2 deux langages construits sur un même alphabet Σ. On peut définir les opérations élémentaires suivantes :

L'*union* de L_1 et L_2, notée, $L_1 \cup L_2$, est le langage contenant toutes les chaînes qui sont soit contenues dans L_1, soit dans L_2.

L'*intersection* de L_1 et L_2 notée, $L_1 \cap L_2$, est le langage contenant toutes les chaînes qui sont soit contenues à la fois dans L_1 et dans L_2.

La *concaténation* de L_1 et L_2 notée, $L_1 \cdot L_2$, est le langage contenant toutes les chaînes formées d'une chaîne de L_1 suivie d'une chaîne L_2.

La *fermeture itérative* d'un langage L_1, notée L_1^*, est l'ensemble des chaînes formées par une concaténation finie de chaînes de L_1.

1.3 L'automate

1.3.1 Définition générale

Un automate est une machine à états qui permet de décrire un système à événements discrets. Un automate est représenté par une succession d'états et de transitions associées à des événements.

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

Définition 1.1 : Un automate fini A est un 5-tuplet $A = (Q, \Sigma, \delta, q_0, Q_m)$ où

Q est l'ensemble fini des états.

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

δ est la fonction de transition d'états de $Q \times \Sigma$ vers Q qui associe un état d'arrivée à un état de départ et un symbole d'entrée.

$q_0 \in Q$ est l'état initial.

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

❑

Un automate peut être représenté par son *graphe de transitions d'états*. Le graphe de transition d'états d'un automate fini A est donné dans la figure 1.2.

Dans la figure 1.2 les états de A sont représentés par des cercles, nous avons : $Q = \{q_0, q_1, q_2, q_3\}$. L'état initial q_0 est indiquée par une flèche entrante. Les états finals sont représentés par des doubles cercles, ainsi : $Q_m = \{q_3\}$. La fonction δ de transition d'états est représentée par des arcs associés à des symboles de l'alphabetΣ. Dans notre exemple, l'alphabet Σ correspond à l'ensemble $\{a, b, c\}$. Il existe une transition d'états associée au symbole « a » entre q_0 et q_1. Cela signifie : $\delta(q_0, a) = q_1$.

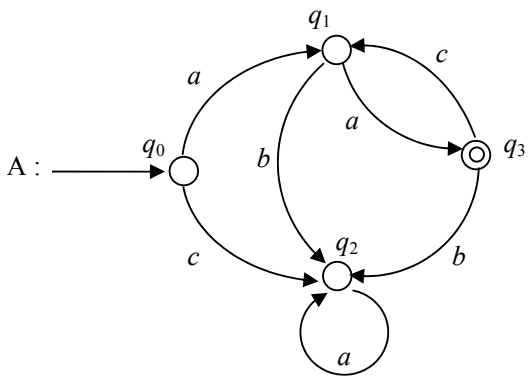


Fig. 1.2 : Graphe de transition d'états d'un automate fini.

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

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

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

C'est un langage régulier qui représente l'ensemble de toutes les chaînes qui permettent de rejoindre un état quelconque de l'automate à partir de son état initial.

Remarque 1.1 : L'automate A est dit *déterministe* dans le sens où depuis tout état, il n'existe pas deux transitions de sortie qui soient associées à un même symbole et qui conduisent à deux états différents.

❑

1.3.2 Composition synchrone de deux automates

Avant de présenter cette opération, nous définissons un type spécial d'automate. On appelle *accepteur* un automate ne comportant pas de sorties. Formellement, on peut définir un accepteur de la façon suivante :

Définition 1.2 : un accepteur A est un quadruplet $A = (Q, \sum, \delta, q_0)$ où Q est l'ensemble fini des états, $\sum$ est un ensemble fini de symboles d'entrée (l'ensemble des événements), appelé alphabet d'entrée, δ est la fonction de transition d'états de $Q \times \sum$ vers Q qui associe un état d'arrivée à un état de départ et un symbole d'entrée et $q_0 \in Q$ est l'état initial.

❑

Un tel modèle ne comporte pas d'état final. Dans cette approche nous intéressons à ce type d'automate et le mot *automate* sera utilisé au lieu d'accepteur. A partir de cette hypothèse, la composition synchrone entre deux automates sera présentée.

Il existe deux sortes de compositions :

La *composition synchrone*, qui est effectuée quand les alphabets des langage associés aux automates considérés ont au moins un événement en commun.

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

Soit deux automates A_1 et A_2 définis respectivement par :

$$A_1 = (Q_1 , \Sigma_1 , \delta_1 , q_{01}) , \qquad A_2 = (Q_2 , \Sigma_2 , \delta_2 , q_{02})$$

La composition synchrone entre A_1 et A_2 est notée $A_1 \|_s A_2 = (Q, \Sigma, \delta, q_0)$ où

$Q = Q_1 \times Q_2$ est l'ensemble des états.

$\Sigma = \Sigma_1 \cup \Sigma_2$ représente l'alphabet.

$q_0 = (q_{01} , q_{02})$ est l'état initial.

Pour tout $q = (q_1, q_2) \in Q$ et pour tout $\sigma \in \Sigma$ la fonction de transition δ est définie de la façon suivante :

- Pour $\sigma \notin \Sigma_1 \cap \Sigma_2$:

$$\delta((q_1, q_2), \sigma) = (q'_1, q_2) \text{ si } \delta_1(q_1, \sigma) = q'_1 \text{ et } \sigma \in \Sigma_1$$

$$\delta((q_1, q_2), \sigma) = (q_1, q'_2) \text{ si } \delta_2(q_2, \sigma) = q'_2 \text{ et } \sigma \in \Sigma_2$$

- Pour $\sigma \in \Sigma_1 \cap \Sigma_2$:

$$\delta((q_1, q_2), \sigma) = (q'_1, q'_2) \text{ si } \delta_1(q_1, \sigma) = q'_1 \text{ et } \delta_2(q_2, \sigma) = q'_2$$

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

Exemple 1.1 : Le fonctionnement du distributeur de jeton est décrit par les 2 modèles automates (accepteurs) de la figure 1.3.

Fig. 1.3. Modèles automates d'un distributeur de jeton

L'automate A_1 de la figure 1.3a est construit sur un alphabet de 2 symboles $\{p, j\}$. Nous pouvons remarquer que l'automate A_1 reconnaît l'ensemble des séquences telles que p se produit en premier et telles que p et j se produisent alternativement. Supposons que le symbole « p » modélise l'événement « introduction d'une *p*ièce » dans la machine et que « j » modélise l'événement « libération d'un *j*eton ». Alors, A_1 modélise la contrainte de fonctionnement que le jeton est délivré seulement si une pièce a préalablement été introduit dans la machine (nous supposons que les événements p et j apparaissent toujours alternativement).

L'automate A_2 de la figure 1.3b est construit sur un alphabet de 2 symboles $\{b, j\}$.Le symbole « b » modélise l'événement « pression du *b*outon par l'utilisateur » et le symbole « j » a la même signification que dans A_1 (libération d'un *j*eton). Ce modèle exprime la contrainte qu'un jeton est délivré seulement si l'utilisateur a préalablement appuyé sur le bouton.

A partir de A_1 et A_2, l'opération de *composition synchrone* permet d'obtenir un modèle global de notre machine. Le composé synchrone M de A_1 et A_2 est représenté dans la figure 1.4. Ce modèle permet d'exprimer la conjonction des deux contraintes de fonctionnement définies par A_1 et par A_2.

Dans l'automate M, un état correspond à un couple (q, x) où q est un état de A_1 et x est un état de A_2. L'état initial (q_0, x_0) est l'état correspondant aux états initiaux de A_1 et de A_2. L'automate M est construit sur un alphabet qui est la réunion des alphabets de A_1 et A_2, c'est-à-dire : $\{p, b, j\}$, où le symbole j est le seul événement qui est en commun dans les alphabets de A_1 et A_2.

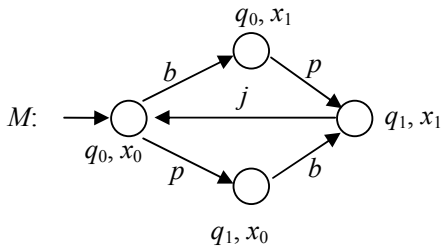


Fig. 1.4. Modèle automate complet d'un distributeur de jeton

La modélisation de SED réels est généralement difficile à mettre en œuvre du fait de l'explosion combinatoire du nombre d'états (même dans les petits systèmes) et même si les automates apparaissent comme un formalisme approprié permettant de modéliser efficacement une large classe de systèmes à événement discrets, certaines applications s'expriment plus facilement sous forme de Réseaux de Petri (RdP). Dans la prochaine section, nous allons faire quelques rappels sur les réseaux de Petri. Pour plus de détails sur les réseaux de Petri, on pourra se reporter à (David et al. 2005).

1.4 Réseaux de Petri

Les réseaux de Petri constituent un outil très approprié pour étudier les systèmes à événements discrets en raison de la puissance de modélisation et de leurs propriétés mathématiques. Les réseaux de Petri ont, par rapport aux automates, l'avantage d'être un modèle beaucoup plus général, bénéficiant de structures beaucoup plus riches, s'adaptant parfaitement à la description de certains type de SED. A partir de RdP, on peut facilement arriver à l'automate par construction du graphe des marquages. Dans cette section nous présenterons d'abord les notions de base de RdP dont nous avons besoin dans notre travail. Le type de RdP qui nous intéresse, c'est le RdP synchronisé où à chaque transition est associé un événement.

1.4.1 Notions de base

Un Réseau de Petri est un graphe orienté comportant :

- Un ensemble fini de places, $P = \{P_1, P_2, P_3, ..., P_m\}$, symbolisées par des cercles et représentant des conditions :

  Une ressource du système (ex. : une machine, un stock, un convoyeur, …)

  L'état d'une ressource du système (ex. : machine libre, stock vide, convoyeur en panne, …)

  …etc

- Un ensemble fini de transitions, $T = \{T_1, T_2, T_3, ..., T_n\}$, symbolisées par des traits et représentant l'ensemble des événements (les actions se déroulant dans le système) dont l'occurrence provoque la modification de l'état du système.
- Un ensemble fini d'événements associés à chaque transition.
- Un ensemble fini d'arcs orientés qui assurent la liaison d'une place vers une transition ou d'une transition vers une place.

Définition 1.3 : Un RdP synchronisé est défini par un 7-tuplet $R = \{\mathcal{P}, \mathcal{T}, \sum, E, Pré, Post, M_0\}$ où

$\mathcal{P} = \{P_1, P_2, P_3, \ldots, P_N\}$ est l'ensemble des places ;

$\mathcal{T}$= $\{T_1, T_2, T_3,\ldots, T_L\}$ est l'ensemble des transitions ;

$\mathcal{P} \cap \mathcal{T} = \varnothing$; c'est à dire que les ensemble $\mathcal{T}$ et $\mathcal{P}$ sont disjoints ;

$\sum = \{\varepsilon, \sigma_1, \sigma_2,\ldots, \sigma_n\}$ est l'ensemble des événement (ε a été utilisé pour présenter l'événement toujours occurrent) ;

E est la fonction de $\mathcal{T} \rightarrow \sum$ qui associe à chaque transition un événement ;

Pré : $\mathcal{P} \times \mathcal{T} \rightarrow \{0,1\}$ est l'application d'incidence avant ;

Post : $\mathcal{T} \times \mathcal{P} \rightarrow \{0,1\}$ est l'application d'incidence arrière ;

M_0 est l'état initial $\mathcal{P} \rightarrow R$.

Au lieu de *Pré* et *Post* en général nous utilisons *W*, la matrice d'incidence qui est calculable à partir de l'ensemble *Pré* et *Post* comme ci-dessous :

$W = Post - Pré$

Dans un RdP le nombre de marques dans les places est quelconque. Cependant dans notre travail, on s'intéresse à la commande, et nous ne considérerons que les RdP saufs où chaque place contient au plus une marque. Ce type de RdP est appelée *sauf ou binaire*.

1.4.2 Le marquage

Dans les RdP, nous pouvons représenter la dynamique du système par l'existence ou l'absence de marque dans les places. La figure 1.5 représente un RdP marqué.

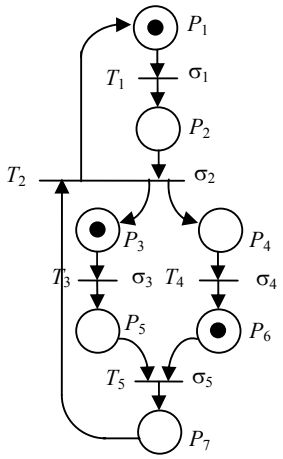


Fig. 1.5. Réseau de Petri marqué

Chaque place contient un nombre entier (positif ou nul) de marques ou jetons. Le nombre de marque contenu dans une place P_i sera noté $M(P_i)$ soit m_i. Pour l'exemple considéré, on a $m_1 = m_3 = m_6 = 1$ et $m_2 = m_4 = m_5 = m_7 = 0$. Le marquage du réseau, *M* est défini par le vecteur de ses marquages, c'est-à-dire $M_0^T = [m_1, m_2, m_3, m_4, m_5, m_6, m_7]$. Le marquage du RdP de la figure 1.6 est donc : $M_0^T = [1, 0, 1, 0, 0, 1, 0]$. Le marquage est l'état du système à un certain instant. L'évolution de l'état correspond donc à une évolution du marquage, évolution qui se produit par le franchissement de transition, comme nous allons le voir.

Dans cette approche nous allons utiliser autre présentation pour l'état. Dans les réseaux saufs, une place peut être marquée ou non, donc si l'existence du nom de places est utilisée pour les places marquées, l'absence de ces noms signifie que les places ne sont pas marquées. Par exemple le

marquage M peut être représenté par l'ensemble des places marquées : $M_0 = \{P_1P_3P_6\}$ ou plus simplement : $M_0 = P_1P_3P_6$. Maintenant nous formalisons ce type de présentation :

Notation 1.1 : Soit P = $\{P_1, P_2, \ldots, P_N\}$ l'ensemble des places d'un RdP $\mathcal{R}$ et M : $M^T = [m_1, \ldots, m_N]$ $m_i \in \{0,1\}$, un marquage quelconque, on peut représenter le marquage M comme ci-dessous:

$$M = \{P_i \mid \quad i \in \{1, \ldots, N\} \text{ et } m_i = 1\}$$

❑

Il est possible de définir formellement cet ensemble, mais définissons d'abord l'ensemble des vecteurs booléens.

Définition 1.4 : L'ensemble $\{0,1\}^N$ représente l'ensemble de tous les vecteurs booléens de dimension N.

❑

Un marquage est un vecteur de $\{0,1\}^N$.

L'ensemble des places marquées d'un marquage est donné par une fonction Support définie ci-dessous.

Définition 1.5 : La fonction Support (X) d'une vecteur $X \in \{0,1\}^N$ est telle que :

Support (X) = Ensemble des places marquées dans le vecteur X

❑

Le support du vecteur $M_0^T = [1, 0, 1, 0, 0, 1, 0]$ est :

Support (M_0) = $\{P_1, P_3, P_6\}$ ou plus simplement :

Support (M_0) = $P_1P_3P_6$

Il est possible de comparer deux marquages. Nous utilisons la définition ci-dessous pour cela :

Définition 1.6 : Soit M_1 et M_2 deux marquages accessibles d'un RdP R_1 et $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ l'ensemble des places de ce RdP:

$M_1 > M_2$ Si $\forall P_i \in \mathcal{P} \ / M_1(P_i) \geq M_2(P_i)$ et $\exists P_i \in \mathcal{P} \ / M_1(P_i) > M_2(P_i)$

$M_1 < M_2$ Si $\forall P_i \in \mathcal{P} \ / M_1(P_i) \leq M_2(P_i)$ et $\exists P_i \in \mathcal{P} \ / M_1(P_i) < M_2(P_i)$

$M_1 = M_2$ Si $\forall P_i \in \mathcal{P} \ / M_1(P_i) = M_2(P_i)$

Remarque 1.2 : Pour les RdP saufs, $M_1 < M_2$ signifie que Support (M_1) $\subset$ Support (M_2). Dans ce cas nous disons que le marquage M_2 est couvert par le marquage M_1.

1.4.3 Propriétés des réseaux de Petri

Dans cette section, on considère les réseaux de Petri ordinaires et nous présentons les propriétés qui nous seront nécessaires pour la présentation de notre contribution.

Marquages accessibles : On notera $\mathcal{M}_R$ l'ensemble des *marquages accessibles* d'un RdP $\mathcal{R}$à partir du marquage initial M_0.

RdP borné : Une place P_i est dit *bornée* pour un marquage initial M_0 s'il existe un entier naturel k, tel que pour tout marquage accessible à partir de M_0, le nombre de marques dans P_i est inférieur ou égal à k. Dans notre travail cette borne $k = 1$ (**RdP sauf**,) sauf pour les places de contrôle comme nous le verrons par la suite.

Transition vivante : Une transition T_j est *vivante* pour un marquage initial M_0 si pour tout marquage accessible $M_i \in \mathcal{M}_R$, il existe une séquence de franchissement S qui contient la transition T_j, à partir de M_i.

RdP vivant : Un RdP est *vivant* pour un marquage initial M_0 si toutes ses transitions sont vivantes pour M_0.

Blocage : un *blocage* est un marquage tel qu'aucune transition ne peut être validée.

Conflit structurel : Un *conflit structurel* correspond à un ensemble d'au moins 2 transitions T_1 et T_2 qui ont une place d'entrée en commun P_i.

Conflit effectif dans un RdP sauf : Il y a un conflit effectif dans un RdP *sauf* lorsqu'il y a au moins deux transitions validées synchronisées avec le même événement.

Invariants : Soit $\mathcal{R}$ un RdP et $\mathcal{P}$ l'ensemble de ses places. On a un *invariant de marquage*, s'il existe un sous-ensemble de places $\mathcal{P}'$ inclus dans $\mathcal{P}$ et un vecteur de pondération $Q=(q_1, q_2,\ldots, q_r)$, dont tous les poids q_i sont des nombres entiers positifs, tels que :

$q_1 M(P_1) + q_2 M(P_2)+\ldots+ q_r M(P_r) =$ constante, pour tout $M \in \mathcal{M}_R$.

Nous expliquerons plus en détail cette propriété dans la section 1.4.6.

1.4.4 Franchissement d'une transition

Le franchissement d'une transition ne peut s'effectuer que si chacune des places en amont de cette transition contient au moins une marque. On dit alors que la transition est *franchissable* ou *validée*. Le franchissement d'une transition fait évoluer le marquage du réseau en retirant un marque de chacune des places en amont de la transition et en ajoutant une marque dans chacune des places en aval de la transition. Pour les RdP saufs, il y a un seul franchissement à la fois, et la durée de ce franchissement est nulle.

Dans un RdP synchronisé, une transition validée est franchie à l'occurrence de l'événement qui lui est associé.

Dans la figure 1.6-*a* la transition T_1 est franchissable lorsque l'événement σ_1 se produit. Le résultat du franchissement de cette transition est indiqué dans la figure 1.6-b.

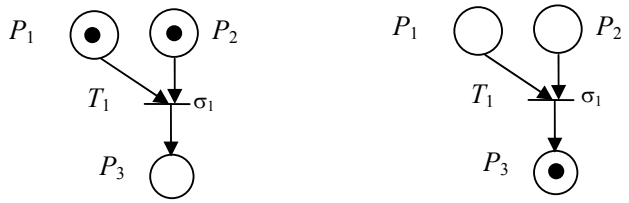


Fig. 1.6. Franchissement d'une transition

A chaque séquence de franchissement, est associé un vecteur caractéristique noté $\bar{s}$. C'est un vecteur de dimension L (nombre de transitions) où la composante numéro j correspond au nombre de franchissements de la transition T_j dans la séquence S. Si la séquence de franchissement S est réalisable à partir d'un marquage M_i, le marquage atteint M_k est donné par l'équation fondamentale :

$$M_k = M_i + W \,.\, \bar{s}$$

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

1.4.5 Ensembles des marquages accessibles

A partir d'état initial par le franchissement des transitions, nous obtenons un ensemble des marquages. Chaque marquage est accessible par une séquence d'événements. Nous pouvons représenter cet ensemble de marquages et des changements entre marquages par un graphe appelé: *graphe des marquages accessibles.*

Le graphe des marquages est composé de nœuds qui correspondent aux marquages accessibles, et d'arcs correspondant aux franchissements de transitions faisant passer d'un marquage à l'autre. Le graphe de marquages d'un RdP est un automate sans états marqués. Formellement un graphe de marquages *G* peut être défini de la façon suivante :

Définition 1.7 : Un graphe de marquages *G* est un 4-tuplet $A = (\mathcal{M}, \sum, \delta, M_0)$ où

$\mathcal{M}$ est l'ensemble fini des marquages (RdP sauf).

$\sum$ est l' ensemble fini des événements associés aux transitions.

δ est la fonction de transition d'états de $\mathcal{M} \times \sum$ vers $\mathcal{M}$ qui à tout couple (état origine et événement) associe un état d'arrivée.

$M_0 \in \mathcal{M}$ est le marquage initial.

❑

La figure 1.5 présente un RdP avec son marquage initial M_0^T = [1, 0, 1, 0, 0, 1, 0]. A partir de M_0, les transitions T_1 et T_3 peuvent être franchies. Si la transition T_1 franchie, on atteint le marquage M_1^T = [0, 1, 1, 0, 0, 1, 0]. Dans ce cas la seule transition franchissable sera T_3. Quand la transition T_3 est franchie, on atteint le marquage M_2^T = [1, 0, 0, 0, 1, 1, 0]. Dans ce cas les deux transitions T_1 et T_5 sont franchissables. Par cette méthode on peut calculer tous les états possibles à partir d'état initial M_0. La figure 1.7 donne le graphe de marquages du RdP de la figure 1.5.

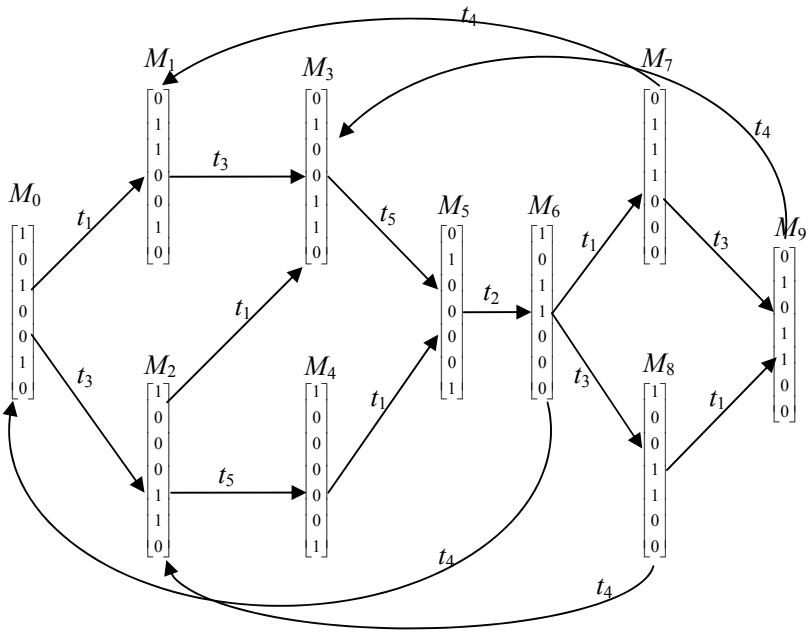


Fig. 1.7. Graphe de marquage du RdP de la figure 1.5

Le graphe de marquages du modèle RdP d'un système est l'automate de ce système en remplaçant chaque transition par l'événement associé. Nous utiliserons donc conjointement les propriétés des deux modèles.

1.4.6 L'invariant de marquage et les réseaux de Petri conservatifs

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

L'invariant de marquage correspond à une somme pondérée de marquages de places qui est constante quel que soit le marquage accessible. Un invariant est déterminé à partir de vecteurs X, calculés en trouvant les solutions de l'équation suivante :

$$X^T . W = 0$$

Où W est la matrice d'incidence du RdP de dimension $N \times L$ et, X est un vecteur de dimension N appelé P-semi-flot[1].

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

$$X^T.M = X^T.M_0 = \text{Constant}$$

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

Il y a des algorithmes efficaces pour trouver tous les invariants minimaux d'un RdP (David et al. 2005).

Dans un invariant, la partie constante correspond à un entier quelconque. Dans cette approche, nous nous intéressons à un cas particulier où les parties constantes des invariants minimaux sont égales à 1.

Pour la majorité des systèmes qui peuvent être modélisés par les RdP saufs, on a des fonctionnements cycliques. Cette propriété d'invariants est donc utilisable dans beaucoup de cas. Par exemple considérons une machine a trois états : marche, repos et panne. Le modèle RdP de cette machine est présenté dans la figure 1.8.

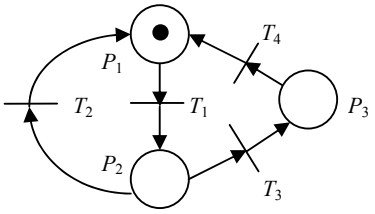


Fig. 1.8. Modèle RdP d'une machine

Dans ce modèle, on a un invariant de marquage pour l'ensemble des places $\mathcal{P} = \{P_1, P_2, P_3\}$:

$$M(P_1) + M(P_2) + M(P_3) = 1$$

On peut réécrire plus simplement cette relation : $m_1 + m_2 + m_3 = 1$

Dans la figure 1.9 on trouve un modèle d'un système de production composé de deux machines couplées. L'ensemble des places $\mathcal{P}$ est : $\mathcal{P} = \{P_1, P_2, P_3, P_3, P_4, P_5, P_6, P_7\}$

[1] - Le vecteur X solution de X^T. W est appelé P- invariant ou P – semi – flot.

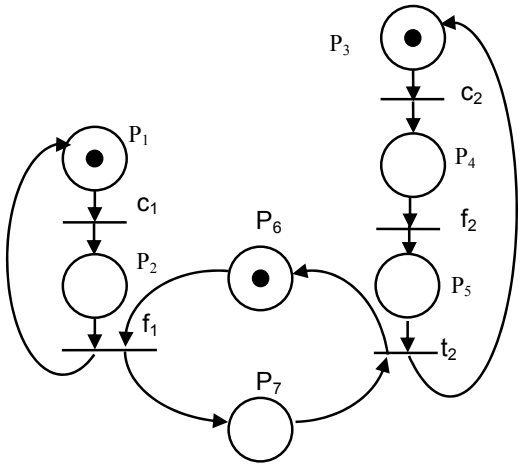


Fig. 1.9. RdP conservatif

Ce RdP a 3 invariants minimaux qui sont :

$$\mathcal{P}' = \{P_1, P_2\} \quad \Rightarrow \quad m_1 + m_2 = 1$$

$$\mathcal{P}'' = \{P_3, P_4, P_5\} \quad \Rightarrow \quad m_3 + m_4 + m_5 = 1$$

$$\mathcal{P}''' = \{P_6, P_7\} \quad \Rightarrow \quad m_6 + m_7 = 1$$

Les ensembles $\mathcal{P}'$, $\mathcal{P}''$, $\mathcal{P}'''$, sont des composantes conservatives. A partir de cet ensemble de relations, nous pouvons construite l'invariant suivant :

$$m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 = 3$$

Dans cet invariant nous pouvons trouver toutes les places du RdP. Ce type de réseau s'appelle : *Réseau de Petri conservatif*. Dans le cas général, un RdP R est dit *conservatif* si $\mathcal{P}$ est une composante conservative. Dans une première partie de notre travail, nous nous intéresserons à ce type de Réseaux de Petri.

1.4.7 Composition synchrone de deux RdP

La synchronisation de deux RdP est une opération plus facile que celle de deux automates. C'est une opération structurelle, elle consiste à fusionner les transitions synchronisées sur le même événement.

Définition 1.8 : Le composé synchrone de deux RdP $\mathcal{R}_1 = \{\mathcal{P}_1, \mathcal{T}_1, \Sigma_1, E_1, Pré_1, Post_1, M_{01}\}$ et $\mathcal{R}_2 = \{\mathcal{P}_2, \mathcal{T}_2, \Sigma_2, E_2, Pré_2, Post_2, M_{02}\}$ est un nouveau RdP $\mathcal{R} = \mathcal{R}_1 \| \mathcal{R}_2 = \{\mathcal{P}, \mathcal{T}, \Sigma, E, Pré, Post, M_0\}$ ou :

$\mathcal{P} := \mathcal{P}_1 \cup \mathcal{P}_2$.

$\Sigma := \Sigma_1 \cup \Sigma_2$

$\mathcal{T} := \mathcal{T}_1 \cup \mathcal{T}_2 - \mathcal{T}_{12}$, $\mathcal{T}_{12} := \{ t_i \in \mathcal{T}_1 \mid \exists\, t_j \in \mathcal{T}_2 / E_1(t_i) = E_2(t_j)\}$

$$E(t_j) = \begin{cases} E_1(t_j) & \text{si } t_j \in \mathcal{T}_1 \\ E_2(t_j) & \text{si } t_j \in \mathcal{T}_2 \end{cases}$$

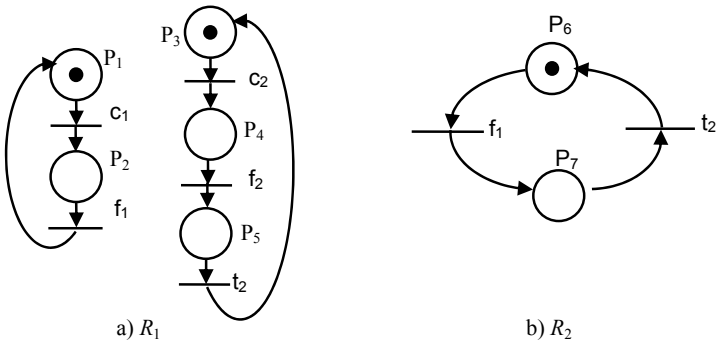
$$Pré\,(p_i\,,\,t_j) = \begin{cases} Pré_1(p_i\,,\,t_j) \text{ si } p_i \in \mathcal{P}_1 \\ Pré_2(p_i\,,\,t_j) \text{ si } p_i \in \mathcal{P}_2 \end{cases}$$

$$Post\,(p_i\,,\,t_j) = \begin{cases} Post_1(p_i\,,\,t_j) \text{ si } p_i \in \mathcal{P}_1 \\ Post_2(p_i\,,\,t_j) \text{ si } p_i \in \mathcal{P}_2 \end{cases}$$

$$M_0(p_i) = \begin{cases} M_{01}(p_i) \text{ si } p_i \in \mathcal{P}_1 \\ M_{02}(p_i) \text{ si } p_i \in \mathcal{P}_2 \end{cases}$$

❑

La synchronisation des RdP $\mathcal{R}_1$ et $\mathcal{R}_2$ présenté dans la figure1.10-a et 1.10-b a été donnée dans la figure 1.9.



a) R_1 b) R_2

Fig. 1.10. Deux modèle RdP avant synchronisation

1.5 Grafcet

Le Grafcet (Graphe Fonctionnel de Commande Etape/Transition) est un graphe qui permet de décrire facilement les fonctionnalités d'un automatisme séquentiel (Blanchard, 1979). Il a été normalisé, dans un premier temps, par l'AFNOR (Association Française de Normalisation) en 1977 et ensuite par la CEI (Commission Electronique Internationale) en 1989. Le Grafcet a été modifié et accepté comme un langage de programmation des automates programmables industriels (API) dans le standard IEC1131. Il porte le nom de SFC (Sequential Function Chart). Dans le standard IEC 60848, le Grafcet est un langage de spécification de systèmes séquentiels. Dans notre travail, nous utiliserons cet outil pour présenter l'implantabilité des résultats de notre approche. Il y a des différences entre un modèle Grafcet et un modèle SFC. Ces différences ajoutent une autre étape de transformation du Grafcet vers SFC. Mais cette transformation se situe en aval de notre approche de simplification. Nous rappelons ci-dessous brièvement les notions de base de Grafcet.

1.5.1 Eléments de base

Un Grafcet est représenté par un graphe qui comporte deux types de nœuds, les étapes et les transitions (un Grafcet contient au moins une étape et une transition). Des arcs orientés relient soit une étape à une transition, soit une transition à une étape.

Une étape dans le Grafcet joue un rôle comme une place dans le RdP. Il y a trois possibilités pour une étape : active, inactive et étape initiale. Ces ensembles d'étapes sont représentés dans la Figure 1.11.

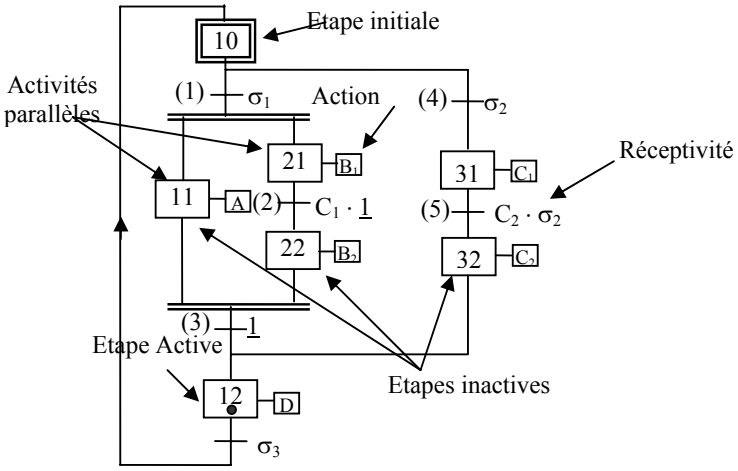


Fig. 1.11. Un Grafcet

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

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

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

1.5.2 Interprétation du graphe

L'interprétation d'un graphe ainsi constitué revient à associer des actions aux étapes et des réceptivités aux transitions, traduisant l'aspect séquentiel du système.

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

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

1.5.3 Règles d'évolution

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

Situation initiale : la situation initiale correspond aux étapes actives au début du fonctionnement. C'est donc le comportement au repos.

Franchissement d'une transition **:** une transition est dite validée lorsque toutes les étapes en amont de cette transition sont actives. Le franchissement d'une transition est effectif lorsque la transition est validée et lorsque la réceptivité associée est vraie.

Activation des étapes : le franchissement d'une transition entraîne immédiatement l'activation des étapes en aval de cette transition et la désactivation de ses étapes en amont.

Evolutions simultanées : plusieurs transitions simultanément franchissables sont effectivement franchies simultanément.

Activation et désactivation simultanée d'une étape : si, au cours du fonctionnement, une étape est simultanément activée et désactivée, alors elle reste active.

1.5.4 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 dans (Blanchard, 1979) (Roussel, 1994) pour la validation de Grafcet s'articulent autour du graphe des situations accessibles. Le but de ces approches :

- Tout d'abord calculer tout les comportements possibles du Grafcet, ce que traduit bien le graphe des situations accessibles.
- Ensuite valider le Grafcet par analyse des comportements obtenus.
- La phase de calcul des comportements du Grafcet consiste à dresser la liste exhaustive des situations accessibles et des possibilités d'évolutions entre ces situations.

La phase de validation repose sur la mise en place d'un langage formel pour l'expression des propriétés à vérifier, et l'utilisation d'un analyseur de systèmes de transitions, comme proposé dans (Leparc, 1994).

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.

1.6 Comparaison RdP sauf et Grafcet

Le RdP sauf est proche de Grafcet, cependant il y a des différences importantes entre deux outils :

Evolutions simultanées **:** Dans le Grafcet, lorsqu'une étape valide deux ou plusieurs transition, il y a alors franchissement simultané de toutes les transitions. Dans un RdP sauf ce n'est pas possible, il y a un conflit. Donc on doit fait attention à ce problème. Ce problème sera discuté dans l'étape de transformation.

Réactivation **:** Dans le Grafcet, lorsqu'une étape active est activée à nouveau, elle reste active, mais dans le RdP le nombre de marques est augmenté. Dans un RdP sauf, ce problème n'existe pas.

Cependant il est possible qu'une place de contrôle synthétisée ait un marquage non binaire rendant le modèle global non sauf. Ce problème sera traité au cours de cette thèse.

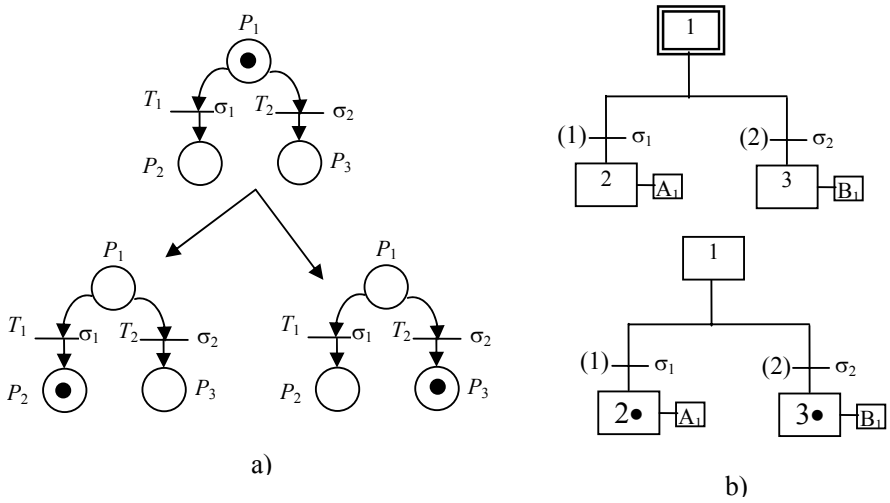


Fig. 1.12. Occurrence simultanée de deux événements non indépendants σ_1 et σ_2
a) Réseau de Petri b) Grafcet

1.7 Conclusion

Ce premier chapitre a présenté quelques notions essentielles sur les SED. Ils peuvent être modélisés par un langage, et si ce dernier est régulier, il est possible de construire un automate dont le langage sera précisément celui qui décrit le comportement du SED.

Les automates peuvent être utilisés dans la modélisation des SED. Lorsqu'il est décrit par plusieurs automates élémentaires l'opération de composition (synchrone ou asynchrone) est alors nécessaire pour modéliser le comportement global du Système. Les modèles SED dans le cas de systèmes réels conduisent souvent à des modèles de taille excessive du fait de l'explosion combinatoire du nombre d'états. Les réseaux de Petri (RdP) ont, par rapport aux automates, l'avantage d'être un modèle bénéficiant de structure plus riche, s'adaptant parfaitement à la description des SED. Par un passage systématique du RdP vers l'automate, il est possible de profiter totalement des propriétés des deux modèles.

Le Grafcet est bien adapté à la spécification de la commande logique. Il est plus proche de la mise en œuvre. C'est pour cela qu'une partie de notre travail sera consacrée à développer une approche pour passer du contrôleur RdP synthétisé vers le Grafcet.

Chapitre 2

Synthèse de superviseur pour les systèmes à Evénements Discrets

Dans ce chapitre nous présentons la théorie de la supervision des SED. La problématique de cette étude sera décrite. Les concepts fondamentaux de cette théorie ainsi que les principaux résultats concernant la synthèse de superviseurs seront présentés. Dans une première parti nous présenterons l'approche de Ramadge et Wonham. La synthèse de contrôleur basée sur les Réseaux de Petri sera ensuite développée.

Chapitre 2

Synthèse de superviseur pour les systèmes à Evénements Discrets

2.1 Introduction

La théorie de la supervision des systèmes à événement discrets a été initiée par les travaux de Ramadge et Wonham (R&W) (Ramadge et al. 1987a, Ramadge et al. 1989]. Dans cette approche des modèles logiques sont utilisés pour décrire le fonctionnement d'un procédé. Un procédé est considéré comme un SED qui évolue spontanément en générant des événements. Le schéma de supervision est présenté dans la figure 2.1.

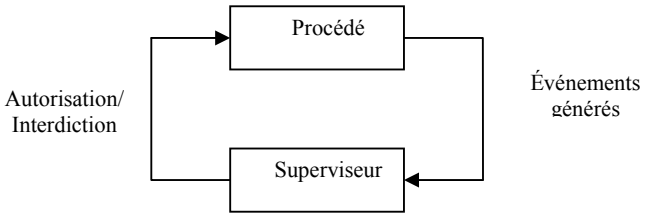


Fig. 2.1. Schéma de supervision

Dans ce schéma, un procédé est couplé à un superviseur. Les entrées du superviseur sont les sorties du procédé. Le rôle de superviseur est *d'autoriser ou d'interdire* l'occurrence d'événements dans le procédé. Par ce rôle, le superviseur peut changer le fonctionnement du procédé. Etant donnés un procédé et un ensemble de spécifications logiques de fonctionnement, l'objectif de la théorie R&W est de synthétiser un superviseur tel que le fonctionnement du procédé couplé au superviseur respecte les spécifications. De plus, on souhaite que le fonctionnement ainsi obtenu soit le plus permissif possible. La synthèse d'un tel superviseur est basée sur le concept de contrôlabilité.

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

La théorie de la supervision des SED fournit de nombreux concepts et résultats théoriques. Néanmoins, seuls les concepts et résultats fondamentaux nécessaires dans la suite de notre étude seront présentés. Ainsi, nous nous focaliserons sur le concept de contrôlabilité qui est le concept clef de la théorie R&W, ainsi que sur la synthèse de la supervision par automates puis par les réseaux de Petri.

2.2 Concept de supervision

2.2.1 Principe de la supervision

Dans cette section, nous supposerons que le système à contrôler (Procédé *P*) est modélisé par un automate *P*. Son comportement est donc donné par le langage *L*(*P*). Le comportement de *P* peut s'avérer ne pas être entièrement satisfaisant dans le sens où il ne respecte pas certaines propriétés appelées *objectifs de contrôle* (*Spécification*). Il est donc nécessaire de réduire ce comportement dans le but d'assurer ces objectifs. Cette restriction est réalisée par le biais d'un superviseur qui peut être vu comme une fonction qui à partir de l'histoire du système, va envoyer à celui-ci l'ensemble des événements qui doivent être interdits pour rester dans l'ensemble des comportements décrits par l'objectif. Contrôler un système consiste donc à lui ajouter des contraintes supplémentaires, induisant une réduction du comportement du système à un comportement souhaité.

Un procédé couplé à un superviseur peut être perçu comme un système qui reçoit en entrée une liste d'événement interdits. Dans la figure 2.2, le procédé *P* reçoit en entrée la liste d'événements interdits Φ et en sortie donne l'ensembles des événement possible dans chaque état moins l'ensemble des événement interdit dans cette état. L'ensemble des événements possible dans l'état *q* de l'automate correspondant est présenté par $\sum(q)$. Ce procédé sera appelé *procédé supervisé* (*S /P*).

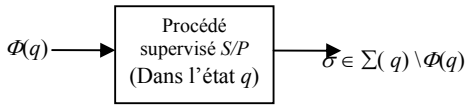


Fig. 2.2. Procédé supervisé

Depuis un état *q*, le procédé supervisé *S/P* évolue de façon spontanée en produisant un événement $\sigma \in \sum(q) \setminus \Phi$, où $\sigma \in \sum(q) \setminus \Phi$ dénote l'ensemble des événements qui appartiennent à l'ensemble $\sum(q)$ et qui n'appartiennent pas à l'ensemble Φ. Cela signifie que le procédé supervisé peut générer un événement σ si σ peut être généré par le procédé en isolation et si σ n'est pas interdit.

Nous pouvons remarquer qu'un procédé supervisé peut être défini de façon équivalente en spécifiant en entrée la liste des événements autorisés. Si $\sum$ est l'alphabet des événements du procédé alors, la liste des événements autorisés correspond à l'ensemble $\sum \setminus \Phi$. La liste des événements interdits ou autorisés est fournie par le superviseur en fonction de l'ensemble des événements de sortie du système. Cette idée est présentée dans la figure 2.3 où l'indice *i* représente le temps logique. Dans cette figure, le procédé supervisé *S/P* est supposé être dans un état *q*. Depuis cet état, *S/P* peut générer à l'instant logique *i*+1, l'événement σ^{i+1}. Cet événement est un élément de l'ensemble de $\sum(q^i) \setminus \Phi^i$. Fondamentalement, l'observation du procédé par le superviseur est asynchrone. L'occurrence de σ^{i+1} peut conduire le superviseur dans un nouvel état. Immédiatement, la liste d'événements interdits Φ^{i+1} est alors fournie au procédé et ainsi de suite. On appellera *fonctionnement en boucle fermée*, le fonctionnement du procédé couplé à son superviseur.

Remarque 2.1 : Le rôle du superviseur se cantonne à interdire l'occurrence d'événements dans le procédé. Il ne peut en aucun cas forcer des événement à se produire. Il s'ensuit donc que le superviseur ne peut que restreindre le fonctionnement du procédé.

❑

En pratique, certains événements générés par un procédé ne peuvent pas être interdits. En effet, si l'on considère l'exemple de la machine de la figure 1.1, il paraît naturel que la panne de la machine ne puisse pas être interdite. Un tel événement sera appelé *événement incontrôlable*. Au contraire, on appellera *événement contrôlable* tout événement qui peut être interdit à n'importe quel moment.

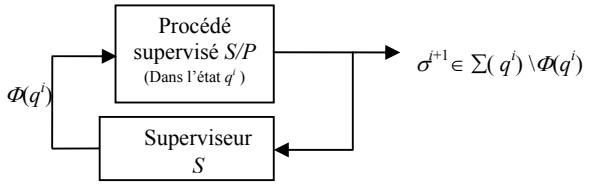


Fig. 2.3. Schéma de supervision avec indice de temps

De manière générale, si Σ est l'alphabet des événements d'un procédé, on peut définir la partition suivante : $\Sigma = \Sigma_c \cup \Sigma_u$, où Σ_c et Σ_u dénotent respectivement les ensembles d'événements contrôlables et incontrôlables sur Σ. Comme les événements incontrôlables ne peuvent pas être interdits par la supervision, il est nécessaire d'exiger qu'une liste Φ d'événements interdits ne contienne aucun événement incontrôlable. Ainsi, dans la figure 2.3 nous aurons pour tout i : $\Phi(q^i) \cap \Sigma_u = \varnothing$.

Hypothèse 2.1: Deux événements indépendants ne peuvent pas être simultanément générés par le procédé.

❑

L'hypothèse 2.1 est basée sur le fait que les procédés que nous considérons évoluent à des instants continus du temps. Comme un événement a une durée qui est nulle, la probabilité que 2 événements indépendants soient simultanément générés par le procédé est alors nulle. Dans la réalité cette hypothèse n'est pas toujours applicable. A cause du fonctionnement cyclique dans les systèmes de commande implantés, il est possible d'avoir des événements simultanés. Ce problème sera pris en compte lorsqu'il s'agira de transférer les résultats de notre travail du RdP vers le Grafcet.

2.2.2 Définition d'un superviseur

Conformément à la figure 2.3, un superviseur peut être perçu comme une machine à états dont les sorties sont des listes $\Phi(q^i)$ d'événements interdits. On peut remarquer qu'entre deux occurrences successives d'événements par exemple, σ^i et σ^{i+1}, la sortie $\Phi(q^i)$ du superviseur reste inchangée. Ainsi, on peut représenter un superviseur par un modèle tel que la sortie ne dépend que de l'état. Un superviseur peut donc être modélisé par une machine de Moore.

Définition 2.1 : Le superviseur S peut être défini par le 6-uplet $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ où V est un ensemble fini d'états; Σ est l'alphabet d'entrée; $\xi : V \times \Sigma \rightarrow V$ est la fonction de transition d'états; v_0 est l'état initial; $2^{\Sigma c}$ est l'alphabet de sortie; et $\theta : V \rightarrow 2^{\Sigma c}$ est la fonction d'affectation de sortie.

❑

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

2.2.3 Supervision d'un système manufacturier

Exemple 2.1 : Considérons un système manufacturier composé de deux machines identiques : M_1 et M_2, et un stock entre ces deux machines. Conformément à la figure 2.4, les deux machines travaillent de façon indépendante, puisent des pièces brutes en amont et rejettent des pièces usinées en aval.

Fig. 2.4. Un système manufacturier

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

Chaque machine de ce procédé peut être modélisée par un automate sans sorties (accepteur). Le graphe de transition d'états de l'automate des machines M_1 et M_2 est le même que celui de l'exemple qui est présenté au début du chapitre (figure 2.5).

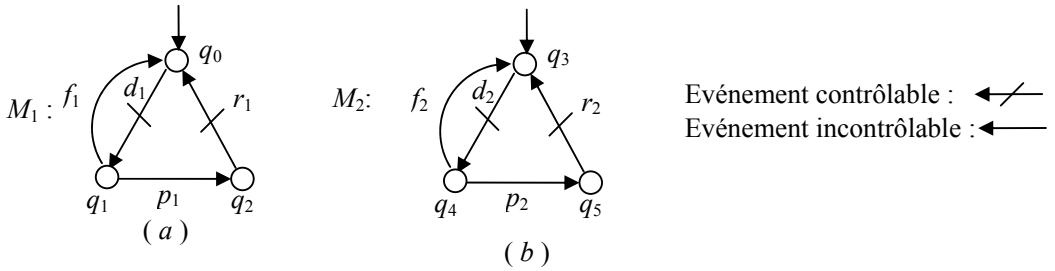


Fig. 2.5. Modèles des machine M_1 et M_2

Considérons la machine M_1 (figure 2.5.a). Dans son état initial (état q_0), la machine est à l'arrêt. L'événement d_1 modélise le début du cycle de la machine, l'occurrence de cet événement conduit la machine dans l'état de marche (état q_1). Dans notre exemple, nous supposons que d_1 est simultané avec la prise d'une pièce en amont. De même, la fin de cycle (événement f_1) est simultanée avec le dépôt d'une pièce en aval. Lorsque la machine M_1 est en marche (état q_1), l'occurrence de l'événement p_1 conduit la machine dans un état de panne (état q_2). Depuis cet état, la réparation de la machine, ramène la machine dans son état initial. La machine M_2 possède un fonctionnement similaire à M_1.

Notons respectivement Σ_1 et Σ_2, les alphabets des machines M_1 et M_2. Nous avons :

$$\Sigma_1 = \{d_1, f_1, p_1, r_1\} \quad \text{et} \quad \Sigma_2 = \{d_2, f_2, p_2, r_2\}$$

Le fonctionnement du système manufacturier est alors défini sur un alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$. Nous avons pris la convention de représenter par un arc barré, toute transition associée à un événement contrôlable. Ainsi, nous supposons que $\Sigma_c = \{d_1, d_2, r_1, r_2\}$ et $\Sigma_u = \{f_1, f_2, p_1, p_2\}$.

Un modèle automate sans sortie de notre système manufacturier peut être obtenu en effectuant le composé synchrone des modèles M_1 et M_2. Le modèle P défini par $P = M_1 \|_s M_2$ est représenté par son graphe de transition d'états dans la figure 2.6.

Dans l'automate P, un état est un couple (q_i, q_j), où q_i est un état de M_1 et q_j est un état de M_2. Cet état (q_i, q_j) est noté q_{ij} dans la figure 2.6.

Modèle de la Spécification : On considère la spécification de fonctionnement suivante pour notre système manufacturier. Le fonctionnement de notre système manufacturier doit respecter la présence d'un stock de capacité limitée à 1, situé entre les 2 machines. Nous supposons donc à présent que les machines travaillent en série. Conformément à la figure 2.7, une pièce doit visiter M_1 puis M_2. La présence du stock entre M_1 et M_2 impose que : (1) la machine M_2 ne peut commencer à travailler que si elle peut prendre une pièce dans le stock, c'est-à-dire, si le stock est plein, et (2) la machine M_1 ne peut déposer une pièce dans le stock que si celui-ci est vide. Le stock est supposé vide dans son état initial.

Le superviseur S de la figure 2.8 permet de garantir le respect de cette spécification de fonctionnement. Dans cet automate les états v_0 et v_1 correspondent respectivement aux états : "stock vide" et "stock plein".

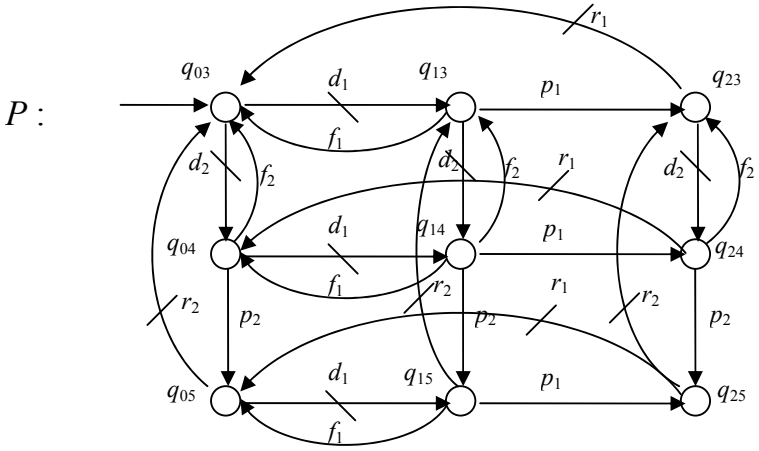


Fig. 2.6. Modèle synchrone de deux machines

Fig. 2.7. Le système manufacturier sous la contrainte de stock

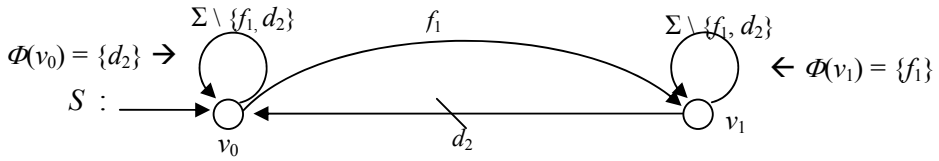


Fig. 2.8. Modèle automate de la spécification

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

Fonctionnement désiré en boucle fermée : On appelle *fonctionnement en boucle fermée*, le fonctionnement du procédé couplé à son superviseur. Conformément à la figure 2.3, un événement peut être généré par le procédé supervisé si, il peut être généré par le procédé P en isolation et s'il est autorisé (non interdit) par le superviseur S. Par extension, une séquence d'événements ω est possible dans le fonctionnement en boucle fermée, si elle est possible dans le procédé en isolation ($\omega \in L(P)$), et si elle est autorisée par le superviseur ($\omega \in L(S)$). Si on note S/P la machine constituée du procédé P couplé au superviseur S, le langage $L(S/P)$ représente alors le fonctionnement en boucle fermée du système. Le langage $L(S/P)$ est simplement défini par :

$$L(S/P) = L(P) \cap L(S).$$

Le modèle automate reconnaissant $L(S/P)$ est obtenu en effectuant le composé synchrone de P et de S.

Définition 2.2 : Le langage $L(S/P)$ généré par le procédé supervisé en boucle fermée est défini par :

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

Où $\sigma \in S(\omega)$ signifie que l'occurrence d'événement σ après le mot ω ne doit pas été interdit par le superviseur.

❑

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

Le modèle de fonctionnement désiré en boucle fermé du système supervisé est donné dan la figure 2.9.

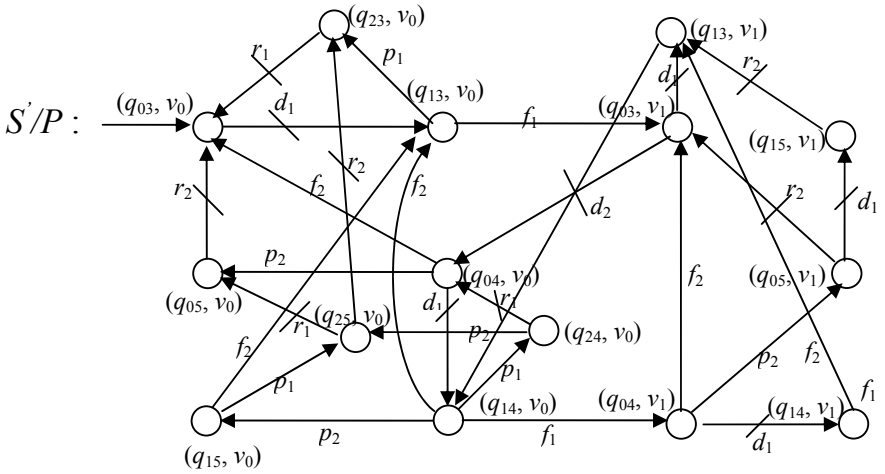


Fig. 2.9. Modèle de fonctionnement désiré en boucle fermée

2.3 Contrôlabilité

2.3.1 Concept de contrôlabilité

Ramadge et Wonham ont introduit la notion de contrôlabilité pour des SED afin de caractériser les langages supervisés d'un procédé (Ramadge et al. 1987b). Etant donnés un procédé P et une spécification de fonctionnement S_{Spec}, on souhaite synthétiser un superviseur S de façon à ce que le système en boucle fermé S/P, respecte la spécification. C'est-à-dire qu'on doit chercher le langage $L(P) \cap L(S_{spec})$. Ce langage appelé *fonctionnement désiré* correspond à l'ensemble des séquences qui peuvent être générées par le procédé et qui sont tolérées par la spécification, ce langage est noté L_D. Il n'est pas toujours possible (prise en compte d'événements incontrôlables Σ_u) de restreindre, par la supervision, le fonctionnement d'un procédé à n'importe quel sous - langage de ce fonctionnement. L'existence d'un superviseur S tel que $L(S/P) = L_D$ réside dans le concept de contrôlabilité.

Définition 2.3 : Un langage K est dit contrôlable par rapport à un langage *L(P)* si :

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

Où $\overline{K}$ représente le préfixe-clôture de langage de spécification et $L(P)$ le langage de procédé.

❑

On peut dire que le langage de spécification K est contrôlable par rapport à un langage $L(P)$ si pour toute chaîne ω de K et pour tout événement incontrôlable τ de Σ_u , la chaîne $\omega\tau$ appartient à $L(P)$, implique qu'elle appartient aussi à K.

La théorie de Ramadge et Wonham permet de résoudre ce problème. Nous allons présenter directement l'algorithme de Kumar (Kumar, 1991) qui permet de déterminer le langage contrôlable maximal permissif.

Pour présenter cet algorithme, il est d'abord nécessaire de définir quelques notions importantes.

Définition 2.4 : Soit $P = (Q, \Sigma, \delta, q_0)$ et $S_{spec} = (V, \Sigma, \xi, v_0)$ les modèles automates du procédé et de la spécification. Par composé synchrone des deux modèles, L'ensemble des *états interdits* Q_I sera défini comme ci-dessous :

$$\mathcal{M}_I = \{(q,v) \mid \exists\, \sigma \in \Sigma_u \text{ avec } \delta(q, \sigma) \text{ défini, et } \xi(v, \sigma) \text{ non défini } \}$$

❑

Remarque 2.2: L'ensemble des états interdits donné par la définition 2.4 est un type d'états interdits. Un autre type d'états interdits correspond aux états de blocage. Pour notre approche cette différence n'est pas importante.

❑

Il y a aussi autre type d'état interdit qu'il faut ajouter à l'ensemble défini ci-dessus : ce sont les états *faiblement interdits* :

Définition 2.5 : Soit $\mathcal{M}_{PS}$ l'ensemble des états possibles et autorisés par la spécification et $\mathcal{M}_I$ l'ensemble des états interdits. L'ensemble des *états faiblement interdits* $\mathcal{M}_F$ sera défini comme ci-dessous :

$$\mathcal{M}_F = \{q_i \mid q_i \in \mathcal{M}_{PS},\ q_j \in \mathcal{M}_I \text{ ou } q_j \in \mathcal{M}_F \text{ et } \sigma \in \Sigma_u\ q_i \xrightarrow{\sigma} q_j\ \}$$

❑

A partir des ensembles $\mathcal{M}_F$ et $\mathcal{M}_I$, on peut construire deux autres ensembles d'états qui nous seront très utiles dans la suite de ce mémoire: 1) l'ensemble des *états interdits frontière* et, 2) l'ensemble des *états autorisés critiques* .

L'ensembles des états interdits frontières correspond aux états interdits ou faiblement interdits atteignables par occurrence d'événements contrôlables. Formellement, cet ensemble est défini ci-dessous.

Soit $\mathcal{M}_A$ l'ensemble des états autorisés par le système supervisé, $\mathcal{M}_I$ l'ensemble des états interdits et $\mathcal{M}_F$ faiblement interdits

Définition 2.6 :.L'ensemble des *états interdits frontière* $\mathcal{M}_{IF}$ est l'ensemble

$$\mathcal{M}_{IF} = \{q_j \mid q_i \in \mathcal{M}_A,\ q_j \in \mathcal{M}_I \text{ ou } q_j \in \mathcal{M}_F \text{ et } \sigma \in \Sigma_C\ q_i \xrightarrow{\sigma} q_j\ \}$$

❑

L'ensemble des états autorisés critiques correspond aux états à partir desquels l'occurrence d'événements contrôlables mène à un état frontière.

Définition 2.7 : L'ensemble des *états autorisés critiques* $\mathcal{M}_{AC}$ sera défini comme ci-dessous :

$$\mathcal{M}_{AC} = \{q_i \mid q_i \in \mathcal{M}_A,\ q_j \in \mathcal{M}_{IF} \text{ et } \sigma \in \Sigma_C\ q_i \xrightarrow{\sigma} q_j\}$$

❑

Tous ces ensembles d'états sont présentés dans la figure 2.10.

Les deux groupes constitués par l'ensemble des états interdits frontières et l'ensemble des états autorisés critiques sont essentiels dans notre approche. Par l'interdiction de franchissement des événements contrôlables d'états autorisés critiques vers les états interdits frontières, on empêche l'atteignabilité de tous les états interdits.

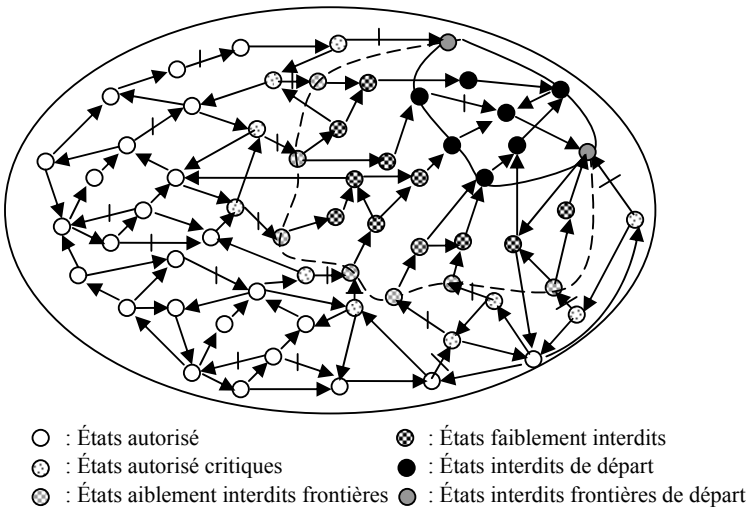


Fig. 2.10. Les ensembles des états possibles

2.3.2. Algorithme de Kumar

A partir des modèles automates P d'un procédé et S_{spec} d'une spécification de fonctionnement, l'algorithme de Kumar permet de vérifier la contrôlabilité du langage de spécification $L(S_{spec})$. De plus, dans le cas où le langage $L(S_{spec})$ n'est pas contrôlable, cet algorithme permet de synthétiser un modèle automate du langage suprême contrôlable du fonctionnement désiré $supC(L_D)$ (figure 2.11).

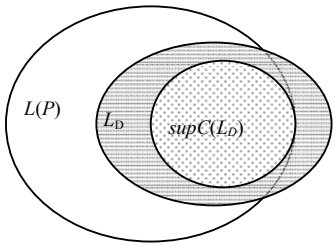


Fig. 2.11. Langage suprême contrôlable d'un fonctionnement désiré

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

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

Pas 2. On détermine l'ensemble des *états interdits*.

Pas 3. On détermine l'ensemble des *états faiblement interdits*.

Pas 4. On supprime de D l'ensemble des états interdits ainsi que l'ensemble des états faiblement interdits (ainsi que les transitions associées à ces états). On supprime de D l'ensemble des états non accessibles, c'est-à-dire, tout état (q, v) tel qu'il n'existe pas de chemin permettant de rejoindre (q, v) depuis l'état initial.

Appliquons cet algorithme sur notre exemple. La figure 2.12 donne l'automate final et l'ensemble des états interdits.

Par application de l'algorithme de Kumar pour notre exemple, nous trouvons les états interdits suivants : $\{(q_{13},v_1), (q_{14},v_1), (q_{15},v_1)\}$. Dans cet exemple il n'y a pas d'états faiblement interdits.

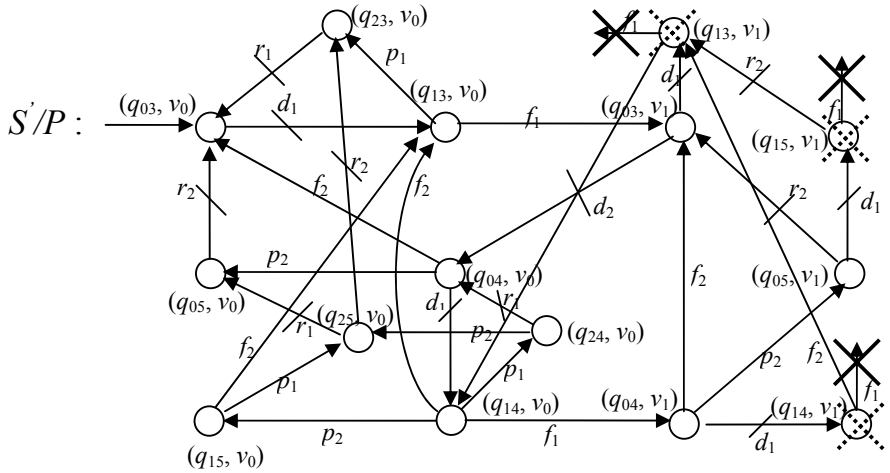


Fig. 2.12. Modèle automate du système supervisé avec des états interdits

Le modèle final de cet automate est présenté dans la figure 2.13.

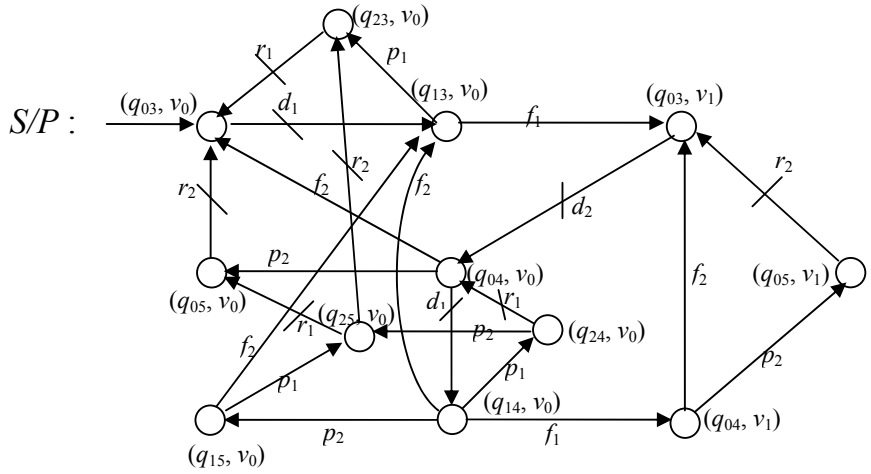


Fig. 2.13. Modèle final de système supervisé sans états interdits

2.3.3 Remarques et conclusion sur l'approche RW

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

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

La théorie de R&W constitue actuellement une activité importante au sein de plusieurs groupes de recherche sur le plan international, cette théorie possède diverses extensions. Les travaux basés sur cette théorie sont très nombreux. L'approche classique utilise des automates à états et un point de vue

centralisé. Les modèles obtenus sont de grande taille, ce qui pose un problème de calculabilité et de lisibilité. Différents travaux proposent des approches permettant de réduire cette taille par la remise en cause du point de vue centralisé : point de vue hiérarchique, modulaire sous observation partielle ou décentralisé (Lin et al. 1990) (Gaudin, 2004) (Takai, 2005). Une fois la commande obtenue, une implantation est nécessaire. Là aussi, différents travaux proposent des interprétations sous forme de schémas *Ladder Diagram* ou de *Grafcet* (Giua et al., 1993)(Uzam et al., 1998)(Charbonnier, 1996).

La théorie RW a été aussi étendue afin de tenir compte des contraintes temporelles (Gaubert, 1995) (Brandin et al. 1994)(Sava, 2002). Parmi les autres extensions de la théorie RW, nous citons l'extension aux systèmes hybrides (Uzam et al., 2006), et l'extension aux SED à structure vectorielle (Li et al., 1994).

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

2.4 Synthèse de la commande basée sur les Réseaux de Petri

L'inconvénient majeur de la synthèse de la commande basée sur les automates est l'explosion combinatoire du nombre d'états quand il s'agit d'étudier des systèmes de taille réelle. Les RdP permettent de pallier à cet inconvénient. De plus, nous sommes intéressés par des outils de modélisation facilement implantables dans un automate programmable industriel (API). Le Grafcet est un outil dérivé des RdP, nous allons étudier les méthodes de synthèse de la commande basées sur les RdP saufs que nous adapterons ensuite au Grafcet. Le caractère sauf du RdP sauf permettra une implantation aisée dans les API.

La modélisation d'un système ou des spécifications par RdP est en général plus facile que par un automate. Le modèle RdP de l'exemple 2.1 est présenté dans la figure 2.14.

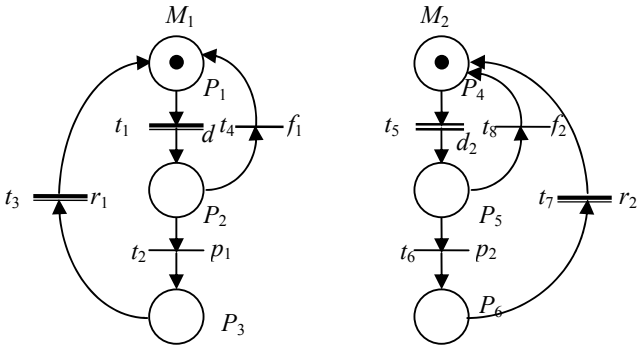


Fig. 2.14. Modèle RdP de la figure 2.4

Les transitions contrôlables sont représentées par un trait double.

Le problème de contrôlabilité apparaît dans les systèmes modélisés par RdP lorsqu'on veut synchroniser le modèle des spécifications avec le modèle du procède par des événements incontrôlables.

Prenons la spécification présentée dans la figure 2.8. Lors de synchronisation entre le modèle du procédé et le modèle de la spécification il y aura des états interdits à cause d'existence de l'événement incontrôlable f_1. Les états interdits se produisent lorsque toutes les places en amont d'une transition contrôlable qui appartiennent au procédé sont marquées et les places qui appartiennent à la spécification, ne sont pas marquées. Le modèle de fonctionnement désiré en boucle fermée par RdP est présenté dans la figure 2.15.

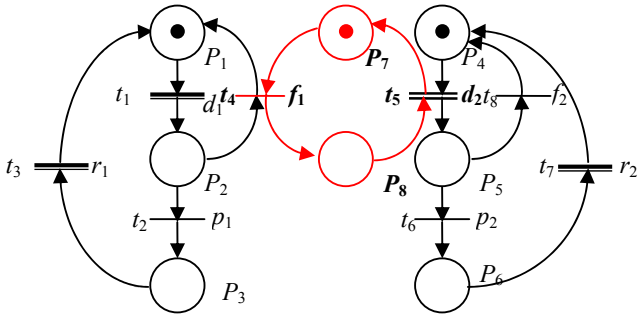


Fig. 2.15. Modèle en boucle fermée RdP pour exemple 2.1

Appelons R_P le modèle RdP du procédé, R_S le modèle de la spécification, R_{PS} le fonctionnement désiré en boucle fermée, et T_u l'ensemble des transitions incontrôlables communes aux deux modèles.

Soit M_P l'ensemble des marquages des places du procédé et M_S celui des spécifications. Les états interdits peuvent être déterminés en appliquant la définition ci-dessous :

Définition 2.8: L'ensemble des états interdits dans le modèle du système en boucle fermée R_{PS} est donné par l'expression :

$$\mathcal{M}_I = \{[M_P, M_S] \mid \exists\, t \in T_u, \forall\, P_i \in {\bullet t} \;\&\; P_i \in R_P, \mathrm{m}(P_i)=1 \text{ et } \exists\, P_j \in {\bullet t} \;\&\; P_j \in R_S, \mathrm{m}(P_j)=0\}$$

Où T_u est l'ensemble des transitions auxquelles sont associés des événements incontrôlables et $\bullet t$ présente l'ensemble des places en amont de la transition t.

❑

A partir du RdP, on peut construire le graphe de marquage que nous considérons comme un automate. Dans la section précédente, nous avons vu comment on peut calculer l'ensemble des états faiblement interdits et l'ensemble des états interdits frontière à partir d'un automate. Nous allons appliquer cette technique sur l'automate déduit du RdP. A cause de l'importance de la notion d'états interdits nous l'expliquons par un exemple.

Exemple 2.2 : On considère un système composé de deux machines et d'un robot (Figure 2.16). Chaque machine opère sur une seule pièce brute à la fois. Lorsque la machine ait fini son travail (événement incontrôlable ftm_i), le robot décharge la machine et lorsqu'il ait finit le déchargement de la machine (événement incontrôlable fdm_i), il transfère la pièce vers un stock. Après la fin du transfert (événement incontrôlable ftr_i), le robot revient à son état initial. Seul le début de tâche sur chaque machine (événement c_1 et c_2) est contrôlable. Les spécifications sont imposées par le robot.

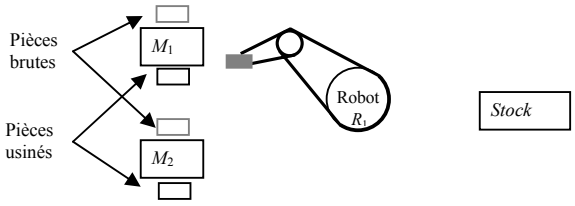
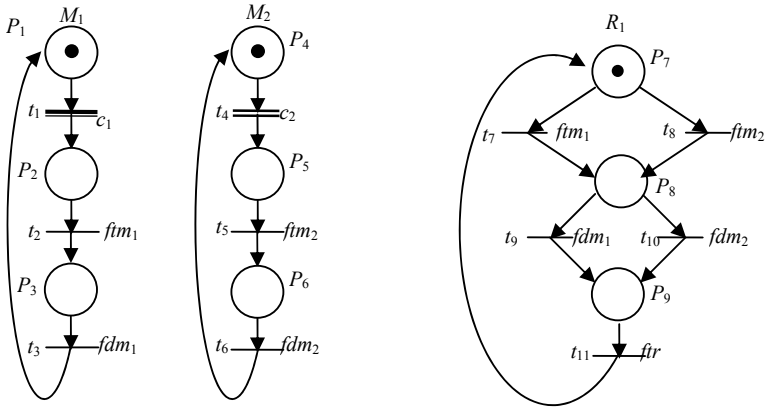


Fig. 2.16. Système manufacturier

Le modèle RdP du procédé et la spécification sont présentés dans la figure 2.17.



a) Modèle RdP des deux machines

b) Modèle RdP de fonctionnement de robot comme une spécification

Fig. 2.17. Modèle RdP d'un système manufacturier

La composition synchrone entre les deux modèles RdP nous donne le modèle RdP en boucle fermée (Figure 2.18).

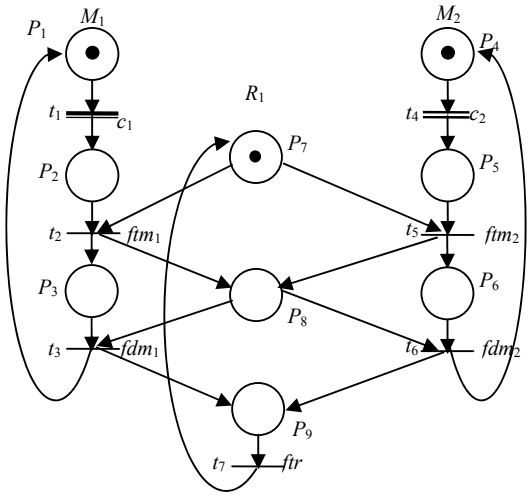


Fig. 2.18. Modèle RdP en boucle fermée

En raison de la synchronisation avec des événements incontrôlable ftm_1 et ftm_2 le procédé, ici ne peut pas respecter les spécifications, et donc il y a des états interdits. Nous allons donner une explication intuitive de l'état interdit. Supposons que le Robot soit entrain de décharger la machine M_1, et que la machine M_2 ait déjà démarré et soit arrivé à la fin de son travail. Dans cet état il est nécessaire que tout de suite le robot vienne pour prendre la pièce. Comme ce n'est pas le cas, la pièce est perdue. C'est donc un état qui ne doit jamais être atteint, c'est un état interdit. Par l'algorithme de Kumar (Kumar et al. 1996) on peut calculer formellement l'ensemble des états interdits et ensuite l'ensemble des états interdits frontière. Le graphe de marquage de ce système est présenté dans la figure 2.19.

$\mathcal{M}_{IF} = \{M_7, M_8, M_9, M_{10}, M_{11}\} = \{P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

La synthèse du contrôleur optimal consiste à supprimer l'ensemble des états interdits frontières de l'ensemble des marquages accessibles. Cela fait qu'il est quasiment impossible de revenir au RdP de départ.

Pour pallier ces inconvénients, plusieurs méthodes de synthèse de commande basées sur les RdP ont été proposées. Dans de nombreux travaux, les spécifications dans la synthèse de commande basée sur

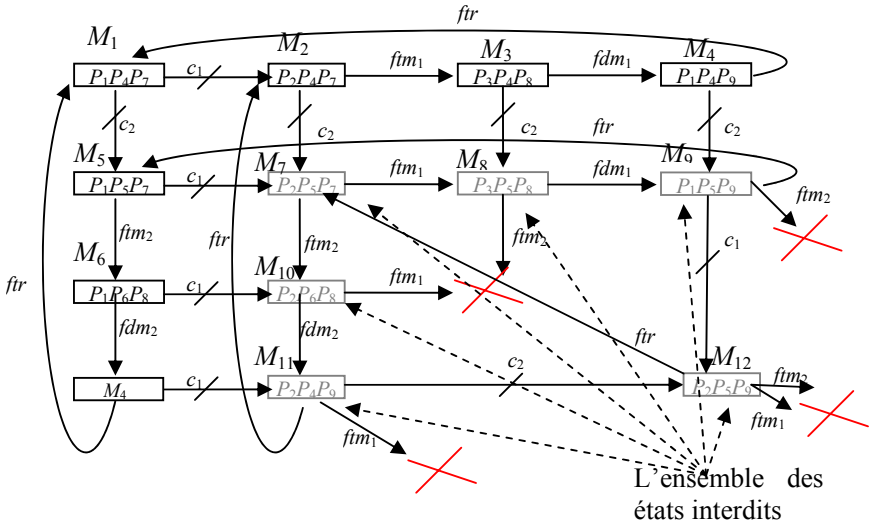


Fig. 2.19. Graphe de marquage accessible

les RdP, sont souvent données comme un ensemble d'états interdits en présence d'événements incontrôlables, ce type de problème a été étudié par plusieurs chercheurs.

Nous allons présenter ci-dessous les principales approches de synthèse de contrôleurs basées sur les RdP. Nous discuterons des avantages et inconvénients de chacune d'elles, cela nous permettra d'introduire notre travail de recherche. Dans les approches rencontrées dans la littérature les mots *contrôleur* et *superviseur* sont utilisés indifféremment. Dans la dernière partie de ce chapitre, nous discuterons de la signification exacte de chacun des deux termes pour bien situer notre contribution. Nous utiliserons l'exemple présenté dans cette section, dans le chapitre suivant.

2.4.1 Spécifications et contraintes linéaires

Dans de nombreux cas, les spécifications peuvent être représentées par des contraintes linéaires. L'ensemble des marquages admissibles est exprimé par un ensemble de contraintes linéaires (Contraintes Généralisées d'Exclusion Mutuelles CGEM). Certaines approches utilisent ce type de spécification pour construire un contrôleur. Cette méthode a été proposée par Giua (Giua et al. 1992) puis formalisée par Yamalidou et Moody (Yamalidou et al. 1996). Il s'agit d'une nouvelle approche pour construire un contrôleur en boucle fermée d'un système modélisé par un RdP. Le contrôleur se compose de places et d'arcs calculés à l'aide du concept d'invariants de marquage. Dans cette approche, les spécifications sont données par des contraintes linéaires entre les marquages des places du RdP.

L'idée consiste à construire un invariant de marquage pour chaque contrainte, de telle manière que l'ensemble des marquages atteignables soit réduit aux seuls marquages vérifiant la contrainte. Le contrôleur correspond à des places qui sont reliées aux transitions du RdP de procédé, de telle façon qu'il est garanti que le système n'atteindra pas un état interdit. Le procédé combiné du RdP du procédé et du contrôleur, possède des invariants de marquage nécessaire pour assurer que l'ensemble de contraintes ne sera pas violé. Dans le cas général, le contrôleur n'est pas nécessairement optimal, mais il est calculé facilement et sa taille dépend du nombre des contraintes. En outre, il peut être utilisé comme une très bonne estimation préliminaire pour les autres méthodes qui sont capables de calculer un contrôleur optimal d'un RdP. Sa dimension étant relativement petite, il simplifiera le calcul nécessaire pour atteindre le contrôleur optimal. Cette approche va être détaillée parce qu'elle sera utilisée dans notre travail de recherche. Un rappel concernant les contraintes linéaires est d'abord présenté, ensuit la synthèse du contrôleur par l'idée d'invariants de marquage sera étudiée. Nous insisterons particulièrement sur le problème de l'optimalité de la solution obtenue.

2.4.1.1 Contraintes Généralisées d'Exclusion Mutuelles

Une CGEM est une condition qui limite la somme pondérée de marques dans un ensemble des places. Guia et al. ont discuté la puissance de modélisation de cette sorte de contrainte. Ils ont prouvé que pour seulement une classe limitée de systèmes, un état interdit peut être exprimé comme une exclusion mutuelle. Il est aussi possible qu'une contrainte soit couverte par une autre contrainte. Mais ils n'ont pas proposé une méthode systématique pour la réduction de nombre de contraintes. Nous présentons brièvement ci-dessous l'approche de Giua (Giua et al. 1992).

Définition 2.9 : Soit $\langle N, M_0 \rangle$ un RdP avec l'ensemble de places P. Un CGEM $(\vec{l}, k)$ définit un ensemble des marquages autorisés :

$$\mathcal{M}(\vec{l}, k) = \{ M \in N^{|P|} \mid \vec{l}.M \leq k \}$$

Ou $\vec{l} : P \rightarrow N$ est vecteur de poids, et $n \in N^+$.

Le support de $\vec{l}$ est l'ensemble : $Q_w = \{ p \in P \mid l(p) > 0 \}$

❑

L'ensemble des CGEM (L, $\vec{k}$) avec $L = \left[\vec{l_1}, \vec{l_2}, ..., \vec{l_m} \right]$ et $\vec{k} = (k_1, \ldots, k_m)^T$ définie l'ensemble des marquages autorisés :

$$\mathcal{M}(L, \vec{k}) = \bigcap_{i=1}^{m} \mathcal{M}(\vec{l}_i, k_i) = \{ M \in N^{|P|} \mid L^T.M \leq \vec{k} \}$$

Il arrive souvent que des contraintes soient redondantes. Lorsque l'ensembles des états accessibles du RdP $\langle N, M_0 \rangle$ est égal à $\mathcal{M}(\vec{l}, k)$, la contrainte $\mathcal{M}(\vec{l}, k)$ est redondante vis-à-vis de $\langle N, M_0 \rangle$. On peut ainsi trouver deux ensembles des contraintes qui sont équivalentes.

Définition 2.10 : Deux l'ensembles CGEM $(L_1, \vec{k}_1)$ et $(L_2, \vec{k}_2)$ sont équivalent vis–à–vis de $\langle N, M_0 \rangle$ si $R(N, M_0) \cap \mathcal{M}(L_1, \vec{k_1}) = R(N, M_0) \cap \mathcal{M}(L_2, \vec{k_2})$.

❑

2.4.1.2 Des états interdits vers les contraintes linéaires

Jusqu'ici, nous avons vu qu'il est possible de présenter l'ensemble des états autorisés par des contraintes linéaires. Mais est-il possible d'interdire l'ensemble des état interdits $\mathcal{M}_{IF}$ par des contraintes linéaires ou peut-on trouver un ensemble des contraintes CGEM (L, $\vec{k}$) tel que :

$$R(N, M_0) \setminus \mathcal{M}_{IF} = R(N, M_0) \cap \mathcal{M}(L, \vec{k})$$

Guia a montré que dans le cas général ce n'est pas toujours possible de décrire l'ensemble des états interdits par des contraintes linéaires. Mais dans le cas des RdP saufs et conservatifs, cela est possible.

Théorème 2.1 Soit $\langle N, M_0 \rangle$ un RdP sauf et conservatif et soit $\mathcal{M}_{IF}$ un ensemble des états interdits. Il y a un ensemble des contraintes CGEM (L, $\vec{k}$) tel que :

$$R(N, M_0) \setminus \mathcal{M}_{IF} = R(N, M_0) \cap \mathcal{M}(L, \vec{k})$$

❑

On peut trouver la démonstration formelle de ce théorème dans (Giua et al. 1992). Nous rappelons que le nombre entier k pour chaque contrainte est égal au nombre de places marquées moins un. Nous présentons intuitivement cette théorie par un exemple. Considérons la figure 2.14 (exemple de deux machines). Supposons que la spécification impose que les deux machines ne peuvent pas être en panne simultanément. Par cette spécification et à partir du graphe de marquage, nous arrivons à l'ensemble des états interdits frontières :

$$\mathcal{M}_{IF} = \{P_2P_5, P_2P_6, P_3P_5\}$$

Le RdP de cet exemple est conservatif. Si l'on considère le premier état interdit, la seule possibilité pour que les places P_2 et P_5 soient marquées en même temps est dans l'état P_2P_5. Cet état est facilement séparable par la contrainte $m_2 + m_5 \leq 1$, où m_2 et m_5 présentent les nombres des marques dans les places P_2 et P_5. De la même manière nous pouvons interdire chacun des états interdits par une contrainte linéaire.

$P_2\ P_5$ interdit $\Leftrightarrow m_2+m_5 \leq 1$

$P_2\ P_6$ interdit $\Leftrightarrow m_2+m_6 \leq 1$

$P_3\ P_5$ interdit $\Leftrightarrow m_3+m_5 \leq 1$

Nous établissons ci-dessous cette propriété dans le cas général :

Définition 2.11 : Soit M_1 un marquage interdit de l'ensemble $\mathcal{M}_{IF}$ d'un RdP sauf et conservatif et soit M un marquage quelconque. Ce marquage peut être interdit par la contrainte équivalente suivante :

$M_1^T . M \leq$ Card [Support (M_1)] - 1

Où Card [Support (M_1)] est le nombre des places marquées dans l'état M_1.

❑

2.4.1.3 Cas des RdP non conservatifs

Lorsque les réseaux ne sont pas conservatifs, on peut avoir un problème. Ce problème se présente quand un état est couvert par un autre état comme l'illustre l'exemple ci-dessous (Remarque 1.2).

Reprenons l'exemple 2.1 et supposons que l'on impose qu'il n'y ait qu'une seule panne possible pour les deux machines. Ce type de comportement est rarement rencontré dans les cas réels mais nous l'utilisons pour mettre en évidence un contre-exemple. Le modèle en boucle fermée dans ce cas est donné dans la figure 2.20.

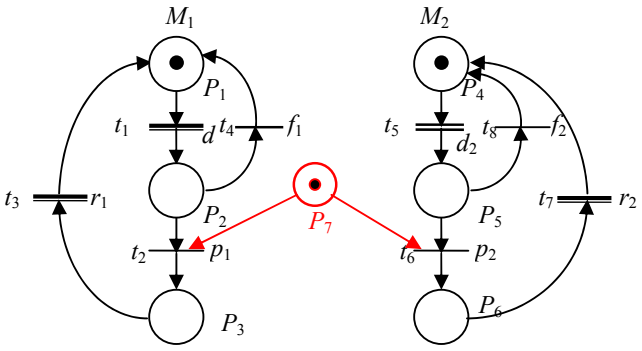


Fig. 2.20. Modèle RdP en boucle fermée

L'ensemble des états interdits $\mathcal{M}_{IF}$ et l'ensemble des états autorisés $\mathcal{M}_A$ sont les suivants :

$\mathcal{M}_{IF} = \{P_2P_5, P_2P_4, P_1P_5, P_2P_5P_7\}$

$\mathcal{M}_A = \{P_1P_4P_7, P_2P_4P_7, P_1P_5P_7, P_1P_6, P_3P_4\}$

Le problème principal, c'est que dans l'ensemble des états interdits, il y a des états pour qui les places marquées existent dans un état autorisé. Par exemple $\{P_2P_4\} \subset \{P_2P_4P_7\}$. Dans ce cas, il n'est pas possible de trouver une contrainte équivalente par le théorème 2.1. En effet par la contrainte $m_2+m_4 \leq 1$ l'état autorisé $P_2P_4P_7$ sera interdit. La seule façon de résoudre ce type de problème est de revenir à la modélisation du procédé ou de la spécification. Pour notre exemple, l'ajout de la place P_8 apporte une solution comme le montre la figure 2.21. Il n'y a pas pour l'instant de règle générale pour résoudre ce type de problème.

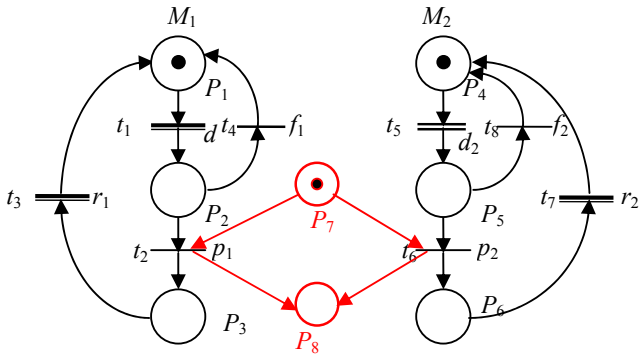


Fig. 2.21. Modèle RdP modifié

2.4.1.4 Simplification des contraintes linéaires

Giua (Giua et al. 1992) et aussi Kattan (Kattan 2004) ont montré qu'il est possible de simplifier les contraintes. Mais ils ne présentent pas de méthode systématique pour arriver toujours au résultat le plus simple possible. Dans l'exemple présenté dans la figure 2.20, il y a deux états interdits P_2P_5, $P_2P_5P_7$. Si l'état P_2P_5 a été interdit par la contrainte $m_2+m_5 \leq 1$, automatiquement l'état $P_2P_5P_7$ sera interdit et donc la contrainte $m_2+m_5+m_7 \leq 2$ sera redondante. Cette simplification est faite de manière manuelle. On comprend tout l'intérêt d'une telle technique. En diminuant le nombre de contraintes, on simplifie considérablement le calcul du superviseur. Notre contribution va consister à présenter deux méthodes systématiques pour la simplification des contraintes linéaires déduites de l'ensemble des états interdits. La première est applicable sur le RdP sauf et conservatif et la deuxième sur le RdP sauf général. Celles-ci seront présentées dans les chapitres 3 et 4.

2.4.1.5 Synthèse de contrôleur par la méthode des invariants

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

Le système qui doit être contrôlé est modélisé par un RdP, ce réseau s'appelle *modèle du procédé*, la matrice d'incidence du réseau du procédé est W_R. Il est possible que le comportement dynamique du procédé viole certaines contraintes imposées. Dans ce cas, un système de commande doit être synthétisé. Le contrôleur fonctionne en parallèle avec le procédé, son rôle est de surveiller le procédé et de prendre des actions correctives pour éliminer tout comportement indésirable.

Soit W_C la matrice d'incidence du RdP correspondant au contrôleur. Ce RdP est formé des transitions du modèle du procédé et d'un ensemble séparé des places. Le RdP du système commandé est formé du RdP du procédé et celui du superviseur.

Pour l'exemple présenté dans la figure 2.13, considérons une spécification qui impose que lorsque la machine M_1 est en marche la machine M_2 ne doit pas être en panne. On peut présenter cette spécification par la contrainte ci-dessous :

$$M(P_2) + M(P_6) \leq 1$$

Cela signifie que les places P_2 et P_6 ne peuvent pas être marquées en même temps.

Maintenant comment peut-on construire un contrôleur qui garantit cette contrainte ? Yamalidou et al. (Yamalidou et al. 1996) ont présenté une méthode qui consiste à ajouter une place de contrôle.

L'objectif de superviseur est de forcer le système à respecter des contraintes de type $(\vec{l}, k)$. On peut représenter ce type de contraintes de la manière suivante :

$$\sum_{i=1}^{n} l_i . M(P_i) \leq k \qquad (2\text{-}1)$$

où : $M(P_i)$ est le marquage de la place P_i, l_i et k sont les nombres entiers. Le nombre l_i représente le poids du marquage d'une place dans la contrainte et k la borne.

Par exemple pour la contrainte $M(P_2) + M(P_6) \leq 1$:

$k = 1$ et $l_1 = 0, l_2 = 1, l_3 = 0, l_4 = 0, l_5 = 0, l_6 = 1$.

Chaque inégalité de type (2-1) peut-être transformée en une égalité en ajoutant une variable positive et entière $M(P_c)$, la contrainte devient : $M(P_2) + M(P_6) + M(P_c) = 1$,

Et en général :

$$\sum_{i=1}^{n} l_i . M(P_i) + M(P_c) = k \qquad (2\text{-}2)$$

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

La structure du réseau de contrôleur sera calculée en observant que l'introduction de la variable $M(P_c)$ introduit un invariant de marquage dans le système contrôlé défini par (2-2).

Dans la matrice d'incidence W associée à un RdP correspond à sa structure (indépendante du marquage), une colonne correspond à la modification du marquage apportée par le franchissement de la transition correspondante. Soit $\overline{S}$ le vecteur caractéristique d'une séquence S qui mène de M_j à M_k. Le marquage atteint M_k est donné par l'équation fondamentale :

$$M_k = M_j + W . \overline{S} \qquad (2\text{-}3)$$

Tout P-semi-flot X, tel que $X^T = [l_1, \ldots, l_i, \ldots, l_n, 1] = [L^T, 1]$, où $l_i \in \mathrm{IN}$, , permet d'avoir l'invariant de marquage donné par la relation suivante :

$$X^T.M_k = X^T.M_j = X^T.M_0 \qquad (2\text{-}4)$$

En multipliant l'équation (2-3) par X^T et en utilisant la relation (2-4) nous avons :

$$X^T.M_k = X^T..M_j + X^T.W.\ \overline{S} \Rightarrow\ 0 = X^T.W.\ \overline{S} \quad \Rightarrow\ X^T.W = 0 \qquad (2\text{-}5)$$

Soit W_{RC} la matrice d'incidence du RdP correspondant au système contrôlé. Chaque place du contrôleur va ajouter une ligne à la matrice d'incidence du système (procédé et spécifications) W_R. La matrice W_{RC} du système contrôlé est composée de deux matrices, la matrice originelle du modèle W_R et

la matrice d'incidence du contrôleur W_c. A partir de relation (2-2), la matrice L est construite[1]. Elle permet de calculer de manière algébrique la matrice d'incidence du contrôleur ainsi que c'est rappelé ci-dessous. C'est ce calcul extrêmement simple qui a rendu cette approche très populaire.

$$W_C = - L^T.W_R \qquad (2\text{-}6)$$

$$M_{c\_ini} = k - L^T.M_{R\_ini}$$

Où M_{c_ini} est le marquage initial des places de contrôle et M_{R_ini} est le marquage initial du procédé.

Prenons l'exemple de la figure 2.14, la matrice d'incidence du procédé W_R est :

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

A partir de la contrainte (2-6), nous pouvons calculer la partie W_C de la matrice incidence du système :

$M(P_2) + M(P_6) \leq 1 \Rightarrow L^T = [\,0\,,\,1\,,\,0\,,\,0,\,0\,,\,1] \Rightarrow W_C = [\,-1\,,\,1\,,\,0\,,\,1\,,\,0,\,-1,\,1,\,0\,]$

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

Le marquage initial de la place de contrôle est :

$M_{C\_ini} = 1 - 0 = 1$

Le modèle du RdP final contrôlé est présenté dans la figure 2.22.

Le contrôleur fonctionne en parallèle avec le procédé. Son rôle est de surveiller le procédé et de prendre des actions correctives pour éliminer tout comportement indésirable. Si une place de contrôle n'est pas marquée, alors la transition correspondante est bloquée à condition que l'événement qui lui est associé soit contrôlable. Car le contrôleur n'est pas capable d'empêcher des événements incontrôlables de se produire. Ce problème est étudié dans la section suivante.

2.4.1.6 Problème des transitions incontrôlables

Soit W_u, la sous-matrice d'incidence représentant la partie incontrôlable, elle contient les colonnes de W_R qui correspondent aux transitions incontrôlables. Le contrôleur peut violer les contraintes si W_{CU} contient au moins un élément strictement positif, c'est-à-dire, s'il y a des arcs partant de la place

[1] - Pour une seule contrainte nous avons une matrice avec une seule ligne

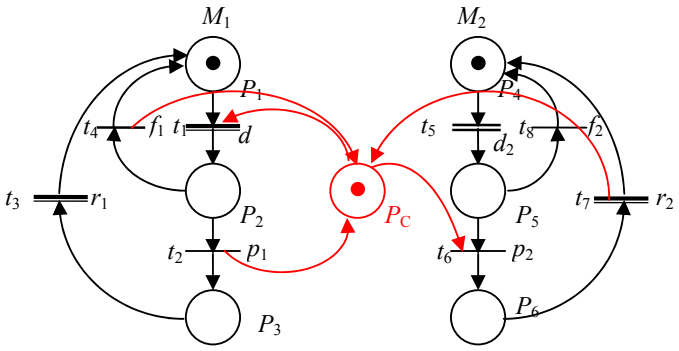


Fig. 2.22. Modèle final RdP avec spécification

de contrôle vers une transition incontrôlable. La matrice d'incidence W_{RC} et les différentes sous-matrices déduites sont présentées dans la figure 2.23.

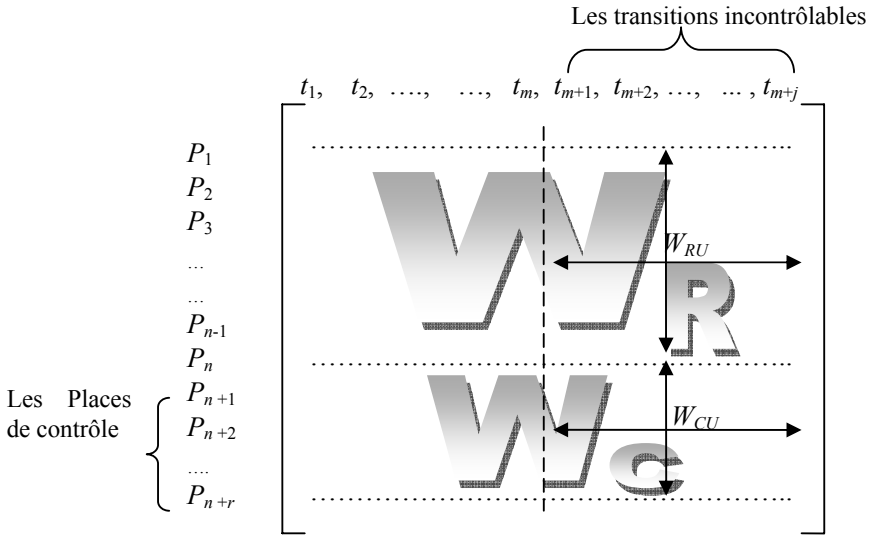


Fig. 2.23. Matrice d'incidence W_{RC} et les ensembles de sous- matrices

Si W_{CU} contient des valeurs positives, les contraintes ne sont pas satisfaites. Pour résoudre ce problème, une méthode intuitive consiste à remonter les branches jusqu'à trouver une transition contrôlable qui soit en aval de la place de contrôle. Cette idée est présentée dans (Yamalidou et al. 1996). Mais cette méthode n'est pas toujours applicable. Moody dans (Moody et al. 2000) a présenté une méthode pour résoudre ce problème. Dans cette méthode les auteurs modifient les contraintes L pour que W_{CU} ne contienne pas des valeurs positives. Son principal inconvénient est qu'elle ne donne pas en général la solution optimale.

2.4.2 Synthèse de contrôleur par retour d'état

Le contrôle des RdP qui suit, est celui qui est présenté par E. Holloway et par Bruce H. Krogh dans (Holloway et al. 1990). Les auteurs ont utilisé le RdP pour contrôler les systèmes qui peuvent être modélisés par des graphes d'évènements cycliques et saufs. Ils constituent une classe particulière de SED (pas de possibilités de conflit ou de choix, ni de cumul). Ils ont montré comment synthétiser un contrôleur maximal permissif en boucle fermée qui garantit qu'aucun des états interdits ne sera atteint. Leur méthode ne nécessite pas une recherche exhaustive de l'espace d'état du système. Le RdP est

contrôlé par inhibition d'événements appartenant à un sous ensemble Σ_c des événements contrôlables. Cet ensemble n'est pas spécifié directement mais est défini en munissant certaines transitions de places de contrôle.

Holloway et ses co-auteurs ont développé une approche pour une classe très large de RdP y compris les RdP non saufs (Holloway et al. 1996). Les inconvénients de cette approche sont que les expressions logiques associées aux transitions sont souvent complexes.

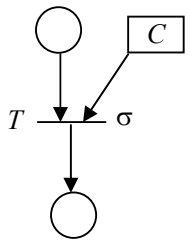


Fig. 2.24 Un RdP contrôlé

La condition de franchissement de transition contrôlée *T* correspond à la présence de jetons dans les places d'entrée de *T* et d'un jeton fictif dans les places de contrôle. On peut interpréter un contrôle comme étant un commutateur binaire qui laisse passer, ou ne laisse pas passer les jetons des ports d'entrée aux ports de sortie. La théorie de Krogh et Holloway sur le contrôle des SED représenté par des RdP tire avantage de la puissance d'expression des RdP pour modéliser les systèmes. Cette richesse permet de modéliser de façon simple des SED qui n'auraient pu être représentés pour des raisons de complexité. Dans le chapitre 5 nous présenterons une méthode de synthèse de contrôleur partant du même concept de base.

Uzam et Wonham ont présenté dans (Uzam et al. 2006) une approche hybride pour construire le contrôleur. Ils ont proposé un contrôleur AUTONET[1] pour calculer les conditions de franchissement des transitions contrôlables. L'avantage de cette approche c'est qu'elle est applicable sur tous les types de RdP mais la taille de modèle du système augmente considérablement. Le contrôleur et le modèle d'un exemple ont été présentés dans la figure 2.25. L'unique avantage de cette approche est d'avoir le même formalisme RdP pour représenter le contrôleur et le procédé physique.

2.4.3 Synthèse de la commande avec la théorie des régions

Ce paragraphe représente une méthode optimale pour la synthèse de superviseurs sous forme de RdP basée sur la théorie des régions (Badouel et al. 1995, Ghaffari et al. 2003b). La notion d'optimalité est liée au nombre d'états atteignables par le système en boucle fermée, c'est-à-dire sous supervision.

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

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

[1] Automaton - Net

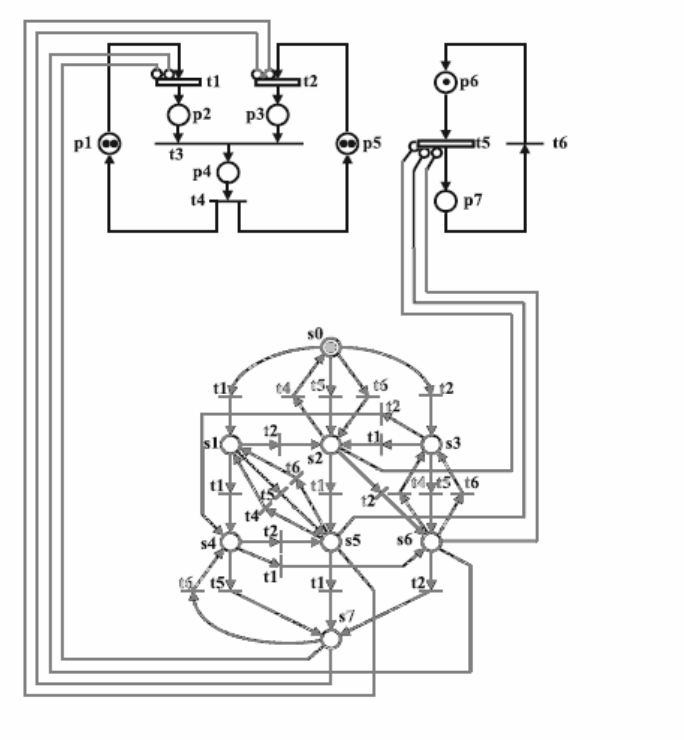


Fig. 2.25. Un modèle de système contrôlé par la méthode hybride présentée par Uzam
(Partie grise présente le modèle du contrôleur et la partie noire le modèle du système)

Etant donné un RdP borné quelconque et un ensemble de marquages interdits, on cherche à déterminer les places qu'il faudrait ajouter au réseau pour restreindre son graphe d'atteignabilité au sous-ensemble de marquages admissibles. Ces places de contrôle constituent la commande à imposer au système pour respecter les contraintes données. La méthode opère en deux phases. Une première phase, basée sur le graphe de marquage, permet de déterminer l'ensemble de marquages admissibles. Puis, en utilisant la théorie des régions, des places de contrôle sont construites pour assurer le comportement désiré. Soit M_0 le marquage initial du modèle RdP à superviser, et soit R_C le graphe d'atteignabilité du modèle RdP du système commandé, pour déterminer l'ensemble des marquages autorisés on suit les étapes suivantes :

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

Etape 2 : Génération du graphe des marquages partiels.

Etape 3 : Détermination de l'ensemble des marquages interdits frontières.

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

Etape 5 : Recherche du comportement autorisé du procédé Rc.

La commande par supervision doit restreindre le comportement du système commandé à R_C en inhibant toute transition contrôlable qui mène à un marquage bloquant ou interdit. Le graphe d'atteignabilité obtenu correspond au comportement vivant maximal permissif du système commandé.

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

La théorie des régions permet de synthétiser un RdP à partir d'un automate à états fini. Cette méthode donne un superviseur optimal mais le nombre des inéquations en nombres entiers engendrées est important et il n'y a pas de méthodes efficaces pour résoudre de tels systèmes.

2.4.4 Vérification de l'absence de blocage par l'idée de siphons

Nous savons qu'il y a deux types d'états interdits (remarque 2.2): ceux qui proviennent des contraintes imposées par la spécification et ceux correspondant aux problèmes de vivacité et de blocage. Pour résoudre le problème de blocage la notion structurelle de siphons peut être utilisée (David et al. 2005). Le problème de blocage apparaît lorsque un siphon devient vide. L'idée principale de cette méthode est de trouver tous les siphons et synthétiser un contrôleur de telle façon à ce que les siphons ne soient jamais vides (Ezpeletaa et al. 1995). Considérons l'exemple de la figure 2.26.

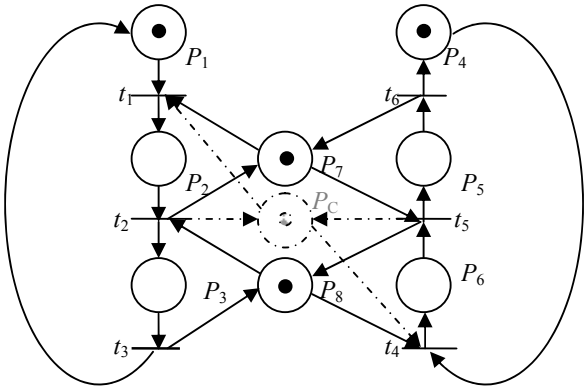


Fig. 2.26. L'exemple de siphons avec sa place de contrôle

Un ensemble de places construit un siphon lorsque toutes les transitions en aval de cet ensemble constituent un sous-ensemble de toutes les transitions qui sont en amont. Par cette définition l'ensemble des places $S_1 = \{P_3, P_5, P_7, P_8\}$ définit un siphon.

Pour qu'un ensemble des places ne devienne pas vide, on peut appliquer la contrainte linéaire comme ci-dessous :

$m_3+m_5+m_7+m_8 \geq 1 \quad \Rightarrow$

$L_1^T = [0, 0, 1, 0, 1, 0, 1, 1], \quad M = [m_1, m_2, \ldots, m_8], \quad k = 1 \quad \Rightarrow$

$L_1^T . \ M \geq k$

Par l'utilisation de l'approche basée sur les invariants de marquage présentée dans la section 2.4.2, on peut construire un contrôleur qui résout le problème pour cet exemple (Iordache et al. 2001).

$$W_C = L^T.W_P = [-1, 1, 0, -1, 1, 0]$$

$$M_0(P_c) = L_1^T . \ M_0 - \ k = 1$$

La place de contrôle ajoutée est indiquée en pointillés dans la figure 2.26. Résoudre ce problème n'est pas toujours facile. Il peut arriver en ajoutant une place de contrôle, qu'un nouveau siphon est créé. Cela repose à nouveau le même problème. En général le nombre de siphons est grand entraînant un grand nombre de places de contrôle. Pour résoudre ce problème, une méthode basée sur des siphons élémentaires a été proposée par (Li et al. 2004). Le problème principal de ces approches est qu'il n'y pas de garantie d'optimalité du contrôleur ; les transitions incontrôlables ne sont pas prises en compte.

2.5 Implantation d'un contrôleur

A la fin de synthèse d'un contrôleur on doit être capable d'appliquer ce contrôleur sur le système industriel. Un modèle RdP n'est pas directement applicable sur un automate programmable industriel (API). Selon la modélisation que nous avons faite, il est possible de transférer le modèle RdP soit vers un modèle Grafcet (Giua et al. 1993a, David et al. 2005) soit un modèle SFC (Music et al. 2000). Il est également possible de transférer le modèle RdP vers LD (Ladder Diagram) qui est un langage de programmation des API (Uzam et al. 1998).

Si nous spécifions le comportement d'un système par le modèle RdP, on peut le transfèrer vers le modèle Grafcet. Pour implanter ce modèle sur un API, (transférer le modèle Grafcet vers le modèle SFC) il faut préciser toutes les entrées\sorties du système et les opérations nécessaires. Pour notre approche de simplification cette dernière étape se situe en aval. Elle est juste étudiée pour montrer que le contrôleur obtenu est directement implantable.

Un modèle spécial de RdP a été présenté dans (Uzam et al. 1998) qui s'appelle le modèle réseaux de Petri automatisé (Automation Petri Nets). Si le modèle RdP est de ce type, la transformation finale vers SFC ou LD sera simple. Les problèmes de transformation d'un modèle RdP vers un modèle Grafcet seront discutés dans le chapitre suivant.

2.6 Superviseur et contrôleur

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

Pour résoudre le problème de ce modèle Balemi, dans (Balemi 1992) a proposé une approche entrées/sorties pour le contrôle des systèmes à événements discrets. Le sous-alphabet Σ_c s'interprète alors comme l'alphabet des commandes qui sont des entrées du procédé, les événements incontrôlables Σ_u modélisant les sorties du procédé. Le système réalise alors une espèce de fonction de transfert de l'action des entrées vers leur effet sur les sorties. Le modèle du superviseur devient alors tout à fait symétrique: il reçoit en entrée les sorties Σ_u du système et produit en sortie les commandes Σ_c du système. Le comportement du superviseur (que nous appellerons contrôleur), est encore modélisé par un langage sur $\Sigma = \Sigma_c \cup \Sigma_u$. Le comportement du procédé à contrôler est toujours décrit par le langage *L(P)* et la notion de langage marqué, tant pour le système que pour le contrôleur reste valable.

Dans ce nouveau schéma, le cadencement des événements est partagé entre le système et son contrôleur. Dans le modèle avec superviseur, le système impose au superviseur l'acceptation de tout événement incontrôlable. Dans le modèle avec contrôleur, le contrôleur n'a pas droit de produire les événements incontrôlables. Le superviseur interdit certains d'événements d'être générés dans le procédé et le contrôleur produit des événements qui agissent sur le procédé. Ces deux méthodes pour construire un système contrôlé sont présentées dans la Figure 2.27.

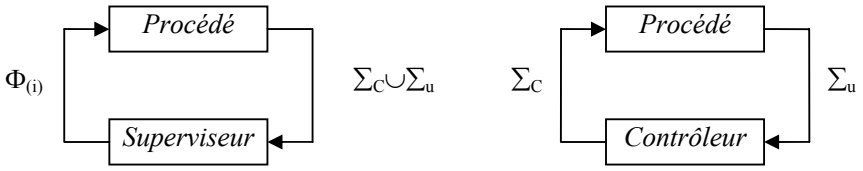


Fig. 2.27. a) l'approche originale de Ramadge/Wonham,
b) l'approche entrées/sorties de Balemi

La différence importante entre contrôleur et superviseur, c'est que le contrôleur peut forcer des événements contrôlables dans le procédé et le contrôleur est nécessairement déterministe alors que le superviseur peut être non déterministe. Dans notre approche nous construirons un contrôleur qui peut empêcher ou forcer les événements contrôlables.

2.7. Conclusion

Malgré les nombreux travaux basés sur la théorie de R&W, il existe encore des difficultés d'implantation que l'on peut regrouper selon les quatre points suivants :

Les événements forcés : Dans la théorie de la supervision, il est supposé que le procédé génère des événements de façon spontanée. Le superviseur peut seulement interdire des événements contrôlables, et ne peut en aucun cas forcer des événements dans le procédé.

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

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

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

Dans cette approche nous allons utiliser la méthode qui simplifie ces problèmes. Pour les problèmes de modélisation et complexité, nous utilisons le modèle RdP, l'analyse sera réalisée par les automates. Pour résoudre le problème d'implantation, nous utiliserons les RdP sauf qui sont transformables en un Grafcet (Giua et al. 1993a, David et al. 2005).

L'existence d'événements incontrôlables entraîne celle des états interdits. Nous avons vu différentes approches permettant de passer des états interdits à des contraintes linéaires. Le nombre de contraintes linéaires peut être très grand augmentant ainsi la complexité du système synthétisé.

Dans les chapitres 3 et 4 nous présenterons des méthodes pour la simplification du nombre et de la taille des contraintes.

Dans le chapitre 5 nous développerons une nouvelle méthode pour construire un contrôleur où des prédicats seront associés aux transitions.

Chapitre 3

Simplification des contraintes linéaires pour les réseaux de Petri conservatifs et saufs

Dans ce chapitre nous allons présenter une méthode de réduction du nombre et de la taille (*borne*) *des contraintes linéaires déduites des états interdits. Cette méthode est applicable sur les RdP conservatifs et saufs. Par l'utilisation de cette méthode nous pouvons construire un contrôleur maximal permissif.*

Chapitre 3

Simplification des contraintes linéaires pour les réseaux de Petri conservatifs et saufs

3.1 Introduction

Dans le chapitre précèdent nous avons vu que la méthode de Giua nous permet de construire un contrôleur pour empêcher d'atteindre les états interdits d'un RdP sauf et conservatif. L'inconvénient majeur de cette approche est dû au grand nombre de contraintes linéaires qui augmente la complexité du système contrôlé. Dans ce chapitre et le suivant, nous présentons deux méthodes systématiques pour réduire la taille et le nombre des contraintes linéaires qui sont déduites des états interdits. La méthode qui sera présentée dans ce chapitre, est applicable sur le RdP sauf et conservatif (Dideban et al. 2005). Pour cela nous avons besoin de l'ensemble des états conservatifs[1] construit à partir des invariants. Cet ensemble est plus grand que l'ensemble des états accessibles. Cependant, les calculs nécessaires se font une seule fois et hors ligne.

Nous avons vu qu'à partir de l'ensemble des états interdits frontières, il est possible de déduire les contraintes linéaires équivalentes (Giua et al. 1993b).

Soit M_i ($M_i^T = [m_{i1}, m_{i2}, \ldots, m_{iN}]$) un état interdit[2] de l'ensemble $\mathcal{M}_{IF}$ (Définition 2.6). Le support de M_i tel que défini dans définition 1.5 est : *Support* (M_i) = $\{P_{i1}P_{i2}P_{i3}...P_{in}\}$ et correspond à l'ensemble des places marquées de l'état M_i. La contrainte linéaire interdisant cet état est présentée comme ci-dessous :

$$\sum_{k=1}^{n} m_{ik} \leq n-1 \qquad (3\text{-}1)$$

Où n = Card [Support (M_i)] est le nombre de places marquées du marquage M_i, et m_{ik} est le nombre de marques (0 ou 1) dans la place P_{ik} de l'état M_i.

On peut réécrire la contrainte linéaire déduite d'un état interdit par l'expression ci-dessous qui a été introduite dans la définition 2.11 :

$$M_i^T . \ M \leq \text{Card [Support } (M_i)] - 1 \qquad (3\text{-}2)$$

Par exemple :

$M_i^T = [0, 1, 1, 0, 0, 0, 1] \Rightarrow$ Card [Support (M_i)] = 3 $\quad \Rightarrow \ m_2 + m_3 + m_7 \leq 2$

La relation (3-1) interdit seulement l'état M_i pour les réseaux conservatifs et saufs. Lorsque les réseaux ne sont pas saufs, il est clair que nous ne pouvons pas représenter chaque marquage par une contrainte équivalente. Par exemple pour le marquage M^T = [0, 1, 1, 0, 0, 0, 2], la contrainte

[1] Il sera défini dans la suite de ce chapitre

[2] Lorsque il n'y pas d'ambiguïté, le mot frontière sera supprimé

équivalente peut interdire plusieurs états. C'est le cas de la contrainte $m_2 + m_3 + m_7 \leq 3$ où l'état $M'^{T} = [0, 1, 2, 0, 0, 0, 1]$ sera interdit.

Le problème pour les réseaux non conservatifs sera étudié dans le chapitre suivant.

En général dans les systèmes réels, il y a un grand nombre d'états interdits, donc un grand nombre de contraintes linéaires. Si pour chaque contrainte linéaire, une place de contrôle est ajoutée, la complexité du système augmente. Le contrôleur doit avoir pour chaque état interdit frontière une place de contrôle. Pour l'exemple 2.2 du chapitre précédent que nous allons utiliser aussi dans ce chapitre, nous avons 5 états interdits et donc 5 contraintes linéaires qui sont :

$\mathcal{M}_{IF} = \{P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

$m_2 + m_5 + m_7 \leq 2$

$m_3 + m_5 + m_8 \leq 2$

$m_2 + m_6 + m_8 \leq 2$

$m_2 + m_4 + m_9 \leq 2$

$m_1 + m_5 + m_9 \leq 2$

On peut présenter chaque contrainte sous la forme d'une paire en ajoutant à l'état frontière interdit la borne de la contrainte correspondante. On obtient alors un ensemble C_{IF} tel que décrit ci-dessous :

$C_{IF} = \{(P_2P_5P_7, 2), (P_3P_5P_8, 2), (P_1P_5P_9, 2), (P_2P_6P_8, 2), (P_2P_4P_92)\}$

Le problème que l'on se propose de résoudre est de réduire la taille (la borne) et le nombre des contraintes par la manipulation des différents espaces d'états (autorisé, interdit, non atteignable). C'est ce que nous allons présenter dans la suite de ce mémoire.

3.2 Simplification du nombre et de la borne des contraintes

3.2.1 Présentation intuitive de la méthode de simplification

Il arrive que dans un système complexe composé de plusieurs éléments, l'état de certains de ces éléments n'ait pas d'influence sur le comportement global du système. Pour illustrer cette idée, reprenons l'exemple 2.2 des deux machines et d'un robot. Le modèle RdP de ce système est représenté dans la figure 3.1.

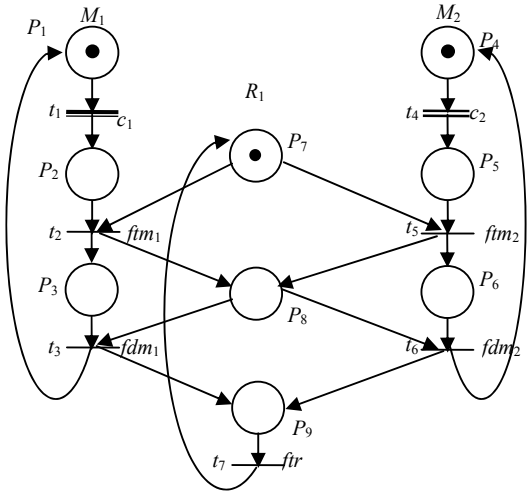


Fig. 3.1. Modèle RdP de l'exemple 2.2

Considérons l'état interdit M_i où la machine M_1 est en marche (Place P_2 marquée) et le robot est en train de transférer la pièce (Place P_9 marquée). Cet état sera interdit quelque soit l'état de la machine M_2. Cette situation pour le robot et la machine M_1 (X_i sur la figure 3.2) est interdit et peut être représentée par la contrainte $m_2 + m_9 \leq 1$. Par cette contrainte deux états $P_2P_4P_9$ et $P_2P_5P_9$ seront interdits. Cette contrainte simplifiée permet de considérer seulement le parti X_i.

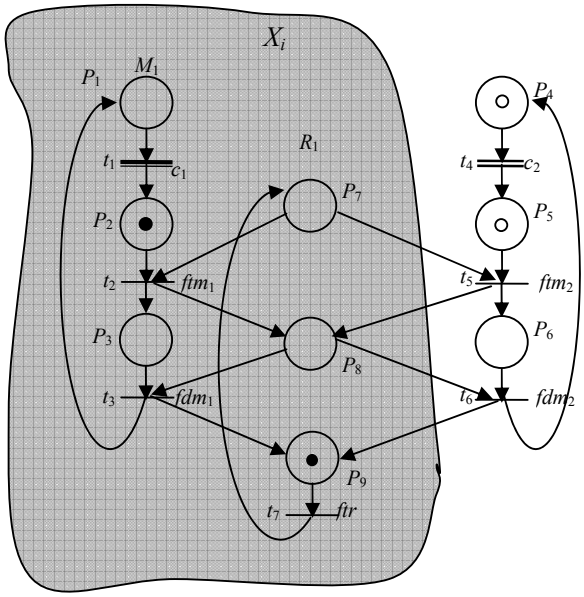


Fig. 3.2. Modèle RdP avec séparation de l'état de machine M_2 de l'état du système

Dans la section suivante nous présenterons des propriétés pour supprimer systématiquement des places redondantes au sens du couple contrainte-invariant et diminuer ainsi le nombre des contraintes.

3.2.2 Simplification par l'invariant

Dans les RdP conservatifs en plus de contraintes linéaires qui se déduisent des états interdits, nous avons aussi des équations déduites des invariants. Nous pouvons utiliser ces équations pour simplifier les contraintes à l'aide de la propriété ci-après.

Propriété 3.1 :

Soit $\mathcal{M} = \{(P_1P_k \ldots P_j), (P_2P_k \ldots P_j)\ldots,(P_r\ P_k \ldots P_j)\}$ un sous-ensemble des états interdits quelconque d'un graphe de marquage et $m_1 + m_2 + \ldots m_r = 1$ un invariant de marquage du RdP. Les r contraintes linéaires déduites de l'ensemble des états interdits sont équivalentes à une seule contrainte :

$m_1 + m_k + \ldots + m_j \leq n\text{-}1$

$m_2 + m_k + \ldots + m_j \leq n\text{-}1$

$\ldots \qquad \xleftarrow{(m_1+m_2+\ldots m_r=1)} \qquad m_k + \ldots + m_j \leq n\text{-}2$

$m_r + m_k + \ldots + m_j \leq n\text{-}1$

Où n est le nombre de places marquées.

Preuve:

Condition nécessaire:

La somme des tous les contraintes donne la contrainte suivent :

$(m_1 + m_2 + \ldots + m_r) + r(m_k + \ldots + m_j) \leq r(n\text{-}1)$

$m_1 + m_2 + \ldots m_r = 1 \Rightarrow$

$1 + r(m_k + \ldots + m_j) \leq r(n\text{-}1) \quad \Rightarrow \quad m_k + \ldots + m_j \leq n - 1 - 1/r$

(m_i:des nombres entiers) $\quad \Rightarrow \quad m_k + \ldots + m_j \leq n\text{-}2$

Condition suffisante :

$m_k + \ldots + m_j \leq n\text{-}2$

$\forall\, i \in \{1,\ldots,r\}\ m_i \leq 1 \quad \Rightarrow \quad m_i + m_k + \ldots + m_j \leq n\text{-}1$

❑

Soit $\mathcal{M}_1 = \{(P_2P_5P_7),(P_2P_5P_8),(P_2P_5P_9)\}$ un sous-ensemble des états interdits et $m_7 + m_8 + m_9 = 1$:

$m_2 + m_5 + m_7 \leq 2$

$m_2 + m_5 + m_8 \leq 2 \quad \xleftrightarrow{(m_7+m_8+m_9=1)} \quad m_2 + m_5 \leq 1$

$m_2 + m_5 + m_9 \leq 2$

Donc on peut utiliser une seule contrainte au lieu de trois contraintes.

Remarque 3.1: Il est possible d'utiliser cette propriété plus d'une fois pour réduire l'ensemble des états interdits.

❑

Par exemple, supposons que l'ensemble des états interdits d'un RdP conservatif et sauf est le suivant :

$\mathcal{M}_1 = \{(P_1P_4P_7P_{10}), (P_1P_4P_7P_{11}), (P_1P_4P_8P_{10}), (P_1P_4P_8P_{11}), (P_1P_4P_9P_{10}), (P_1P_4P_9P_{11})\}$

Supposons également que nous avons deux invariants :

$m_{10} + m_{11} = 1$, $m_7 + m_8 + m_9 = 1$

Il est possible de diviser l'ensemble $\mathcal{M}_1$ en trois sous-ensembles. Pour chacun d'entre eux on peut utiliser la propriété 3-1.

$\mathcal{M}_{11} = \{(P_1P_4P_7P_{10}), (P_1P_4P_7P_{11})\}$

$m_1 + m_4 + m_7 + m_{10} \leq 3$

$m_1 + m_4 + m_7 + m_{11} \leq 3 \quad \xleftrightarrow{(m_{10}+m_{11}=1)} \quad m_1 + m_4 + m_7 \leq 2 \Rightarrow (P_1P_4P_7)$

$\mathcal{M}_{12} = \{(P_1P_4P_8P_{10}), (P_1P_4P_8P_{11})\}$

$m_1 + m_4 + m_8 + m_{10} \leq 3$

$m_1 + m_4 + m_8 + m_{11} \leq 3 \quad \xleftrightarrow{(m_{10}+m_{11}=1)} \quad m_1 + m_4 + m_8 \leq 2 \Rightarrow (P_1P_4P_8)$

$\mathcal{M}_{13} = \{(P_1P_4P_9P_{10}), (P_1P_4P_9P_{11})\}$

$m_1 + m_4 + m_9 + m_{10} \leq 3$

$m_1 + m_4 + m_9 + m_{11} \leq 3 \quad \xleftrightarrow{(m_{10}+m_{11}=1)} \quad m_1 + m_4 + m_9 \leq 2 \Rightarrow (P_1P_4P_9)$

Ainsi, en appliquant la propriété 3-1, combinée à l'invariant $(m_{10} + m_{11} = 1)$, le résultat est :

$\mathcal{M}_2 = \{(P_1P_4P_7), (P_1P_4P_8), (P_1P_4P_9)\}$.

De même il est possible d'utiliser la propriété 3-1 avec d'autres éventuels invariants tel que $m_7 + m_8 + m_9 = 1$:

$$\begin{array}{l} m_1 + m_4 + m_7 \le 2 \\ m_1 + m_4 + m_8 \le 2 \\ m_1 + m_4 + m_9 \le 2 \end{array} \quad \xleftarrow{(m_7 + m_8 + m_9 = 1)} \quad m_1 + m_4 \le 1 \quad (P_1P_4)$$

Le résultat final sera

$\mathcal{M}_3 = \{(P_1P_4)\} \quad \Rightarrow \quad m_1 + m_4 \le 1$

Reprenons l'exemple précèdent mais en supposant cette fois que :

$\mathcal{M}_1 = \{(P_1P_4P_7P_{10}), (P_1P_4P_7P_{11}), (P_1P_4P_8P_{10}), (P_1P_4P_8P_{11})\}$

À la fin de la première simplification, le résultat est :

$\mathcal{M}_2 = \{(P_1P_4P_7), (P_1P_4P_8)\}$.

Dans ce cas particulier, l'application directe de la propriété 3-1 ne nous permet pas de simplifier davantage les contraintes équivalentes. Mais est-il possible d'exploiter davantage les invariants, pas nécessairement sous la même forme pour simplifier encore les contraintes ? c'est ce que nous allons voir dans la section qui suit.

3.2.3 Simplification par l'invariant partiel

Une situation intéressante peut arriver lorsque l'on considère l'ensemble des états interdits. A cause de l'absence d'une ou plusieurs places comme nous l'avons remarqué ci-dessus, nous ne pouvons pas utiliser la relation d'invariant. Par exemple, dans l'ensemble $\mathcal{M}_2$, à la fin de la section précédent, nous ne trouvons pas $P_1P_4P_9$, faute de quoi nous ne pouvons pas utiliser d'invariant pour plus de simplification. Néanmoins, on peut mettre à jour un autre type de relation du type : « il n'est pas possible que P_7 et P_8 soient marquées en même temps ». En effet :

$m_7 + m_8 + m_9 = 1$ et $m_9 \ge 0 \quad \Rightarrow \quad m_7 + m_8 \le 1$

Dans ce cas, au lieu d'une égalité on peut utiliser l'inégalité $m_7 + m_8 \le 1$, cela peut aider à générer de nouvelles simplifications comme nous allons le voir ci-dessous.

Définition 3.1 : Soit $\mathcal{P}_i = \{P_1, \ldots, P_n\}$ l'ensemble des places d'un invariant, $m_1 + \ldots + m_n = 1$. L'ensemble $\mathcal{P}_{i1} = \{P_1, \ldots, P_m\}$, est un invariant partiel si $\mathcal{P}_{i1} \subset \mathcal{P}$. Dans ce cas on peut écrire : $m_1 + \ldots + m_m \le 1$, avec $m < n$.

❑

Remarque 3.2 : Pour cette nouvelle simplification, le nombre de places et la borne de la contrainte ne sont plus corrélés, il faut donc indiquer la borne de la contrainte.

$c_1 = (P_{i1}\ P_{i2}\ \ldots\ P_{in}, k)$

Lorsque la borne n'est pas indiquée, cela signifie que la borne k est égale à n-1.

❑

La propriété 3.2, donnée ci-dessous, établit cette nouvelle simplification.

Propriété 3.2 :

Soit $\mathcal{M} = \{(P_1P_i \ldots P_j, k)\ (P_2P_i \ldots P_j, k), \ldots, (P_r\ P_i \ldots P_j, k)\}$ un sous-ensemble de contraintes quelconque (états interdits et borne correspondante) et $m_1 + m_2 + \ldots m_r \le 1$.

Les r contraintes linéaires sont équivalentes à une seule contrainte comme indiqué ci-dessous :

$$
\begin{aligned}
& m_1 + m_i + \ldots + m_j \le k \\
& m_2 + m_i + \ldots + m_j \le k \\
& \ldots \qquad \xleftarrow{(m_1 + m_2 + \ldots m_r \le 1)} \qquad (m_1 + m_2 + \ldots + m_r) + m_i + \ldots + m_j \le k \\
& m_r + m_i + \ldots + m_j \le k
\end{aligned}
\tag{3-3}
$$

Preuve:

Condition nécessaire

$\forall\, q \in \{1,\ldots, r\} \qquad m_q + m_i + \ldots + m_j \le k \;\Rightarrow\; m_i + \ldots + m_j \le k$

Par la contrainte d'invariant partiel : $m_1 + m_2 + \ldots + m_r \le 1 \;\Rightarrow$

La somme des deux contraintes donne : $(m_1 + m_2 + \ldots + m_r) + m_i + \ldots + m_j \le k + 1$

Nous allons montrer que la borne $k+1$ n'est jamais atteinte. En effet supposons qu'elle est atteinte, alors $(m_1 + m_2 + \ldots + m_r = 1)$ et $m_i + \ldots + m_j = k$. Cependant, si $m_i + \ldots + m_j = k$, et en tenant compte de l'ensemble des contraintes (3-3), alors $m_1 = m_2 = \ldots = m_r = 0$. Ainsi, $(m_1 + m_2 + \ldots + m_r) = 0$, ce qui n'est pas. Donc la borne $k+1$ n'est jamais atteinte.

Condition suffisante :

$(m_1 + \ldots + m_r) + m_i + \ldots + m_j \le k$

$\forall\, q \in \{1,\ldots, r\} \;\; m_q \ge 0 \;\Rightarrow$

$\forall\, q \in \{1,\ldots, r\} \qquad m_q + m_i + \ldots + m_j \le k$

❑

Considérons à nouveau l'exemple présenté à la fin de la section 3.2.2. Soit l'ensemble des états interdits $\mathcal{M}_1 = \{(P_1P_4P_7P_{10}), (P_1P_4P_7P_{11}), (P_1P_4P_8P_{10}), (P_1P_4P_8P_{11})\}$.

Par l'utilisation de la propriété 3-1 nous avions trouvé l'ensemble des contraintes simplifiées suivant :

$\mathcal{M}_2 = \{(P_1P_4P_7, 2), (P_1P_4P_8, 2)\}$

Considérons l'invariant $m_7 + m_8 + m_9 = 1$, la place P_9 n'apparaît pas dans $\mathcal{M}_2$, cependant on peut écrire : $m_7 + m_8 \le 1$

Par l'utilisation de la propriété 3.2 nous avons :

$$
\begin{aligned}
& m_1 + m_4 + m_7 \le 2 \\
& m_1 + m_4 + m_8 \le 2
\end{aligned}
\quad \xleftarrow{(m_7 + m_8 \le 1)} \quad (m_7 + m_8) + m_1 + m_4 \le 2 \;\Rightarrow\; \mathcal{M}_3 = \{(P_1P_4P_7P_8, 2)\}
$$

Bien que la taille de la contrainte augmente, nous avons ici un avantage certain : le nombre des contraintes diminue et la borne de la contrainte reste inchangée. L'objectif final pour nous est la simplification du modèle final (le nombre de places et le nombre d'arcs). Celui-ci est atteint par la diminution du nombre de contraintes, de plus l'augmentation de la taille de la contrainte avec la borne constante diminue dans certains cas le nombre d'arcs. Ainsi, l'augmentation de la taille de la contrainte n'est pas toujours un point négatif.

3.3 Simplification par l'utilisation des états non atteignables

3.3.1) Etats non atteignables

L'objectif de notre approche est toujours de réduire la borne et le nombre des contraintes. Nous avons vu qu'en disposant de plus de contraintes, nous avons la possibilité d'arriver à un résultat plus simple. Dans le cas général, il y a beaucoup d'états qui ne sont pas utilisés. Si ces états peuvent aider à réduire les contraintes, il sera possible d'utiliser ces ensembles d'états et de les considérer comme des états interdits pour les besoins de la simplification. Cependant, à la fin du processus, il ne sera pas nécessaire de retenir une contrainte qui ne correspond pas à un état interdit effectif. Mais comment peut-on déterminer cet ensemble d'états ?

Ce sont les états non atteignables qu'on peut classer en deux catégories : 1) les états admissibles qui par interdiction des états interdits frontière deviennent non atteignables, et 2) les états non effectivement atteignables à partir du marquage initial. Dans l'exemple 2.2 que nous avons déjà utilisée dans ce chapitre, l'état $P_1P_6P_9$ (figure 3.3) est un état non atteignable à partir de l'état initial $P_1P_4P_7$.

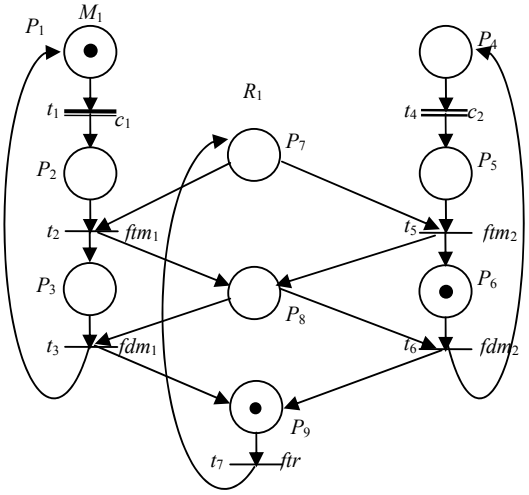


Fig. 3.3. Un état non atteignable à partir d'état initial $P_1P_4P_7$

Dans l'approche développée ici, on va s'intéresser uniquement aux états non atteignables qui permettront d'exploiter davantage les propriétés 3.1 et 3.2. Pour connaître tous ces états, que nous appellerons *états conservatifs*, on calcule l'ensemble des invariants minimaux par la méthode présentée dans (David et al. 2005).

Définition 3.2 : Appelons $\mathcal{M}_C$ l'ensemble des *états conservatifs* pour un système modélisé par un RdP conservatif. On considère un RdP $\mathcal{R}$; et soit m le nombre d'invariants, n_i le nombre de places dans l'invariant i et $\mathcal{P}_i$ l'ensemble des places de cet invariant. L'ensemble $\mathcal{M}_C$ est alors donné par l'expression :

$$\mathcal{M}_C = \left\{ \prod_{i=1}^{m} P_{ij} \,\middle|\, P_{ij} \in \mathcal{P}_i \,, \forall j \in [1,..,n_i] \right\} \tag{3-4}$$

❑

Remarque 3.3 : L'état conservatif est un suite concaténée de m places. Chaque place de cette suite correspond à une place et une seule d'un invariant. On peut donc retrouver plusieurs fois la même place dans un état conservatif. En procédant ainsi, on obtient un ensemble structuré des états conservatifs qui facilitera la conception des algorithmes de simplification.

❑

Remarque 3.4 : Dans un RdP conservatifs les états autorisées et les états interdits sont des états conservatifs, mais il est évident que la réciproque est fasse.

❑

Nous expliquons la construction de cet ensemble sur un exemple simple**.** Prenons le modèle RdP présenté dans la figure 3.4.

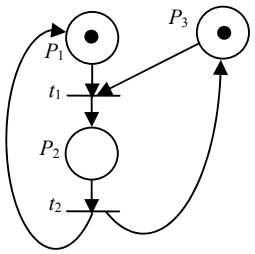


Fig. 3.4. Exemple pour présenter les états conservatifs

Dans cet exemple il y a deux invariants :

$m_1 + m_2 = 1 \quad \Rightarrow \mathcal{P}_1 = \{P_1, P_2\}$

$m_2 + m_3 = 1 \quad \Rightarrow \mathcal{P}_2 = \{P_2, P_3\}$

L'ensemble des états conservatifs sera construit par une arborescence comme indiquée dans la figure 3.5 :

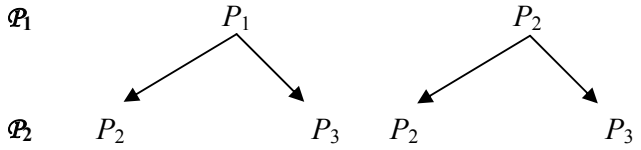


Fig. 3.5. Construire l'ensemble des états conservatifs

Il y a quatre états conservatifs ; $\mathcal{M}_C = \{P_1P_2, P_1P_3, P_2P_2, P_2P_3\}$

L'état P_1P_2 et P_2P_3 sont des états non atteignables. L'état P_2P_2 est équivalent à P_2 représente la marquage $M_i = [0, 1, 0]$.

Comme dans la méthode de simplification des expressions logiques, nous considérons les états non atteignables comme des états interdits. Dans ce cas, l'ensemble des contraintes équivalent aux états interdits (C_{IF}) sera modifié et nous avons un nouvel ensemble de contraintes que nous appelons (C_0). Néanmoins l'ensemble des états autorisés ($\mathcal{M}_A$) toujours reste constant.

Maintenant nous construisons l'ensemble des états conservatifs pour l'exemple de la figure 3.1. Dans ce modèle nous avons 4 invariants indépendants :

$m_1 + m_2 + m_3 = 1 \quad \Rightarrow \mathcal{P}_1 = \{P_1, P_2, P_3\}$

$m_4 + m_5 + m_6 = 1 \quad \Rightarrow \mathcal{P}_2 = \{P_4, P_5, P_6\}$

$m_7 + m_8 + m_9 = 1 \quad \Rightarrow \mathcal{P}_3 = \{P_7, P_8, P_9\}$

$m_3 + m_6 + m_7 + m_9 = 1 \quad \Rightarrow \mathcal{P}_4 = \{P_3, P_6, P_7, P_9\}$

La construction de l'ensemble des états conservatifs est présentée dans la figure 3.6 :

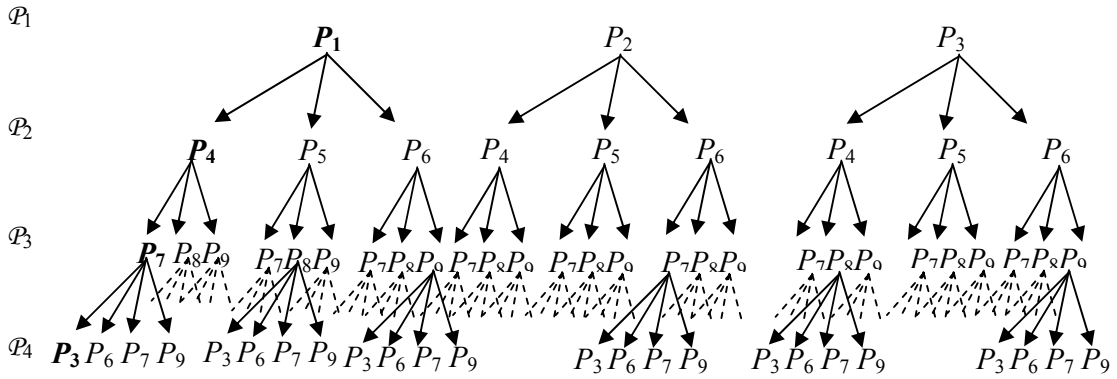


Fig. 3.6. Construction de l'ensemble des états conservatifs pour le RdP figure 3.1

Le nombre d'états conservatifs pour cet exemple est : $N = 3*3*3*4 = 108$.

L'ensemble des états conservatifs est présenté dans le tableau 3.1. Le premier état ($P_3P_7P_4P_1$) a été indiqué sur la figure 3.6.

L'état	S	L'état	S	L'état	S	L'état	S	L'état	S	L'état	S
$P_3P_7P_4P_1$	Φ	$P_3P_7P_4P_2$	Φ	$P_3P_7P_4P_3$	Φ	$P_6P_7P_4P_1$	Φ	$P_6P_7P_4P_2$	Φ	$P_6P_7P_4P_3$	Φ
$P_3P_7P_5P_1$	Φ	$P_3P_7P_5P_2$	Φ	$P_3P_7P_5P_3$	Φ	$P_6P_7P_5P_1$	Φ	$P_6P_7P_5P_2$	Φ	$P_6P_7P_5P_3$	Φ
$P_3P_7P_6P_1$	Φ	$P_3P_7P_6P_2$	Φ	$P_3P_7P_6P_3$	Φ	$P_6P_7P_6P_1$	Φ	$P_6P_7P_6P_2$	Φ	$P_6P_7P_6P_3$	Φ
$P_3P_8P_4P_1$	Φ	$P_3P_8P_4P_2$	Φ	$P_3P_8P_4P_3$	1	$P_6P_8P_4P_1$	Φ	$P_6P_8P_4P_2$	Φ	$P_6P_8P_4P_3$	Φ
$P_3P_8P_5\,P_1$	Φ	$P_3P_8P_5P_2$	Φ	$P_3P_8P_5P_3$	0	$P_6P_8P_5\,P_1$	Φ	$P_6P_8P_5P_2$	Φ	$P_6P_8P_5P_3$	Φ
$P_3P_8P_6\,P_1$	Φ	$P_3P_8P_6P_2$	Φ	$P_3P_8P_6P_3$	Φ	$P_6P_8P_6\,P_1$	1	$P_6P_8P_6P_2$	0	$P_6P_8P_6P_3$	Φ
$P_3P_9P_4P_1$	Φ	$P_3P_9P_4P_2$	Φ	$P_3P_9P_4P_3$	Φ	$P_6P_9P_4P_1$	Φ	$P_6P_9P_4P_2$	Φ	$P_6P_9P_4P_3$	Φ
$P_3P_9P_5\,P_1$	Φ	$P_3P_9P_5P_2$	Φ	$P_3P_9P_5P_3$	Φ	$P_6P_9P_5\,P_1$	Φ	$P_6P_9P_5P_2$	Φ	$P_6P_9P_5P_3$	Φ
$P_3P_9P_6\,P_1$	Φ	$P_3P_9P_6P_2$	Φ	$P_3P_9P_6P_3$	Φ	$P_6P_9P_6\,P_1$	Φ	$P_6P_9P_6P_2$	Φ	$P_6P_9P_6P_3$	Φ
$P_7P_7P_4P_1$	1	$P_7P_7P_4P_2$	1	$P_7P_7P_4P_3$	Φ	$P_9P_7P_4P_1$	Φ	$P_9P_7P_4P_2$	Φ	$P_9P_7P_4P_3$	Φ
$P_7P_7P_5P_1$	1	$P_7P_7P_5P_2$	0	$P_7P_7P_5P_3$	Φ	$P_9P_7P_5P_1$	Φ	$P_9P_7P_5P_2$	Φ	$P_9P_7P_5P_3$	Φ
$P_7P_7P_6P_1$	Φ	$P_7P_7P_6P_2$	Φ	$P_7P_7P_6P_3$	Φ	$P_9P_7P_6P_1$	Φ	$P_9P_7P_6P_2$	Φ	$P_9P_7P_6P_3$	Φ
$P_7P_8P_4P_1$	Φ	$P_7P_8P_4P_2$	Φ	$P_7P_8P_4P_3$	Φ	$P_9P_8P_4P_1$	Φ	$P_9P_8P_4P_2$	Φ	$P_9P_8P_4P_3$	Φ
$P_7P_8P_5\,P_1$	Φ	$P_7P_8P_5P_2$	Φ	$P_7P_8P_5P_3$	Φ	$P_9P_8P_5\,P_1$	Φ	$P_9P_8P_5P_2$	Φ	$P_9P_8P_5P_3$	Φ
$P_7P_8P_6\,P_1$	Φ	$P_7P_8P_6P_2$	Φ	$P_7P_8P_6P_3$	Φ	$P_9P_8P_6\,P_1$	Φ	$P_9P_8P_6P_2$	Φ	$P_9P_8P_6P_3$	Φ
$P_7P_9P_4P_1$	Φ	$P_7P_9P_4P_2$	Φ	$P_7P_9P_4P_3$	Φ	$P_9P_9P_4P_1$	1	$P_9P_9P_4P_2$	0	$P_9P_9P_4P_3$	Φ
$P_7P_9P_5\,P_1$	Φ	$P_7P_9P_5P_2$	Φ	$P_7P_9P_5P_3$	Φ	$P_9P_9P_5\,P_1$	0	$P_9P_9P_5P_2$	Φ	$P_9P_9P_5P_3$	Φ
$P_7P_9P_6\,P_1$	Φ	$P_7P_9P_6P_2$	Φ	$P_7P_9P_6P_3$	Φ	$P_9P_9P_6\,P_1$	Φ	$P_9P_9P_6P_2$	Φ	$P_9P_9P_6P_3$	Φ

S (Situation) ; 1 : Autorisé 0 : Interdit Φ : Non atteignable

Tableau 3.1. Ensemble des états conservatifs

3.3.2 Algorithme de simplification

Nous venons de voir qu'il est possible de diminuer le nombre de contraintes par l'utilisation de propriétés basées sur l'invariant de marquages ou sur un invariant partiel. Nous avons également vu qu'il est possible de considérer les états non atteignables. Nous allons voir ci-dessous que cela permettra d'introduire de nouvelles simplifications. Pour systématiser l'utilisation des propriétés 3.1 et 3.2, nous présentons ci-dessous un algorithme simplifié permettant d'utiliser conjointement les deux propriétés de simplifications présentées dans ce chapitre.

***Algorithme* 3.1 :**

Pas 1 : *Calculer l'ensemble des invariant minimaux* : *soit m* : *le nombre d'invariants* n_i : *le nombre de places dans invariant i,* $\mathcal{P}_i$ *: l'ensemble de places de l'invariant i.* P_{ij} : *jème place de l'ensemble* $\mathcal{P}_i$.

Pas 2 : *Calculer l'ensemble des états conservatifs à partir des invariants* :

$$\mathcal{M}_{\mathcal{P}} = \left\{ \prod_{i=1}^{m} P_{ij} \,\middle|\, P_{ij} \in \mathcal{P}_i \,, \forall j \in [1,..,n_i] \right\}$$

Pas 3 : *A partir de l'ensemble des états autorisés et interdits frontière, indiquer le type de chaque état* : 0 : *interdit* 1 : *autorisé* 2 : *non atteignable*

Pas 4 : *Application de la propriété* 3.1 *jusqu'à ce qu'il n'y ait plus de simplification.*

Pas 5 : *Application de la propriété* 3.2 *jusqu'à ce qu'il n'y ait plus de simplification.*

Pas 6 : *Sélection de l'ensemble final des contraintes.*

Pas 7 : *Fin*

❑

La sélection de l'ensemble final des contraintes sera présenté dans la section 3.4 et l'algorithme détaillé de simplification est présenté en annexe A.

3.3.3 Exemple d'application

On considère à nouveau l'exemple de la figure 3.1. En présence des états non atteignables, le nombre des états qu'on peut utiliser pour la simplification est augmenté. Pour notre exemple, nous allons d'abord considérer les 3 premiers invariants. Cela nous permettra d'avoir une présentation simple de la technique de simplification. A partir de ces 3 invariants, il est possible de construire un invariant qui contient toutes les places du RdP qui est alors conservatif. Comme ce RdP est sauf, alors tous les états atteignables sont couverts par ces invariants. Nous considérerons par la suite l'ensemble des 4 invariants.

En appliquant le « *pas* 1 » de l'algorithme 3.1, nous avons :

$m = 3$

$\mathcal{P}_1 = \{P_1, P_2, P_3\} \Rightarrow n_1 = 3$

$\mathcal{P}_2 = \{P_4, P_5, P_6\} \Rightarrow n_2 = 3$

$\mathcal{P}_3 = \{P_7, P_8, P_9\} \Rightarrow n_3 = 3$

Le nombre des états conservatifs est : $N = 3*3*3 = 27$.

Les résultats des pas 2 et 3 sont donnés dans le tableau 3.2.

L'état	S	L'état	S	L'état	S
$P_7P_4P_1$	1	$P_7P_4P_2$	1	$P_7P_4P_3$	Φ
$P_7P_5P_1$	1	$P_7P_5P_2$	0	$P_7P_5P_3$	Φ
$P_7P_6P_1$	Φ	$P_7P_6P_2$	Φ	$P_7P_6P_3$	Φ
$P_8P_4P_1$	Φ	$P_8P_4P_2$	Φ	$P_8P_4P_3$	1
$P_8P_5\ P_1$	Φ	$P_8P_5P_2$	Φ	$P_8P_5P_3$	0
$P_8P_6\ P_1$	1	$P_8P_6P_2$	0	$P_8P_6P_3$	Φ
$P_9P_4P_1$	1	$P_9P_4P_2$	0	$P_9P_4P_3$	Φ
$P_9P_5\ P_1$	0	$P_9P_5P_2$	Φ	$P_9P_5P_3$	Φ
$P_9P_6\ P_1$	Φ	$P_9P_6P_2$	Φ	$P_9P_6P_3$	Φ

Tableau 3.2. L'ensemble des états conservatifs pour 3 invariants

Maintenant, appliquant la propriété 3.1 correspondant au pas 4.

Pour avoir le maximum de contraintes simplifiées, nous allons considérer de manière équivalente les états interdits et les états non atteignables. Considérons le premier état non atteignable $P_7P_4P_3$. Par permutation de la première position de cet état, correspondant au premier invariant ($\mathcal{P}_1$), on trouve les états $P_7P_4P_2$ et $P_7P_4P_1$. Malheureusement ces états sont autorisés, et il n'y a alors pas de possibilité de simplification. Mais en opérant de la même manière pour les places du deuxième invariant, les trois états $P_7P_4P_3$, $P_7P_5P_3$, $P_7P_6P_3$ sont non atteignables et nous pouvons les simplifier.

$(P_7P_4P_3, 2), (P_7P_5P_3, 2), (P_7P_6P_3, 2) \xrightarrow{m_4+m_5+m_6=1} (P_7P_3, 1)$

En exploitant tous les états interdits et non atteignables, nous arrivons à l'ensemble ci-dessous :

$C_1 = \{P_7P_3, P_5P_2, P_5P_3, P_7P_6, P_6P_2, P_6P_3, P_8P_4P_1, P_8P_2, P_8P_5, P_9P_2, P_9P_3, P_9P_5, P_9P_6\}$

L'ensemble C_1 contient toutes les contraintes simplifiées et celles qui ne l'ont pas été.

La simplification par la propriété 3.1 est résumée sur la figure 3.7.

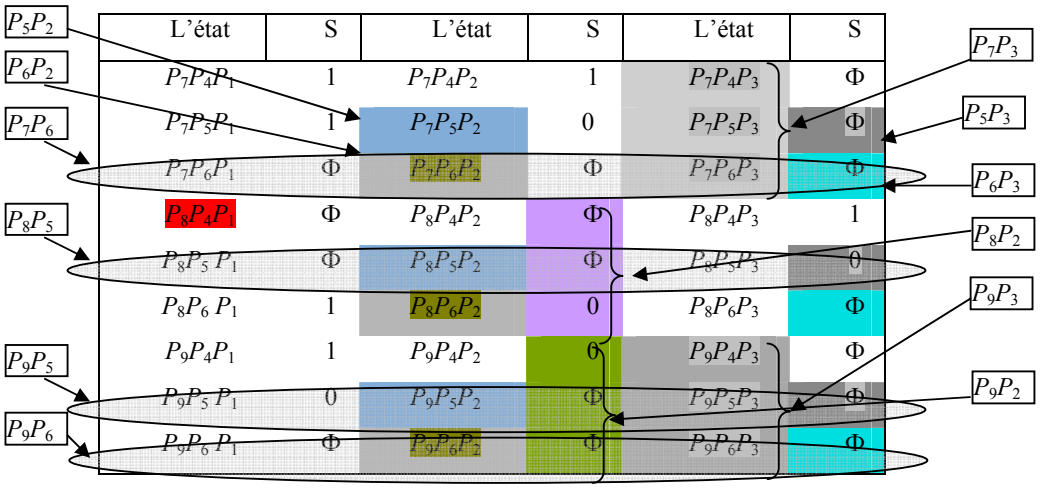


Figure 3.7. Application de la propriété 1

Dans l'algorithme 3.1 nous sommes arrivé au *pas* 5. Nous allons maintenant exploiter la propriété 3.2.

L'ensemble C_1 est réécrit en ajoutant la borne de contrainte. Celle-ci va être prise en compte à partir de maintenant (Remarque 3.2) :

C_2 = {(P_7P_3, 1), (P_5P_2, 1), (P_5P_3, 1), (P_7P_6, 1), (P_6P_2, 1), (P_6P_3, 1), ($P_8P_4P_1$, 2), (P_8P_2, 1), (P_8P_5, 1), (P_9P_2, 1), (P_9P_3, 1), (P_9P_5, 1), (P_9P_6, 1)}.

La propriété 3.2 est applicable pour les deux contraintes c_1= (P_7P_3, 1) et c_2= (P_9P_3, 1) pour donner la contrainte c_3 suivante :

$\{(P_7P_3,1),(P_9P_3,1)\} \xrightarrow{m_7+m_9\leq 1} c_3 = (P_7P_9P_3, 1)$

Nous ne pouvons pas appliquer la propriété 3.2 pour les contraintes c_1= (P_7P_3, 1) et c_2 = (P_5P_3, 1), parce que les places P_7 et P_5 ne sont pas dans le même invariant.

L'ensemble des contraintes après une première utilisation de la propriété 3.2 est donné dans l'ensemble C_3 :

C_3 = {($P_9P_7P_3$, 1), ($P_6P_5P_2$, 1), ($P_5P_2P_3$, 1), ($P_6P_5P_3$, 1), ($P_9P_7P_6$, 1), ($P_6P_2P_3$, 1), ($P_8P_4P_2P_1$, 2), ($P_8P_4P_5P_1$), 2), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1), ($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1)}

Il est possible de simplifier davantage en appliquant encore la propriété 3.2. Pour c_{11} = ($P_6P_5P_2$, 1) et c_{12} = ($P_6P_5P_3$, 1), on obtient la contrainte c_{13} :

$\{(P_6P_5P_2, 1), (P_6P_5P_3, 1)\} \xrightarrow{m_2+m_3\leq 1} c_{13} = (P_6P_5P_2P_3, 1)$

Aussi pour c_{21}= ($P_5P_2P_3$, 1) et c_{22}= ($P_6P_2P_3$, 1) nous avons :

$\{(P_5P_2P_3, 1), (P_6P_2P_3, 1)\} \xrightarrow{m_5+m_6\leq 1} c_{23} = (P_6P_5P_2P_3, 1)$

Les contraintes simplifiés c_{13} et c_{23} sont identiques, on en supprime une. L'ensemble C_4 sera construit en utilisant la propriété 3.2 *jusqu'à ce qu'il n'y ait plus de simplification.*

C_4 = {($P_9P_7P_3$, 1), ($P_6P_5P_2P_3$, 1), ($P_9P_7P_6$, 1), ($P_8P_4P_2P_1$, 2), ($P_8P_4P_5P_1$, 2), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1), ($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1)}

Dans cet ensemble, il y a des contraintes qui ne concernent aucun état interdit. On peut les supprimer. D'autres contraintes peuvent être redondantes dans le sens où elles sont la conséquence de contraintes plus simples et seront aussi supprimées. L'ensemble final C_5 des contraintes est :

C_5 = {($P_6P_5P_2P_3$, 1), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1), ($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1)}

Ces suppressions sont systématisées par la définition de la relation de couverture, notée $R(M_i,c_j)$. Cette relation *re*présente la couverture d'un état interdit M_i par la contrainte c_j.

Définition 3.3 : Soit C = {c_1, c_2, …, c_m} l'ensemble des contraintes simplifiées et $\mathcal{M}_{IF}$ = {M_1, M_1,…, M_N} l'ensemble des états interdits frontières. La relation R: $\mathcal{M}_{IF} \times C \rightarrow$ {0, 1} est définie comme suit :

$$R(M_i,c_j) = \begin{cases} 1 & \text{Si } c_j \Rightarrow c_i \\ 0 & \text{Si non} \end{cases}$$

Ou c_i est la contrainte équivalente à M_i.

❑

La relation de couverture peut être déterminée par la propriété suivante :

Propriété 3.3 : Soit M_i = $P_{i1}P_{i2}$,…, P_{in} un état interdit et c_i =($P_{i1}P_{i2}$,…, P_{in}, n-1) la contrainte correspondante et soit c_j = ($P_{j1}P_{j2}$,…, P_{jm}, k-1) une contrainte simplifiée, où $k \leq m$ et $k \leq n$,.

La contrainte c_j couvre c_i si au moins k éléments de c_j existent dans c_i.

❑

Preuve :

Supposons que les k premiers éléments de c_j sont égaux à k éléments de c_i :

$\{P_{i1}P_{i2},\ldots, P_{ik}\}$= $\{P_{j1}P_{j2},\ldots, P_{jk}\}$ (3-5)

$c_j = (P_{j1}P_{j2},\ldots, P_{jm}, k\text{-}1)$ est vérifiée donc :

$m_{j1} + m_{j2} + \ldots + m_{jm} \leq k\text{-}1 \Rightarrow m_{j1} + m_{j2} + \ldots + m_{jk} \leq k\text{-}1$ (3-6)

(3-5) , (3-6) $\Rightarrow m_{i1} + m_{i2} + \ldots + m_{ik} \leq k\text{-}1$ (3-7)

Le RdP est sauf $\Rightarrow m_{i(k+1)} + m_{i(k+2)} + \ldots + m_{in} \leq n - k$ (3-8)

(3-7) , (3-8) $\Rightarrow m_{i1} + m_{i2} + \ldots + m_{ik} + m_{i(k+1)} + m_{i(k+2)} + \ldots + m_{in} \leq n - k + k\text{-}1 \Rightarrow$

$m_{i1} + m_{i2} + \ldots + m_{in} \leq n\text{-}1$

❑

Exemple 3.1: Prenons l'état interdit $M_1 = P_2P_5P_7$ et la contrainte simplifiée $c_1 = (P_6P_5P_2P_3, 1)$; $\Rightarrow$ n =3, m = 4, k = 2 et 2 places de l'ensemble des places de M_1 (P_2 et P_5) existent dans l'ensemble $\{P_6P_5P_2P_3\}$ donc :

$m_2 + m_3 + m_5 + m_6 \leq 1 \Rightarrow m_2 + m_5 + m_7 \leq 2$ signifie que la contrainte c_1 couvre la contrainte équivalente à l'état M_1.

La relation $R(M_i,c_j)$ entre chaque état interdit et une contrainte de $\mathcal{C}_5$est présenté dans le tableau 3.3.

$R(M_i,c_j)$ $\downarrow c_j$ $M_i \rightarrow$	$P_2P_5P_7$	$P_3P_5P_8$	$P_1P_5P_9$	$P_2P_6P_8$	$P_2P_4P_9$
$(P_6P_5P_2P_3, 1)$	1	1	0	1	0
$(P_9P_8P_2, 1)$	0	0	0	1	1
$(P_9P_8P_5, 1)$	0	1	1	0	0
$(P_9P_2P_3, 1)$	0	0	0	0	1
$(P_9P_5P_6, 1)$	0	0	1	0	0

Tableau 3.3. Relation $R(M_i, c_j)$

L'utilisation des propriétés 3.1 et 3.2 garantit que cet ensemble de contraintes n'interdit aucun état autorisé. Cependant une contrainte peut couvrir plusieurs états interdits, il sera alors nécessaire de faire un choix final. Celui-ci est présenté dans la section suivante où le contrôleur maximal permissif est déterminé.

Nous allons appliquer l'algorithme 3.1 en considérant les 4 invariants du RdP de la figure 3.1.

$\mathcal{C}_5' = \{(P_9P_2P_3, 1), (P_9P_5P_6, 1), (P_9P_8P_2, 1), (P_9P_8P_5, 1), (P_3P_6P_9P_2, 1), (P_3P_6P_9P_5, 1), (P_6P_5P_2P_3, 1), (P_6P_9P_2P_3, 1), (P_3P_9P_5P_6, 1)\}$

3.4 Contrôleur maximal permissif

3.4.1 L'idée principale

Pour avoir un contrôleur optimal nous devrons respecter deux conditions :1) tous les états interdits doivent être effectivement interdits, et 2) aucun état autorisé ne doit être interdit. Si ces deux conditions sont vérifiées, l'ensemble des états autorisés ($\mathcal{M}_A$) est égal à l'ensemble des états qui vérifient les contraintes ($\mathcal{M}_{NI}$). La théorie présentée par Guia (Théorème 2.1) et les propriétés que nous avons utilisées garantissent l'équivalence entre les contraintes de départ et les contraintes simplifiées finales. Nous montrons cette équivalence dans la figure 3.8.

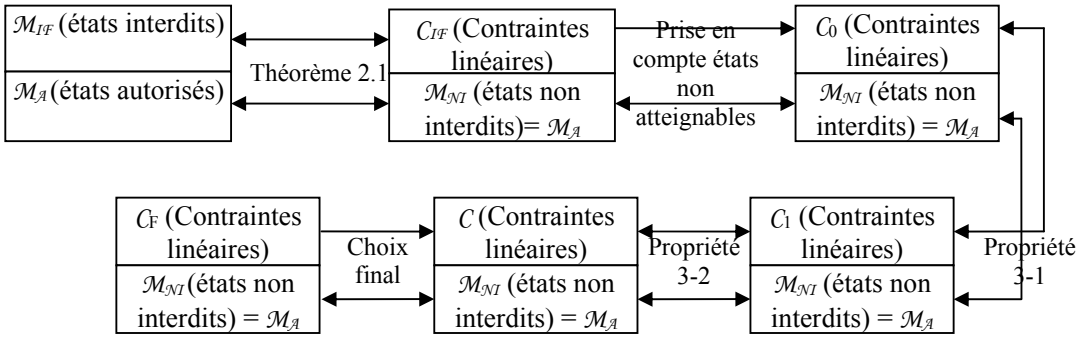


Figure 3.8. Différentes étape de simplification

3.4.2 Le choix optimal

Après la simplification, un même état interdit peut être couvert par plusieurs contraintes. Il faut faire un choix, les règles pour ce choix sont similaires à celles de la méthode de Mac Cluskey pour simplifier les expressions logiques (Morris Mano, 2001).

L'algorithme pour ce choix est présenté ci-après mais avant la présentation de cet algorithme, nous présentons la relation de couverture d'un état par un ensemble de contraintes. Elle indique qu'un état est couvert par au moins une contrainte.

Définition 3.4 : Soit $C = \{c_1, c_2, \ldots, c_m\}$ l'ensemble des contraintes simplifiées et $\mathcal{M}_{IF} = \{M_1, M_1, \ldots, M_N\}$ l'ensemble des états interdits frontières. La relation $Cv(M_i, C) : \mathcal{M}_{IF} \times C \rightarrow \{0, 1\}$ est définie comme suit :

$$Cv(M_i, C) = \bigcup_{j=1}^{m} R(M_i, c_j)$$

❑

Propriété 3.4 : Pour le RdP conservatif et sauf, $\forall M_i \in \mathcal{M}_{IF}$, $Cv(M_i, C) = 1$.

Preuve:

Cette propriété est vraie grâce à l'équivalence qu'il y a dans la construction des ensembles de contraintes (Figure 3.8).

❑

On déduit de la propriété 3.4 que l'ensemble des états autorisé ($\mathcal{M}_A$) est égal à l'ensemble des états accessibles qui est vérifié par l'ensemble de contraintes $C(\mathcal{M}_{NI})$.

L'algorithme 3.2 permet de déterminer l'ensemble final des contraintes.

***Algorithme* 3.2:** *Détermination de l'ensemble des contraintes final.*

Soit $M_i \in \mathcal{M}_{IF}$ une état interdit, $c_k \in C_5$ une contrainte simplifiée, C_F ; l'ensemble final de contraintes (à déterminer). $R(M_i, c_k)$; la relation de couverture entre contrainte c_k et état interdit M_i, $Cv(M_i, C_5)$; le couverture M_i par l'ensemble C_5 .

Pas 1: *Trouver l'état interdit M_i tel que $Cv(M_i, C_5)$ vérifie : i) non nulle, ii) la plus simple, et iii) $CV(M_i, C_F) = 0$;*

S' il n'y a pas de M_i, aller au pas 4 ;

Pas 2: a) *Trouver l'ensemble des contraintes* $\{c_1, \ldots, c_k, \ldots, c_m\}$ *telle que* : $R(M_i, c_k) = 1$,

b) *Trouver la contrainte c_j dans l'ensemble* $\{c_1,\ldots, c_k,\ldots, c_m\}$ *telle qu'elle couvre le plus grande nombre d'état M_r. avec* $Cv(M_r, C_F) = 0$, *et c*) *Prendre la plus simple contrainte c_j en cas égalité.*

Pas 3: *enregistrer c_j dans C_F ; aller au pas* 1.

Pas 4: *Fin*.

❑

L'application de l'algorithme 3.2 sur l'ensemble des contraintes C calculé à partir des 3 invariants et l'ensemble des états interdits $\mathcal{M}_{IF}$ est donné dans le tableau 3.4.

C_5= {($P_6P_5P_2P_3$, 1), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1), ($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1)}

$\mathcal{M}_{IF}$ = {$P_2P_5P_7$, $P_3P_5P_8$, $P_1P_5P_9$, $P_2P_6P_8$, $P_2P_4P_9$}

$\downarrow c_j$ $\quad \rightarrow M_i$	$P_2P_5P_7$	$P_3P_5P_8$	$P_1P_5P_9$	$P_2P_6P_8$	$P_2P_4P_9$	C_F
($P_6P_5P_2P_3$, 1)	√	√	-	√	-	√
($P_9P_8P_2$, 1)	-	-	-	√	√	√
($P_9P_8P_5$, 1)	-	√	√	-	-	√
($P_9P_2P_3$, 1)	-	-	-	-	√	-
($P_9P_5P_6$, 1)	-	-	√	-	-	-
$CV(M_i,C_F)$	√	√	√	√	√	

Tableau 3.4. Relations $R(M_i, c_j)$ et $Cv(M_i,C_F)$ et l'ensemble C_F

On choisit d'abord la contrainte telle qu'il existe un état interdit ne pouvant être couvert que par celle-ci. Ensuite, pour les états interdits qui sont couverts par deux ou plusieurs contraintes, il faut choisir la contrainte qui contient le plus d'états interdits non affectés ou la contrainte la plus simple.

Exemple : l'état $P_2P_5P_7$ est couvert par une seule contrainte, donc il faut retenir la contrainte c_1 = ($P_6P_5P_2P_3$),1). En choisissant cette contrainte, les deux états $P_3P_5P_8$, $P_2P_6P_8$ sont couverts. Il reste alors deux états $P_1P_5P_9$, $P_2P_4P_9$.Pour chaque état nous avons deux choix et il n'y a pas des différences entre ces choix. Nous choissions les deux premières dans le tableau.

C_F = {($P_6P_5P_2P_3$, 1), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1)}

Comme nous venons de le voir dans cet exemple, il peut arriver d'avoir plusieurs réponses équivalentes. Le critère de choix final sera la structure du contrôleur lié au nombre d'arcs entre la place de contrôle et le modèle du système. Après le calcul des étapes de contrôle, il sera possible de faire ce choix.

Pour notre exemple qui est un modèle RdP conservatif, en observant le tableau 3.4, on vérifie bien que $\forall\ M_i \in \mathcal{M}_{IF,}\ Cv(M_i,C_F) = 1$.

Nous avons utilisé la même approche pour notre exemple en considérant 4 invariants. Le résultat est donné dans le tableau 3. 5.

C_5' = {($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1), ($P_9P_8P_2$, 1), ($P_9P_8P_5$, 1), ($P_3P_6P_9P_2$, 1), ($P_3P_6P_9P_5$, 1), ($P_6P_5P_2P_3$, 1), ($P_6P_9P_2P_3$, 1), ($P_3P_9P_5P_6$, 1)}

C'_F = {($P_6P_5P_2P_3$, 1), ($P_6P_9P_2P_3$,1), ($P_3P_9P_5P_6$, 1)}

Pour cet exemple, il n'y a pas de différences importantes entre les réponses dans le cas de 3 ou 4 invariants. Après le calcul des places de contrôle nous verrons les différences qu'il y a entre ces deux cas.

$\downarrow c_j$ $M_i \rightarrow$	$P_2P_5P_7$	$P_3P_5P_8$	$P_1P_5P_9$	$P_2P_6P_8$	$P_2P_4P_9$	C'_F
$(P_9P_2P_3, 1)$	-	-	-	-	√	-
$(P_9P_5P_6, 1)$	-	-	√	-	-	-
$(P_9P_8P_2, 1)$	-	-	-	√	√	-
$(P_9P_8P_5, 1)$	-	√	√	-	-	-
$(P_3P_6P_9P_2, 1)$	-	-	-	√	√	-
$(P_3P_6P_9P_5,1)$	-	√	√	-	-	-
$(P_6P_5P_2P_3, 1)$	√	√	-	√	-	√
$(P_6P_9P_2P_3,1)$	-	-	-	√	√	√
$(P_3P_9P_5P_6, 1)$	-	√	√	-	-	√
$CV(M_i,C_F)$	√	√	√	√	√	

Tableau 3.5. Relations $R(M_i, c_j)$ et $Cv(M_i,C_F)$ en considérant 4 invariants et l'ensemble C_F

3.4.3 Calcul du contrôleur maximal permissif

Pour calculer des places de contrôle pour chaque contrainte linéaire, nous utilisons la méthode basée par les invariants (Yamalidou et al. 1996). Cette méthode a été présentée en détail dans la section 2.4.2.5.

Yamalidou a montré que si tous les événements sont contrôlables, le contrôleur est maximal permissif. Cependant, s'il y a des événements incontrôlables, la méthode développée présentée dans (Moody et al. 2000) et (Basile et al. 2006) ne donne pas la solution optimale dans le cas général. Le problème existe quand une place de contrôle est synchronisée avec une place du procédé par un événement incontrôlable comme indiqué dans la figure 3.9.

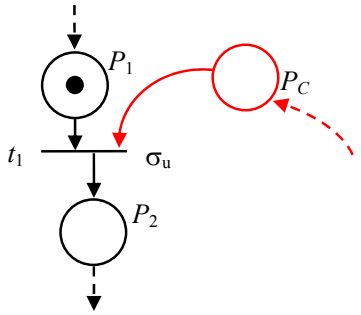


Fig. 3.9. Place de contrôle synchronisée avec le modèle du procédé par un événement incontrôlable σ_u

Dans l'exemple présenté, le procédé ne peut pas respecter des règles de franchissement normal d'un RdP. Supposons que la place de contrôle P_C ne soit pas marquée et P_1 est marquée. A cause de l'incontrôlabilité de la transition t_1, lorsque l'événement σ_u se produit la transition t_1 sera franchie alors que le franchissement de cette transition a été interdit par la place de contrôle. C'est la définition de base du concept de contrôlabilité. Cela signifie que le nombre des états accessibles sera plus grand que cela donné par le modèle RdP. Nous indiquerons cet ensemble d'états par $\mathcal{M}_{AC}$.

Nous allons montrer que le contrôleur obtenu est maximal permissif même si des transitions incontrôlables existent.

Remarque 3.5 : un marquage de $\mathcal{M}_{AC}$ est diffèrent du marquage de $\mathcal{M}_A$ à cause des places de contrôle supplémentaires. C'est seulement un codage pour cet ensemble. Pour pouvoir comparer les deux ensembles de marquages et donc les deux automates, nous n'avons pas considéré les places de contrôle pour les éléments de $\mathcal{M}_{AC}$.

❑

Propriété 3.5 : Soit $\mathcal{M}_{NI}$ l'ensemble d'états autorisés par les contraintes déduites de C_F et $\mathcal{M}_{AC}$ l'ensemble des états accessibles dans le système commandé. Pour les RdP conservatifs, si $\mathcal{M}_A = \mathcal{M}_{NI}$ alors $\mathcal{M}_{AC}$ est isomorphe à $\mathcal{M}_A$ et le contrôleur obtenu est maximal permissif.

Preuve:

Par l'approche basée sur les invariants nous avons toujours :

$$\mathcal{M}_{NI} \subseteq \mathcal{M}_{AC} \tag{3-9}$$

Montrons que :

$\mathcal{M}_{AC} \subseteq \mathcal{M}_{NI}$

Supposons que $\exists\, M_i \in \mathcal{M}_{AC}$ et $M_i \notin \mathcal{M}_{NI}$

$\Rightarrow \exists\, \sigma_u \in \Sigma_u$ et $\exists\, M_j \in \mathcal{M}_{NI},\ M_j \xrightarrow{\sigma_u} M_i$

Par ailleurs $\mathcal{M}_A = \mathcal{M}_{NI} \Rightarrow M_j \in \mathcal{M}_A$ et $M_i \notin \mathcal{M}_A$

$M_i \notin \mathcal{M}_A \Rightarrow M_i \in \mathcal{M}_{IF}$ $\quad (\mathcal{M}_{IF} = \mathcal{M}_R \setminus \mathcal{M}_A)$

Il est clair que si : $M_j \xrightarrow{\sigma_u} M_i$ Donc $M_j \in \mathcal{M}_{IF}$ (La définition des états interdits)

$M_j \in \mathcal{M}_A$ et $M_j \in \mathcal{M}_{IF}$ (contradiction) donc

$$\mathcal{M}_{AC} \subseteq \mathcal{M}_{NI} \tag{3-10}$$

(3-9) et (3-10) $\Rightarrow$ $\mathcal{M}_{NI} = \mathcal{M}_{AC} \Rightarrow$ $\mathcal{M}_{AC} = \mathcal{M}_A$

❑

Maintenant, appliquons cette méthode sur notre exemple.

$\forall\, M_i \in \mathcal{M}_{IF}\ Cv(M_i, C_F) = 1 \Rightarrow$ $\mathcal{M}_{NI} = \mathcal{M}_A \Rightarrow$ Le contrôleur est maximal permissif.

$C_F = \{(P_6P_5P_2P_3, 1), (P_9P_8P_2, 1), (P_9P_8P_5, 1)\}$

Nous considérons d'abord le cas de 3 invariants.

$$\left.\begin{array}{l} m_2 + m_3 + m_5 + m_6 \le 1 \\ m_2 + m_8 + m_9 \le 1 \\ m_5 + m_8 + m_9 \le 1 \end{array}\right\} \Rightarrow L_1^{\,T} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Soit W la matrice d'incidence du RdP correspondant au système contrôlé. Chaque place du contrôleur va ajouter une ligne à la matrice d'incidence du procédé W_R. La matrice W sera composée de deux matrices, la matrice originelle du modèle du procédé W_R et la matrice d'incidence du contrôleur W_c :

$$W = \begin{bmatrix} W_R \\ W_c \end{bmatrix}$$

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

$$W_{C1} = - L^{T}_{1}.W_R$$

$$M_{c\_ini} = k \ - L^{T}_{1}.M_{R\_ini}$$

$$W_{C1} = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$M_{C\_ini1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Le modèle RdP final est représenté dans la figure 3.10.

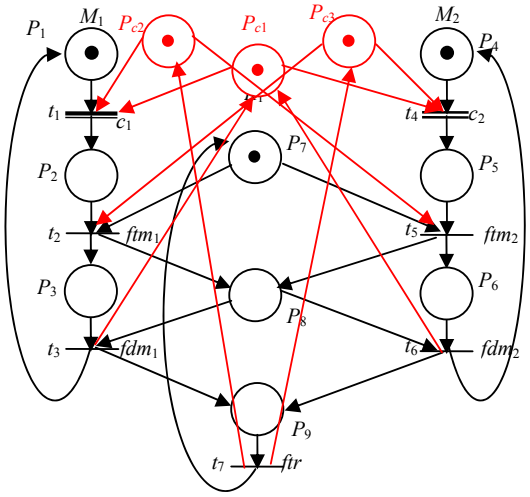


Fig. 3.10. Modèle RdP contrôlé

Nous calculons W_{C2} en utilisant les trois contraintes pour le cas de 4 invariants.

$$\left.\begin{array}{l} m_2 + m_3 + m_5 + m_6 \leq 1 \\ m_2 + m_3 + m_6 + m_9 \leq 1 \\ m_3 + m_5 + m_6 + m_9 \leq 1 \end{array}\right\} \Rightarrow L_2{}^{T} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$W_{C2} = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

La matrice W_{C2} est la même que celle qui a été calculée dans le cas de 3 invariants. Dans la majorité des cas nous arrivons au même résultat. La seule différence ici concerne les contraintes ($P_9P_2P_3$, 1), ($P_9P_5P_6$, 1)} où les sorties de places de contrôle sont modifiées. Ce changement est montré dans la figure 3.11. Il est évident que ce changement ne change pas le graphe de marquage.

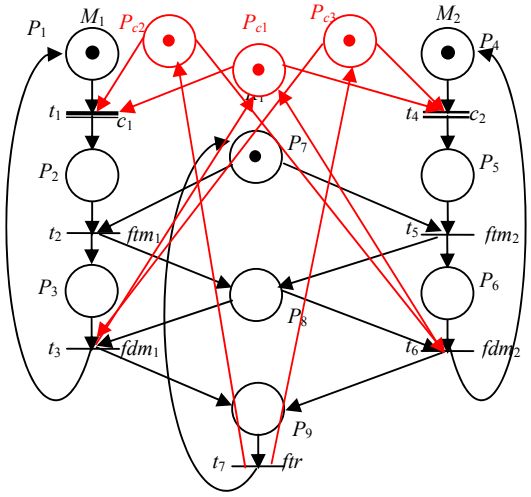


Fig. 3.11. Autre modèle RdP contrôlé

Dans le modèle final nous voyons qu'il y a des places de contrôles qui ont des transitions de sortie incontrôlables (le modèle n'est pas structurellement contrôlable selon la définition (Basile et al. 2006), mais cela ne conduit jamais à un mauvais fonctionnement. Quand une place de contrôle n'est pas marquée, il y a au moins une place d'entrée de cette transition incontrôlable qui appartient au procédé et qui n'est pas marquée. Cette propriété est garantie grâce à la vérification des propriétés 3.3 et 3.4.

3.5 Du modèle RdP vers le modèle Grafcet

Le modèle RdP contrôlé final n'est pas directement implantable dans un automate programmable industriel (API). Pour pouvoir réaliser effectivement la commande, le RdP peut être traduit en *Langage Ladder* ou *Grafcet* ou *SFC*. On peut trouver dans la littérature certaines approches qui proposent des méthodes pour ces traductions (David et al. 2005, Uzam et al. 1998, Giua et al. 1993, Music et al. 2000).

3.5.1. Le modèle RdP interprété de commande

Le modèle RdP interprété de commande (RdPIC) est transformable en un modèle Grafcet. Selon la définition présentée dans (David et al. 2005), un RdPIC possède les quatre caractéristiques suivantes :

1) Il est non temporisé

2) Il est sauf

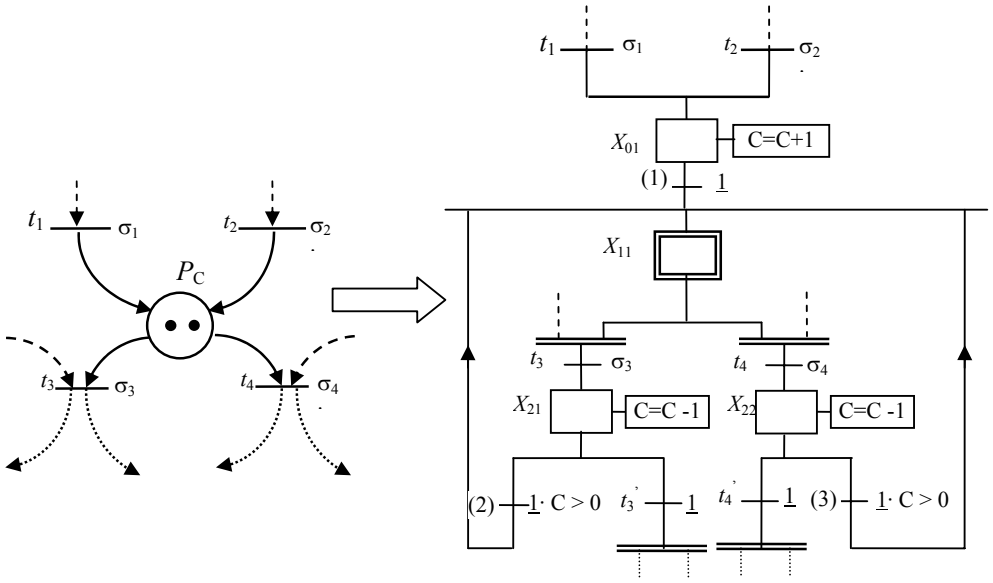
3) Il est déterministe

4) Ses possibilités d'entrées et de sorties sont étendues pour être homogènes avec celles du Grafcet

Notre modèle RdP contrôlé est-il un modèle RdPIC ?

1) Il est non temporisé.

2) Le modèle RdP initial dans notre approche est toujours sauf, mais il est possible d'avoir des places de contrôle synthétisées dont le marquage est non Booléen. On peut transformer la place de contrôle non sauf en un modèle Grafcet équivalent comme présenté dans la figure 3.12.



a) Modèle RdP b) Modèle Grafcet avec C(0) =2

Fig. 3.12. Transformation d'une place de contrôle non sauf à un modèle équivalent Grafcet

3) Le modèle RdP peut être non déterministe, alors que le modèle Grafcet doit être toujours déterministe pour réaliser effectivement la commande. Dans la section 1.4.3 nous avons présenté le conflit effectif dans le RdP sauf et synchronisé. Dans ce cas le même événement est associé aux transitions en sortie des places en conflit. C'est un cas que l'on rencontre rarement. La résolution de ce problème consiste à choisir une stratégie pour lever les conflits : priorité, aléatoire, alternatif, … .

4) Dans un RdPIC seuls des événements interviennent, ceci est compatible avec les réceptivités d'un Grafcet.

Remarque 3.6 : Dans les implantations sur API, à cause du fonctionnement synchrone par cycle,, il est possible que deux événements indépendants se produisent dans le même cycle. Cela peut créer de nouveaux conflits effectifs qu'il faut résoudre au niveau de l'application matérielle.

❑

L'élaboration d'un algorithme systématique de construction du grafcet à partir du RdP contrôlé est une de nos perspectives de recherche.

3.5.2 Modèle Grafcet de l'exemple

Dans notre exemple, la transformation du modèle RdP présenté dans la figure 3.10 à un modèle Grafcet est simple. En effet, toutes les places de contrôle ajoutées sont binaires et le e modèle est déterministe. Le modèle Grafcet final est donné dans la figure 3.13. A_i représente les opérations exécutées par la machine i sur une pièce. B_i représente le déchargement d'une pièce de la machine i. C représente l'opération d'évacuation de la pièce par le robot tandis que D représente le transfert d'une pièce vers un stock.

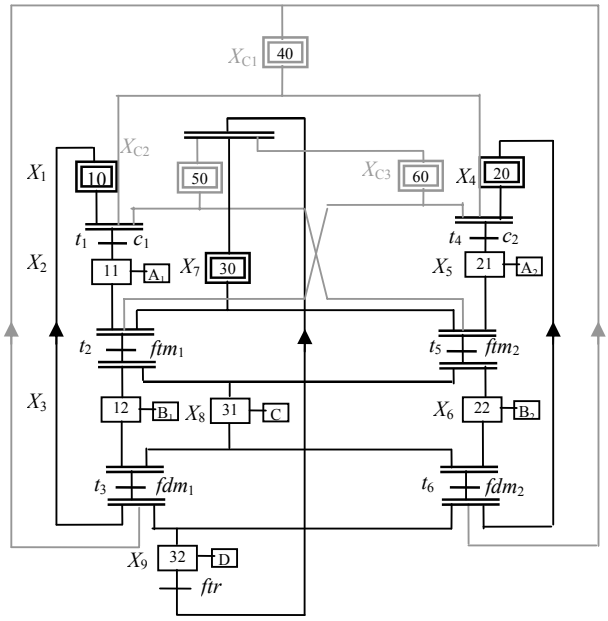


Fig. 3.13. Modèle Grafcet du RdP Figure 3.10

3.6) Etude de cas

Pour présenter l'efficacité de notre méthode, nous l'avons appliquée sur des différents exemples trouvés dans la littérature. Nous les présentons brièvement ci-dessous et nous donnons ensuite les résultats de la simplification obtenue dans le tableau 3.6.

CAS	Nombre d'états interdits	Borne de la contrainte primaire	Nombre de contraintes finales	Borne de la contrainte finale
1	5	2	3	1
2	5	2	2	1
3	6	3	1	2
4	5	1	5	1
5	12	1	1	1
6	3	3	1	1
Total	36	12	13	7

Tableau 3.6 : Réduction du nombre des contraintes pour les différents cas

***Cas* 1** : Un système composé de deux machines et d'un robot étudié dans ce chapitre.

***Cas* 2** : Un système composé de deux machines et d'un poste d'assemblage présenté dans (Kattan 2004).

***Cas* 3** : Un système composé de 5 machines où 3 machines peuvent exécuter un travail 1 et 2 machines un travail 2. La gamme de chaque pièce consiste à faire d'abord l'opération 1 et ensuite l'opération 2 (Dideban et al. 2004).

***Cas* 4** : L'exemple du chat et de la souris présenté dans (Yamalidou et al. 1994).

***Cas* 5** : Un zone d l'exemple AGV (Automated Guided Vehicle) présenté dans Yamalidou et al. 1994).

***Cas* 6** : Boucle Externe de l'AIP (AIP, 2001) qui sera présentée en détail dans le chapitre suivant.

Les résultats de ce tableau montre qu'en moyenne, il y a une réduction de 66% pour le nombre des contraintes et 46% pour la borne des contraintes.

La méthode qui est présentée dans ce chapitre est applicable pour les systèmes modélisés par des réseaux de Petri saufs et conservatifs. Notre objectif est de réduire la borne des contraintes mais lorsque la borne de contrainte est différente de 1 ; il est possible que le marquage initial pour les places de contrôle devienne non sauf. Nous avons proposé ici une méthode de réduction en plusieurs étapes successives, ce qui permet de diminuer la borne, et donc de faire en sorte que le modèle RdP final peut rester sauf.

3.7 Conclusion

Nous avons proposé dans ce chapitre une solution structurelle de synthèse de contrôleurs facile à mettre en œuvre. La solution obtenue par notre approche donne le contrôleur optimal. Cette optimalité vient de l'équivalence entre l'ensemble des marquages autorisées $\mathcal{M}_A$ et l'ensemble des contraintes linéaires déduites de l'ensemble des états interdits $\mathcal{M}_{IF}$. L'idée de base est d'utiliser l'invariant de marquage et les états non atteignables pour simplifier des contraintes linéaires. L'introduction des marquages non atteignables permet de simplifier ces contraintes. Des propriétés générales ont été établies et illustrées sur un exemple simple. En fin nous avons introduit la technique de transformation d'un modèle RdP contrôlé vers un modèle Grafcet. Nous allons dans le chapitre suivant montrer qu'il est possible de lever la propriété de RdP conservatif pour réaliser les simplification des contraintes.

Chapitre 4

Simplification des contraintes linéaires des Réseaux de Petri saufs par la notion de sur - état

Dans ce chapitre nous allons présenter une autre méthode de réduction du nombre et de la borne des contraintes linéaires. Cette méthode est applicable sur les RdP saufs. L'idée principale de cette méthode est basée sur le notion de sur – état. Par l'utilisation de cette méthode nous pouvons construire un contrôleur maximal permissif lorsqu'il existe.

Chapitre 4

Simplification des contraintes linéaires des Réseaux de Petri saufs par la notion de sur-état

4.1 Introduction

Dans le chapitre précédent nous avons présenté une méthode de réduction pour les réseaux conservatifs et saufs. Cette méthode a permis une réduction significative du nombre de contraintes. Elle présente cependant deux inconvénients majeurs : 1) le RdP doit être conservatif, et 2) le nombre des états conservatifs est souvent beaucoup plus grand que le nombre d'états accessibles, ce qui aggrave davantage le problème de l'explosion combinatoire.

Dans ce chapitre nous présentons une méthode de simplification des contraintes applicable pour les RdP saufs non nécessairement conservatifs (Dideban 2005, Dideban et al. 2007). L'avantage de cette méthode est que nous n'aurons besoin ni des états conservatifs ni de la propriété de RdP conservatif. Cette méthode est basée sur la notion fondamentale de « sur – état ».

4.2 Sur – état

4.2.1 Définition

Un sur-état peut représenter un état complet ou une partie de celui-ci. Nous le définissons de la manière suivante ;

***Définition* 4.1** : Soit $M_2 = P_{21}\, P_{22} \ldots P_{2m}$ un état atteignable, $M_1 = P_{11}\, P_{12} \ldots P_{1n}$ est un sur-état de M_2 si :

$M_1 \leq M_2$

❑

Par exemple l'état $M_1 = P_2P_5$ est un sur-état de l'état $M_2 = P_2P_5P_7$.

Reprenons l'exemple 2.2, l'exemple de deux machines et un robot. Le modèle RdP de système est représenté dans la figure 4.1.

Dans notre façon de représenter un état par son support, chaque élément du support apporte une information sur l'état du système. Par exemple dans l'état $P_2P_4P_7$, P_2 signifie que la machine M_1 est dans l'état de marche, P_4 signifie que la machine M_2 est dans l'état de repos et P_7 signifie que la robot

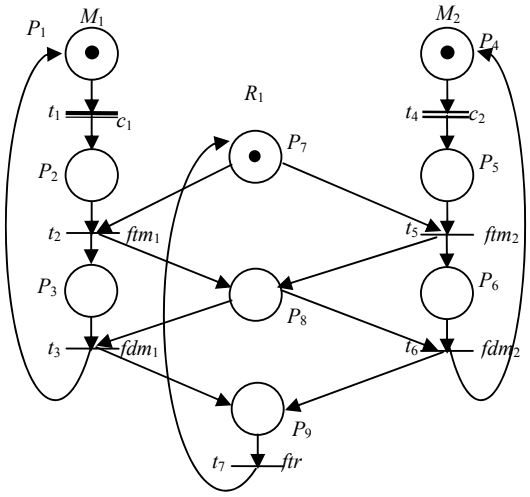


Fig. 4.1. Modèle RdP de l'exemple 2.2

R_1 est également dans l'état de repos. Donc si au lieu d'un état, nous prenons une partie de cet état (un sur-état), la déduction des informations reste possible. Si nous nous intéressons à l'état de repos du robot, c'est P_7 qui le représente. P_7 est un sur-état de l'état $P_2P_4P_7$. En conclusion on peut dire qu'un sur-état représente une situation d'une partie du système et l'autre partie peut être dans différentes configurations. Mais ceci n'est qu'une interprétation qui n'est absolument pas nécessaire au développement théorique de notre approche.

L'appellation *sur-état* dans cette définition vient du fait que la contrainte correspondante à un sur-état couvre celle de l'état. Par exemple la contrainte $m_2 + m_5 \leq 1$ relative au sur-état $M_1 = P_2P_5$ couvre les deux contraintes suivantes :

$m_2 + m_5 + m_7 \leq 2$

$m_2 + m_5 + m_8 \leq 1$

Ces deux contraintes interdisent les états $M_2 = P_2P_5P_7$ et $M_3 = P_2P_5P_8$ qui vérifient bien $M_2 \geq M_1$ et $M_3 \geq M_1$. Donc par utilisation de la seule contrainte $m_2 + m_5 \leq 1$, les deux états M_2 et M_3 sont interdits.

Remarque 4.1 : À chaque sur-état b_i, nous associons une contrainte c_i de la façon suivante

$b_i = (P_{i1}P_{i2}P_{i3} \ldots P_{in}) \Rightarrow c_i = (P_{i1}P_{i2}P_{i3} \ldots P_{in}, n\text{-}1)$

Cela signifie que:

$m_{i1} + m_{i2} + \ldots + m_{in} \leq n\text{-}1$

❑

Remarque 4.2 : Soit $M_i = (P_{i1}P_{i2}P_{i3} \ldots P_{in})$ un sur-état de $M_j = (P_{j1}P_{j2}P_{j3} \ldots P_{jm})$ ($M_i \leq M_j$ et $n \leq m$). Il y a deux relations d'inclusion qui vont en sens contraire, une inclusion ensembliste et une inclusion de marquage :

1) L'ensemble des places marquées dans le sur-état M_i est inclus dans l'ensemble des places marquées dans l'état M_j.

2) L'ensemble des marquages couverts par M_i contient le marquage M_j.

❑

Il est clair que l'application de la contrainte équivalente associée à un sur–état garantit que l'état d'origine sera interdit. Nous le présentons par la propriété suivante :

Propriété 4.1 : Soit M_1 et M_2 deux vecteurs de l'ensemble $\{0, 1\}^N$ et c_1 et c_2 les deux contraintes correspondantes. Si $M_1 \leq M_2$ (M_1est un sur-état de M_2) et c_1 est vraie, donc c_2 est aussi vraie :

$M_1 \leq M_2$ and c_1 : $M_1^T . M \leq$ Card[Support(M_1)]- 1

$\Rightarrow$ c_2: $M_2^T . M \leq$ Card[Support(M_2)]- 1

***Preuve*:**

Le modèle RdP de système est sauf donc :

$(M_2^T - M_1^T).M \leq$ Card[Support(M_2)] - Card[Support(M_1)]

Et:

$M_2^T . M = (M_2^T - M_1^T + M_1^T).M = (M_2^T - M_1^T).M + M_1^T.M$

Par l'utilisation de la contrainte c_1, nous avons :

$(M_2^T - M_1^T).M + M_1^T.M \leq$ (Card[Support(M_2)] - Card[Support(M_1)]) + Card[Support(M_1)] – 1

$\Rightarrow$ $M_2^T . M \leq$ Card [Support (M_2)] - 1

❑

L'utilisation des sur-états n'est pas toujours simple. Parfois il est possible que la contrainte équivalente à un sur-état interdit également des états autorisés. Nous présentons dans ce chapitre une méthode de simplification qui garantit que les contraintes interdisent seulement les états interdits.

Remarque 4.3 : Il est possible de déterminer des contraintes déduites de sur-états, sans tenir compte du fait que ces contraintes peuvent interdire des états autorisés. Dans ce cas, il sera possible que le contrôleur ne soit pas maximal permissif.

❑

4.2.2 Ensemble des sur-états

Dans la section précédente, nous avons vu que l'utilisation des contraintes équivalentes aux sur-états est plus intéressante que des contraintes équivalentes aux états originaux. Elles couvrent en général plusieurs états. Il reste à définir alors quel sur-état doit-on utiliser ?

L'objectif final est de construire l'ensemble minimal de sur-états que l'on peut interdire. Donc plus l'expression du sur-état est simple (en terme du nombre de places marquées), plus elle regroupera de contraintes. Pour atteindre cet objectif, on doit construire l'ensemble des sur-états des états interdits. L'ensemble des sur-états pour un état est défini comme les parties de l'ensemble de places marquées de cet état :

Définition 4.2 : Soit $M = \{P_{i1}\ P_{i2} P_{i3} ... P_{in}\}$ un état du système. L'ensemble des sur-états de M, noté M^{sur} est égal à l'ensemble des parties de M sans l'ensemble vide.

❑

Par exemple, l'état interdit :

$M_1 = P_2 P_5 P_7$ donne :

$M_1^{sur} = \{P_2, P_5, P_7, P_2P_5, P_2P_7, P_5P_7, P_2P_5P_7\}$

Remarque 4.4 : Chaque état est un sur-état de son état d'origine.

❑

Pour les besoins de notre approche, nous allons calculer l'ensemble des sur-états pour les états interdits frontière.

Pour les états interdits frontières M_{IF}, l'union de ces ensembles $\mathcal{M}_{IF}^{sur}$ noté $\mathcal{B}_1$ pour avoir une notation systématique, est calculé comme suit :

$$\mathcal{B}_1 = \bigcup_{i=1}^{Card\,(M_{IF})} M_i^{sur} = \mathcal{M}_{IF}^{Sur}$$

Nous reprenons l'exemple figure 4.1

$\mathcal{M}_{IF} = \{P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$.

Pour cet exemple $\mathcal{B}_1$ est calculé comme ci-dessous :

$M_1 = P_2P_5P_7 \Rightarrow M_1^{sur} = \{P_2, P_5, P_7, P_2P_5, P_2P_7, P_5P_7, P_2P_5P_7\}$

$M_2 = P_3P_5P_8 \Rightarrow M_2^{sur} = \{P_3, P_5, P_8, P_3P_5, P_3P_8, P_5P_8, P_3P_5P_8\}$

$M_3 = P_1P_5P_9 \Rightarrow M_3^{sur} = \{P_1, P_5, P_9, P_1P_5, P_1P_9, P_5P_9, P_1P_5P_9\}$

$M_4 = P_2P_6P_8 \Rightarrow M_4^{sur} = \{P_2, P_6, P_8, P_2P_6, P_2P_8, P_6P_8, P_2P_6P_8\}$

$M_5 = P_2P_4P_9 \Rightarrow M_5^{sur} = \{P_2, P_4, P_9, P_2P_4, P_2P_9, P_4P_9, P_2P_4P_9\}$

$\mathcal{B}_1 = M_1^{sur} \cup M_2^{sur} \cup M_3^{sur} \cup M_4^{sur} \cup M_5^{sur} = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2P_5, P_2P_7, P_5P_7, P_3P_5, P_3P_8, P_5P_8, P_1P_5, P_1P_9, P_5P_9, P_2P_6, P_2P_8, P_6P_8, P_2P_4, P_2P_9, P_4P_9, P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

Les sur-états de l'ensemble $\mathcal{B}_1$ couvrent tous les états interdits. Mais il est possible que certains sur-états couvrent des états autorisés. C'est par exemple le cas du sur-état P_1. Dans la section suivante nous présentons une approche pour trouver les sur-états qui ne couvrent que les états interdits. C'est bien de ceux-là dont nous avons besoin pour faire la synthèse d'un contrôleur.

4.3 Construction de l'ensemble minimal de contraintes

Comme cela a été le cas dans la méthode de simplification présentée dans le chapitre 3, celle présentée ici comportera aussi trois étapes. Dans la première étape, les contraintes seront simplifiées en utilisant la notion de sur-états. Dans la deuxième étape on cherchera à trouver des invariants partiels, proche de l'application de propriété 3.2. Dans la troisième étape, nous appliquons la méthode de construction du contrôleur maximal permissif présentée dans la section 3.4.

4.3.1 Simplification des contraintes en utilisant les sur-états

Soit $\mathcal{M}_{IF}$ l'ensemble des états interdits frontières et $\mathcal{M}_A$ l'ensemble des états autorisés. L'idée de base de la simplification consiste à construire l'ensemble $\mathcal{M}_{IF}^{sur} / \mathcal{M}_A^{sur}$. C'est ce qui est présenté ci-dessous par l'algorithme simple 4.2.

Algorithme 4.1 :

Pas 1 : Calculer $\mathcal{B}_1$ (l'ensemble des sur-états pour tous les états interdits de $\mathcal{M}_{IF}$) et $\mathcal{A}_1$ (l'ensemble des sur-états pour tous les états autorisées de $\mathcal{M}_A$)

Pas 2 :

2.1 Si $\mathcal{B}_1$ est vide allez au pas 3.

2.2 Soit b_i un sur-état de l'ensemble $\mathcal{B}_1$,

Si $b_i \in \mathcal{A}_1$ supprimer b_i de $\mathcal{B}_1$

Si non {

$\mathcal{B}_2 = \mathcal{B}_2 \cup \{b_i\}$,

Supprimer tous les sur-états $b_j \in \mathcal{B}_1$ tels que : $b_i \subseteq b_j$

}

Aller au pas 2

Pas 3 : Fin

❑

Notre objectif est de trouver une méthode permettant de réduire le nombre et la taille des contraintes. Nous avons déjà construit l'ensemble de tous les sur-états des états interdits pour l'ensemble $\mathcal{M}_{IF}$ dans la section 4.2.2:

$\mathcal{B}_1 = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2P_5, P_2P_7, P_5P_7, P_3P_5, P_3P_8, P_5P_8, P_1P_5, P_1P_9, P_5P_9, P_2P_6, P_2P_8, P_6P_8, P_2P_4, P_2P_9, P_4P_9, P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

$\mathcal{B}_1$ couvre tous les états interdits, mais dans $\mathcal{B}_1$ il y a des sur-état aussi simples que P_1 ou P_2. On voit bien que ces sur-états couvrent des états autorisés. Par exemple P_1 couvre les états autorisés $P_1P_4P_7$, $P_1P_4P_9$, $P_1P_5P_7$, $P_1P_6P_8$. Donc il faut supprimer de $\mathcal{B}_1$ tous les éléments de $\mathcal{A}_1$.

Soit $\mathcal{M}_A$ l'ensemble des états autorisées :

$\mathcal{M}_A = \{P_1P_4P_7, P_2P_4P_7, P_3P_4P_8, P_1P_4P_9, P_1P_5P_7, P_1P_6P_8\}$

$\mathcal{A}_1 = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_1P_4, P_1P_7, P_4P_7, P_2P_4, P_2P_7, P_3P_4, P_3P_8, P_4P_8, P_1P_5, P_1P_9, P_4P_9, P_5P_7, P_1P_6, P_1P_8, P_6P_8, P_1P_4P_7, P_2P_4P_7, P_3P_4P_8, P_1P_4P_9, P_1P_5P_7, P_1P_6P_8\}$

On appellera : $\mathcal{B}_2 = \mathcal{B}_1 / \mathcal{A}_1$ l'ensemble des sur-états interdits ne couvrant aucun état autorisé.

Corollaire 4.1 : Soit $\mathcal{B}_1$ l'ensemble des sur-états des états interdits et $\mathcal{A}_1$ l'ensemble des sur-états des états autorisés. L'ensemble des contraintes équivalentes à l'ensemble des sur-états $\mathcal{B}_2 = \mathcal{B}_1 \setminus \mathcal{A}_1$ est vérifié par tous les états autorisés.

❑

Il est possible que certains états interdits ne soient pas couverts par $\mathcal{B}_2$, cela peut arriver dans des RdP non conservatifs. Dans ce cas le contrôleur synthétisé ne sera pas maximal permissif. Ce problème sera ultérieurement traité dans ce chapitre.

Pour notre exemple,

$\mathcal{B}_2 = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2P_5, P_2P_7, P_5P_7, P_3P_5, P_3P_8, P_5P_8, P_1P_5, P_1P_9, P_5P_9, P_2P_6, P_2P_8, P_6P_8, P_2P_4, P_2P_9, P_4P_9, P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

$= \{P_2P_5, P_3P_5, P_5P_8, P_5P_9, P_2P_6, P_2P_8, P_2P_9, P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

Enfin il peut arriver que dans cet ensemble il y ait des sur-états qui couvrent d'autres sur-états. Par exemple $P_2P_6P_8$ couvre par P_2P_6. Donc le sur état $P_2P_6P_8$ dans cet ensemble est redondant et on doit le supprimer. L'ensemble $\mathcal{B}_3$ sera construit comme la suit :

$\mathcal{B}_3 = \mathcal{B}_2 - \{b_{2i} \in \mathcal{B}_2 \mid \exists\, b_{2j} \in \mathcal{B}_2 \;\; b_{2j} \subset b_{2i}\}$

$\mathcal{B}_3 = \{P_2P_5, P_3P_5, P_5P_8, P_5P_9, P_2P_6, P_2P_8, P_2P_9, P_2P_5P_7, P_3P_5P_8, P_1P_5P_9, P_2P_6P_8, P_2P_4P_9\}$

$= \{P_2P_5, P_3P_5, P_5P_8, P_5P_9, P_2P_6, P_2P_8, P_2P_9\}$

4.3.2 Réduction du nombre de contraintes en construisant des invariants partiels

Dans la première méthode de la simplification, deux propriétés ont été présentées dans le cas des RdP conservatifs. Notons que jusqu'ici le résultat de la simplification, est proche de celui obtenu par la méthode de simplification des RdP conservatifs en utilisant uniquement la première propriété de simplification (Propriété 3.1). Pour appliquer la deuxième propriété nous avons utilisé des contraintes d'inégalités déduites des invariants de marquage. Dans la section suivante nous présentons une méthode de construction de contraintes sans le besoin d'invariants. Dans cette méthode comme nous le verrons dans l'exemple, il est possible d'arriver à un nombre de contraintes plus réduit que celui déduit des invariants.

4.3.2.1 Construction de l'invariant partiel

Nous allons construire des invariants partiels indépendamment de la théorie de la conservation classique de RdP. Cette construction repose sur une propriété que nous allons d'abord établir dans le cas de deux sur-états et ensuite généraliser à n sur-états. Cela permettra d'utiliser à nouveau la propriété 3.2.

Remarque 4.5 : Jusqu'ici le sur-état à interdire et la contrainte correspondante contenaient la même information. Mais dans la simplification en utilisant l'invariant partiel que nous développons ici, la borne de la contrainte va changer. Nous manipulerons donc la contrainte au lieu du sur-état.

❑

Propriété 4.2. Soient m_{i1} et m_{i2} les nombres de marques dans les places P_{i1} et P_{i2}. Si $P_{i1}P_{i2}$ n'est pas un sur-état des états autorisés (ne couvre pas un état autorisé), alors l'ensemble des états autorisés vérifie:

$$m_{i1} + m_{i2} \leq 1$$

Preuve :

Supposons que $P_{i1}P_{i2}$ n'est pas un sur-état des états autorisés et que pour l'état autorisé M_i, son marquage ne vérifie pas cette contrainte donc :

$$m_{i1} + m_{i2} > 1$$

Pour les RdP saufs :

$$m_{i1} + m_{i2} = 2 \Rightarrow m_{i1} = m_{i2} = 1$$

Ce qui signifie que le sur-état $P_{i1}P_{i2}$ de M_i est dans les états autorisés. Ceci est contraire à l'hypothèse, donc :

$$m_{i1} + m_{i2} \leq 1$$

❑

La propriété 4.2 montre que si la contrainte supplémentaire pour la simplification dans la propriété 3.2 construite à partir des invariants n'existe pas, on peut néanmoins la construire pour deux variables. Il est possible d'aller au-delà de deux variables. Pour ça nous montrons comment on peut augmenter la taille de la contrainte d'invariant partiel par l'ajout d'une variable.

Supposons que la contrainte d'un invariant partiel soit $m_{i1} + \ldots + m_{in} \leq 1$.Nous montrons par la propriété 4.3 dans quelles conditions on peut développer cette contrainte en ajoutant la variable correspondant au marquage d'une place $P_{i(n+1)}$.

Propriété 4.3 (Extension de la propriété 4.2) : Considérons la contrainte $m_{i1} + \ldots + m_{in} \leq 1$ vérifiée par l'ensemble des états autorisés. Si l'ensemble des sur-états $\{P_{i1}P_{i(n+1)}, \ldots, P_{in}P_{i(n+1)}\}$ ne sont pas dans l'ensemble de sur-état autorisés, alors la contrainte $m_{i1} + \ldots + m_{in} + m_{i(n+1)} \leq 1$ est vérifiée par l'ensemble des états autorisés.

❑

Preuve :

La preuve est proche de la propriété 4-2.

Supposons que l'état autorisé M_i ne vérifie pas cette contrainte donc :

$m_{i1} + m_{i2} + \ldots + m_{in} + m_{i(n+1)} > 1$.

Nous avons :

$$\left.\begin{array}{l} m_{i1} + \ldots + m_{in} \le 1 \\ m_{i(n+1)} \le 1 \text{ (pour les RdP saufs)} \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} m_{i1} + \ldots + m_{in} = 1 \Rightarrow \exists\, m_{ik} = 1 (\, k \in [1,n]) \\ m_{i(n+1)} = 1 \end{array}\right.$$

Ce qui signifie que le sur-état $P_{ik}P_{i(n+1)}$ est un sur-état de M_i. Ce qui n'est pas.

❑

4.3.2.2 Simplification par l'invariant partiel

Le point de départ de cette simplification est l'ensemble des sur-états interdits calculé dans la section 4.3.1. Elle utilise l'invariant partiel déterminé par la propriété 4.3 et la propriété 3.2 que nous rappelons ci-dessous.

Propriété 3.2 : Soit $\mathcal{M} = \{(P_1P_k \ldots P_j, k),\ldots,(P_r P_k \ldots P_j, k)\}$ un sous – ensemble des contraintes vérifiant $m_1 + m_2 + \ldots + m_r \le 1$. Les r contraintes linéaires sont équivalentes à une contrainte comme ci-dessous :

$$\begin{array}{l} m_1 + m_k + \ldots + m_j \le k \\ m_2 + m_k + \ldots + m_j \le k \\ \ldots \\ m_r + m_k + \ldots + m_j \le k \end{array} \xleftrightarrow{(m_1+m_2+\ldots m_r \le 1)} (m_1 + m_2 + \ldots + m) + m_k + \ldots + m_j \le k$$

❑

Maintenant, appliquons les propriétés 4.3 et 3.2 sur notre exemple. Le résultat de la dernière étape permet d'établir à partir de l'ensemble $\mathcal{B}_3$, l'ensemble des contraintes $\mathcal{C}_3$:

$\mathcal{B}_3 = \{P_2P_5, P_3P_5, P_5P_8, P_5P_9, P_2P_6, P_2P_8, P_2P_9\} \Rightarrow$

$\mathcal{C}_3 = \{(P_2P_5, 1), (P_3P_5, 1), (P_5P_8, 1), (P_5P_9, 1), (P_2P_6, 1), (P_2P_8, 1), (P_2P_9, 1)\}$

L'exploitation de la propriété 4.3 consiste à trouver le nombre maximal de sur-états qui ne différent que d'une seule place. C'est le cas de l'ensemble $\{(P_2P_5, 1), (P_3P_5, 1), (P_5P_8, 1), (P_5P_9, 1)\}$. Le sur-état P_2P_3 n'existe pas dans les états autorisés donc la contrainte $m_2 + m_3 \le 1$ est construite. Mais le sur-état P_3P_8 existe dans les états autorisés et il n'est pas possible de créer une contrainte contenant la variable m_8. Cependant il est possible de développer cette contrainte en ajoutant la variable m_9. La figure 4.2 présente les 6 contraintes nécessaires pour la simplification.

L'application de la propriété 3.2, en utilisant les 6 contraintes calculées dans la figure 4.2 donne les 3 résultats indiqués dans la figure 4.3.

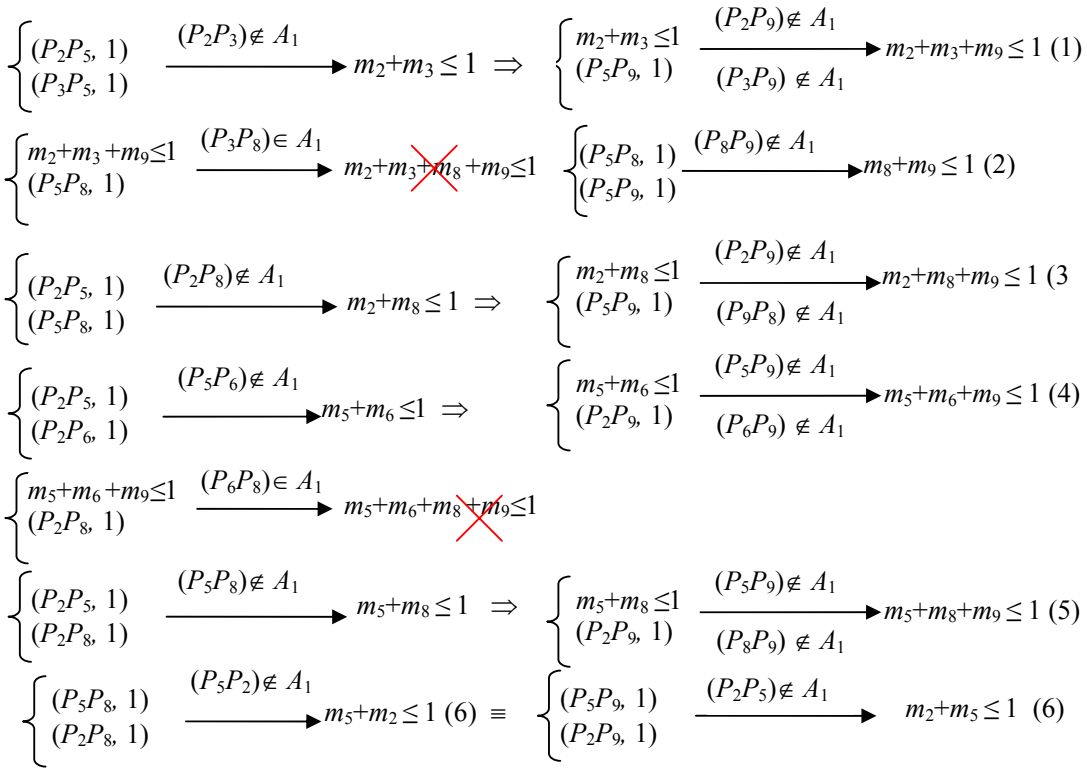


Fig. 4.2. Calcul des contraintes en utilisant les propriétés (4.2) et (4.3)

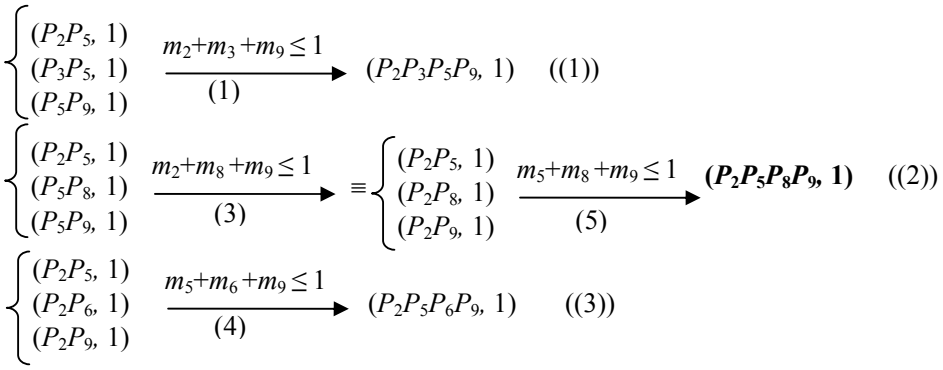


Fig. 4.3. Application de la propriété 3.2

$C_4 = \{((P_2P_3P_5P_9,\ 1), (P_2P_5P_8P_9,\ 1), (P_2P_5P_6P_9,\ 1)\}$

Il est possible de simplifier d'avantage les contraintes de la figure 4.3 en utilisant les propriétés 4.3 et 3.2. Finalement nous arrivons à 2 contraintes simplifiés comme la suit :

$C_5 = \{(P_2P_5P_8P_9,\ 1), (P_2P_3P_5P_6P_9,\ 1)\}$

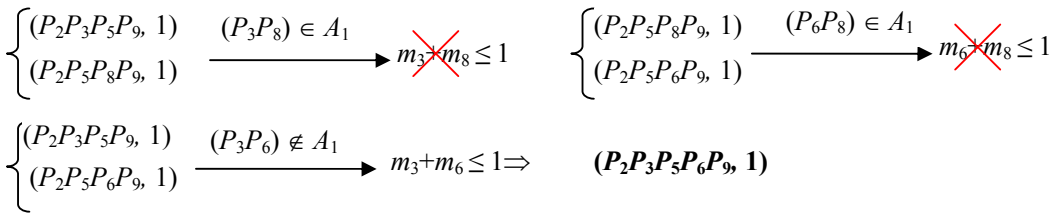


Fig. 4.4. Simplification au maximum par les propriétés 4.3 et 3.2

4.4 Le contrôleur maximal permissif

4.4.1 Le meilleur choix

La méthode pour la sélection finale à partir des contraintes simplifiées est identique à la méthode présentée pour les RdP conservatifs. L'ensemble final des contraintes simplifiées a été calculé.

$C_5 = \{(P_2P_5P_8P_9,\ 1), (P_2P_3P_5P_6P_9,\ 1)\}$

Pour la sélection finale, nous avons besoin des relations de couverture. Nous pouvons utiliser les définition 3.3 pour $R(M_i,c_j)$ et 3.4 pour $Cv(M_i,C_F)$.

Le tableau 4.1 donne les relations $R(M_i,c_j)$ et $Cv(M_i,C_F)$ pour notre exemple.

$\downarrow c_j$ $\quad M_i \rightarrow$	$P_2P_5P_7$	$P_3P_5P_8$	$P_1P_5P_9$	$P_2P_6P_8$	$P_2P_4P_9$	C_F
$(P_2P_5P_8P_9,\ 1)$	√	√	√	√	√	√
$(P_2P_3P_5P_6P_9,\ 1)$	√	√	√	√	√	-
$CV(M_i,C_F)$	√	√	√	√	√	

Tableau 4.1 Relations $R(M_i, c_j)$ et C_F et $Cv(M_i,C_F)$

Les deux contraintes couvrent toutes les états interdits. Donc nous avons deux solutions possibles. Après le calcul des places de contrôle, nous verrons que les résultats pour les deux solutions sont les mêmes.

$C_F = \{(P_2P_5P_8P_9,\ 1)\}$

Pour cet exemple, nous voyons que : $\forall\ M_i \in \mathcal{M}_{IF}\ \ Cv(M_i,C_F) = 1$

A partir des contraintes données par C_F on peut construire le contrôleur. Nous avons vu dans le chapitre 3 que ce contrôleur est toujours maximal permissif pour un RdP conservatif. On va voir dans la section suivante que ceci n'est pas toujours le cas pour un RdP non conservatif.

4.4.2 Problème des RdP non conservatifs

Les résultats du travail présenté par Giua (Giua et al. 1992) ont montré que l'ensemble des états interdits est équivalent à l'ensemble des contraintes linéaires lorsque le modèle RdP est conservatif. Le problème qui apparaît dans le cas de RdP non conservatif, c'est que la contrainte équivalente à un état interdit peut empêcher un état autorisé. Cela arrive dans le cas où l'ensemble des places marquées d'un état interdit est inclus dans l'ensemble des places marquées d'un état autorisé ou quand un sur-état interdit est un sur-état des états autorisés. Par exemple si $M_1 = P_2P_3P_5$ un état interdit et $M_2 = P_2P_3P_5P_7$ un état autorisé, la contrainte équivalente à l'état M_1 interdit l'état autorisé M_2. Dans la

construction de l'ensemble $\mathcal{B}_2$ (Section 4.3.1), nous supprimons tous les sur-états autorisés de l'ensemble des sur-états interdits. Donc dans ce cas, la contrainte équivalente à l'état M_1 sera supprimée de l'ensemble final. Ce problème est illustré dans l'exemple ci-dessous :

Exemple 4.1 : Prenons un système composé de deux machines Ma_1 et Ma_2. La machine Ma_1 peut tomber en panne, mais seule une panne est possible. La spécification impose de faire travailler les machines l'une après l'autre : Ma_1, Ma_2, Ma_1, et ainsi de suite. Les événements c_1 et c_2 sont contrôlables et les autres événements sont incontrôlables. Le modèle RdP de ce système est présenté dans la figure 4.5.

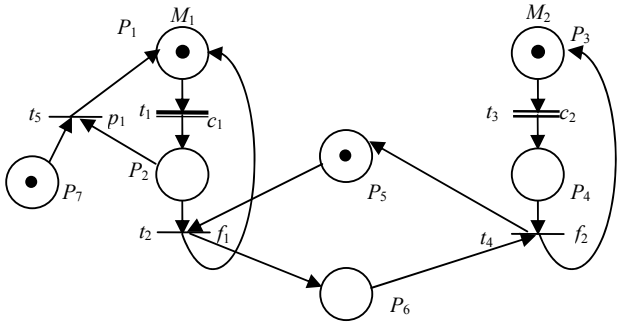


Fig. 4.5. Modèle RdP de l'exemple 4.1

Le graphe de marquage de ce modèle est donné dans la figure 4.6.

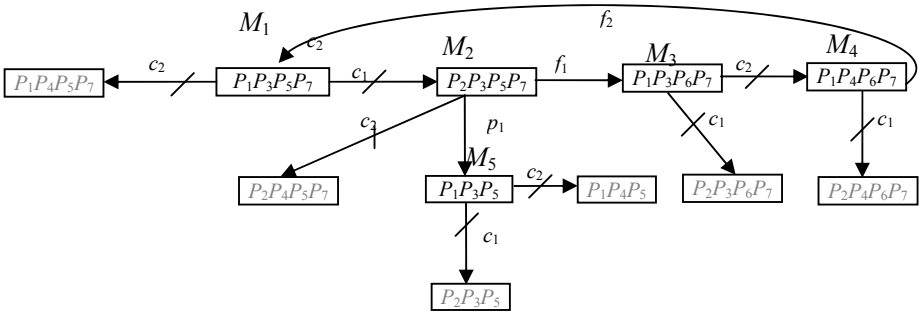


Fig. 4.6. Graphe des marquages accessibles

L'ensemble des états autorisés et l'ensemble des états interdits frontières sont :

$\mathcal{M}_A = \{P_1P_3P_5P_7, P_2P_3P_5P_7, P_1P_3P_6P_7, P_1P_4P_6P_7, P_1P_3P_5\}$

$\mathcal{M}_{IF} = \{P_1P_4P_5P_7, P_2P_4P_5P_7, P_2P_3P_6P_7P_2P_4P_6P_7, P_2P_3P_5, P_1P_4P_5\}$

L'ensemble des sur-états $\mathcal{B}_1$ sera calculé comme ci-dessous :

$\mathcal{B}_1 = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_1P_4, P_1P_5, P_1P_7, P_4P_5, P_4P_7, P_5P_7, P_2P_4, P_2P_5, P_2P_7, P_2P_3, P_2P_6, P_3P_6, P_3P_7, P_6P_7, P_4P_6, P_3P_5, P_1P_4P_5, P_1P_4P_7, P_1P_5P_7, P_4P_5P_7, P_2P_4P_5, P_2P_4P_7, P_2P_5P_7, P_2P_3P_6, P_2P_3P_7, P_2P_6P_7, P_3P_6P_7, P_2P_4P_6, P_2P_4P_7, P_2P_6P_7, P_4P_6P_7, P_2P_3P_5, P_1P_4P_5P_7, P_2P_4P_5P_7, P_2P_3P_6P_7, P_2P_4P_6P_7\}$

Nous calculons directement $\mathcal{B}_3$:

$\mathcal{B}_3 = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_1P_4, P_1P_5, P_1P_7, P_4P_5, P_4P_7, P_5P_7, P_2P_4, P_2P_5, P_2P_7, P_2P_3, P_2P_6, P_3P_6, P_3P_7, P_6P_7, P_4P_6, P_3P_5, P_1P_4P_5, P_1P_4P_7, P_1P_5P_7, P_4P_5P_7, P_2P_4P_5, P_2P_4P_7, P_2P_5P_7, P_2P_3P_6, P_2P_3P_7, P_2P_6P_7, P_3P_6P_7, P_2P_4P_6, P_2P_4P_7, P_2P_6P_7, P_4P_6P_7, P_2P_3P_5, P_1P_4P_5P_7, P_2P_4P_5P_7, P_2P_3P_6P_7, P_2P_4P_6P_7\} = \{P_4P_5, P_2P_4, P_2P_6\}$

L'ensemble des contraintes simplifiées final est :

$C_3 = \{(P_4P_5, 1), (P_2P_4, 1), (P_2P_6, 1)\}$

Le sur-état $P_2P_3P_5$ qui est un état interdit, a été supprimé de cet ensemble.

Le tableau 4.2 contient les relations $R(M_i,c_j)$ et $Cv(M_i,C_F)$.

c_j \ M_i	$P_1P_4P_5P_7$	$P_2P_4P_5P_7$	$P_2P_3P_6P_7$	$P_2P_4P_6P_7$	$P_2P_3P_5$	$P_1P_4P_5$	C_F
$(P_4P_5, 1)$	√	√	-	-	-	√	√
$(P_2P_4, 1)$		√		√			
$(P_2P_6, 1)$	-	-	√	√	-	-	√
$CV(M_i,C_F)$	√	√	√	√	-	√	

Tableau 4.2 Fonction $R(M_i, c_j)$ et C_F et $Cv(M_i,C_F)$ de exemple 4.1

Nous voyons que pour cet exemple l'état $P_2P_3P_5$ n'est couvert par aucune contrainte, donc : $Cv(P_2P_3P_5, C_F) = 0$, C'est la raison pour laquelle on ne peut pas construire le contrôleur maximal permissif.

Dans la section suivant nous généraliserons cette idée pour établir une condition nécessaire et suffisante pour avoir un contrôleur maximal permissif.

Ce type de comportement est rarement rencontré dans les systèmes réels. Nous l'avons construit artificiellement. Néanmoins, dans cet exemple nous pouvons modifier le modèle du système en ajoutant la place P_8 pour que l'état interdit ne soit pas un sur-état des états autorisés. Ce modèle est présenté dans la figure 4.7.

4.4.3 Condition d'optimalité

Pour avoir un contrôleur maximal permissif, il est nécessaire que l'ensemble des états autorisés ($\mathcal{M}_A$) soit égal à l'ensemble des états non interdit ($\mathcal{M}_{NI}$) qui vérifient l'ensemble des contraintes finales C_F. Cette égalité sera vraie lorsqu'il existe au moins une contrainte pour chaque état interdit. Ce ci est formalisé par la propriété ci-après :

Propriété 4.4 : Soit $\mathcal{M}_{NI}$ l'ensemble des états non interdits donné par l'ensemble des contraintes $\mathcal{C}_3$. L'ensemble $\mathcal{M}_{NI}$ est égal à l'ensemble $\mathcal{M}_A$ si et seulement si $\forall\ M_i \in \mathcal{M}_{IF}\ \ Cv(M_i, \mathcal{C}_3) = 1$. Le contrôleur synthétisé est maximal permissif.

Preuve:

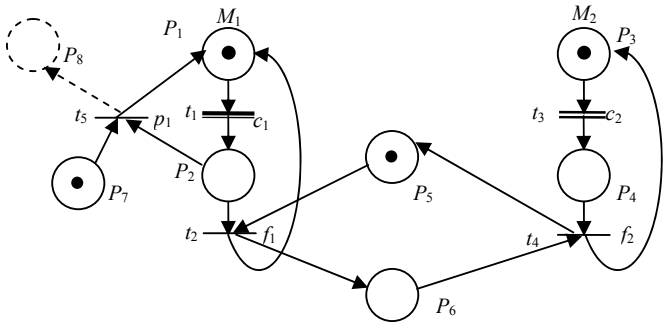


Fig. 4.7. Modèle RdP corrigé de l'exemple 4.1

Condition nécessaire :

Supposons $\exists M_i \in \mathcal{M}_{IF}$ $Cv(M_i, \mathcal{C}_3) = 0$ donc M_i n'est interdit par aucune contrainte et bien sûr $M_i \in \mathcal{M}_{NI,}$ mais M_i est un état interdit $\Rightarrow M_i \notin \mathcal{M}_A \Rightarrow \mathcal{M}_A \neq \mathcal{M}_{NI}$

Condition suffisante :

Si $\forall M_i \in \mathcal{M}_{IF}$ $Cv(M_i, \mathcal{C}_3) = 1$ signifie que $\forall M_i \in \mathcal{M}_{IF}$ $\exists c_j \in \mathcal{C}_3$ tel que c_j interdit M_i. Donc tous les états interdits sont non atteignables.

Maintenant nous devons monter qu'aucun état admissible n'est interdit.

Supposons $\exists M_k \in \mathcal{M}_A$ tel que M_k soit interdit, donc il faut que $\exists c_l \in \mathcal{C}_3$ tel que le marquage M_k ne vérifie pas c_l. Par la propriété de simplification 3.2, l'ensemble des contraintes $\mathcal{C}_3$ est équivalent à l'ensemble des contraintes déduites de $\mathcal{B}_3$. Donc il faut qu'il existe un sur-état interdit de l'ensemble $\mathcal{B}_3$ qui interdise M_k. Or nous avons supprimé de l'ensemble $\mathcal{B}_3$ tous les sur-états autorisés.

En conclusion, il n'y a aucun $M_k \in \mathcal{M}_A$ tel que M_k soit interdit et donc l'ensemble $\mathcal{M}_{NI}$ est égal à l'ensemble $\mathcal{M}_A$.

❑

Si $\forall M_i \in \mathcal{M}_{IF}$ $Cv(M_i, \mathcal{C}_3) = 1$ il est clair qu'en appliquant l'algorithme 3.2, on peut choisir l'ensemble minimal C_F tel que $\forall M_i \in \mathcal{M}_{IF,}$ $Cv(M_i, C_F) = 1$ et dans ce cas, l'ensemble $\mathcal{M}_{NI}$ sera identique à l'ensemble $\mathcal{M}_A$.

Pour l'exemple que nous traitons dans ce chapitre, nous voyons que dans le tableau 4.1, $\forall M_i \in \mathcal{M}_{IF}$ $Cv(M_i, C_F)) = 1$. Donc le contrôleur construit sera maximal permissif.

$C_F = \{(P_2P_5P_8P_9, 1)\}$

La méthode pour le calcul des places de contrôle à partir des contraintes simplifiées est identique à la méthode présentée pour les RdP conservatifs. Nous l'appliquons sur notre exemple dans la section suivante.

4.4.4 Calcul de la place de contrôle

En utilisant la méthode de calcul pour les places de contrôle présentée dans la section 3.4, nous calculons le contrôleur final.

$$W = \begin{bmatrix} W_R \\ W_c \end{bmatrix}$$

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

$$L^T = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$W_C = - L^{\mathrm{T}}.W_R$$

$$M_{c\_ini} = k \; - L^{\mathrm{T}}.M_{R\_\mathrm{ini}}$$

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

$$M_{cinit} \; = \; b \; - \; L^T.M_{pinit} \; = 1 - 0 = 1$$

Le modèle RdP final est représenté dans la figure 4.8.

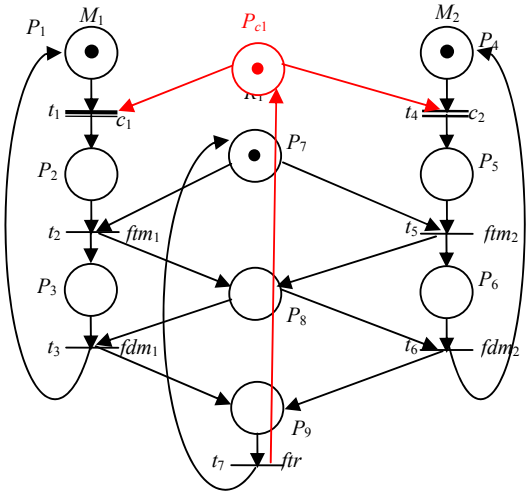


Fig. 4.8. Modèle RdP contrôlé

Comme nous pouvons le voir dans la figure 4.8, le modèle final est très simple comparé au modèle final déterminé dans le chapitre 3. Dans la conclusion nous discuterons des différences qu'il y a entre les deux méthodes sachant d'ores et déjà que celle présentée dans ce chapitre est plus générale.

4.5 Application sur un système manufacturier

Dans cette section, nous appliquons cette méthode à un système industriel d'assemblage constitué de convoyeurs et de postes robotisés. Notre objectif est de déterminer un contrôleur pour la boucle extérieure du système. Elle est composée d'un convoyeur et d'une station d'assemblage. 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 4.9).

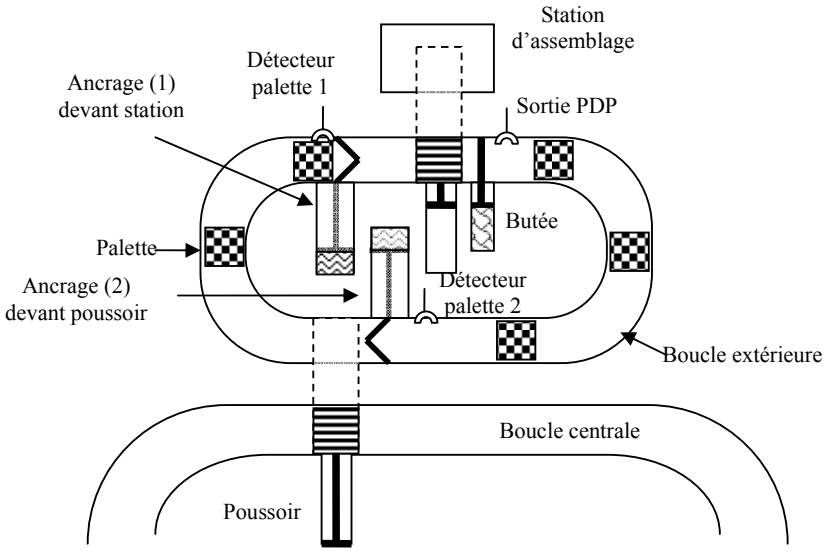


Fig. 4.9. Système d'assemblage

Nous avons choisi 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 3.

Les événements associés à chaque transition sont explicités au tableau 4.3.

Evénement	Action	Evénement	Action	Evénement	Action
DPA1	Détection palette devant Ancrage 1	OA1	Ouverture ancrage 1	APA1	Absence palette devant Ancrage 1
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		

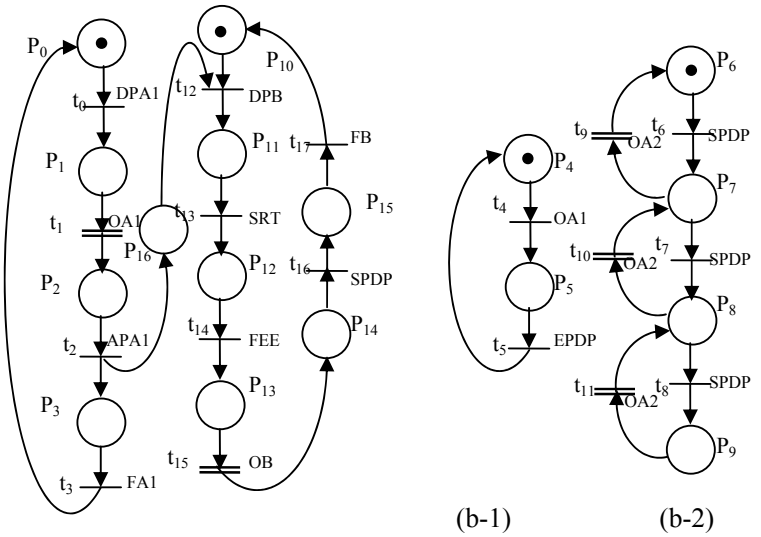
Tableau 4.3. Evénements associés aux transitions

Nous avons modélisé ce système par un modèle RdP sauf. L'ensemble des places $\{P_0, P_1, P_2, P_3\}$ présente la situation de la palette près de l'ancrage et l'ensemble $\{P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}\}$ celle de la butée. La place P_{16} présente la situation de la palette entre l'ancrage et la butée. Chaque place est présentée dans le tableau 4.4.

Nom de place	Description	Nom de Place	Description
P_0	Fermeture de l'ancrage 1 L'absence de palettes	P_9	Trois palettes entre butée et actuateur
P_1	Présence de palettes devant l'ancrage	P_{10}	Butée est fermé
P_2	Ouverture de l'ancrage	P_{11}	Présence de palette devant butée
P_3	Absence de palet lorsque l'ancrage est ouvert	P_{12}	Opération de lecture, assemblage et écriture.
P_4	Absence de palettes entre l'ancrage et butée	P_{13}	Fin d'assemblage
P_5	Une palette entre l'ancrage et butée	P_{14}	Butée est ouvert
P_6	Zéro palette entre butée et actuateur	P_{15}	Absence de palette après butée
P_7	Une palette entre butée et actuateur	P_{16}	Présence de palette entre l'ancrage et butée
P_8	Deux palettes entre butée et actuateur		

Tableau 4.4. Description de chaque place

Le modèle RdP de ce système avec ses spécifications est donné dans la figure 4.10. Le composé synchrone entre le modèle du système et celui des spécifications donne le RdP correspondant au comportement désiré en boucle fermée (figure 4.11).



a) Modèle du système b) Modèle des spécifications

Fig. 4.10. Modèle RdP de Système manufacturier et spécifications

L'événement incontrôlable SPDP (Sortie palette du PDP) entraîne l'existence d'états interdits. Les transitions t_6, t_7 et t_8 concernées par cet événement incontrôlable ne respectent pas les conditions de franchissement contraintes par les spécifications. Par exemple si la place P_{14} est marqué (la palette est en train de sortir de la butée) et qu'aucune place de l'ensemble $\{P_6, P_7, P_8\}$ ou place P_5 ne sont

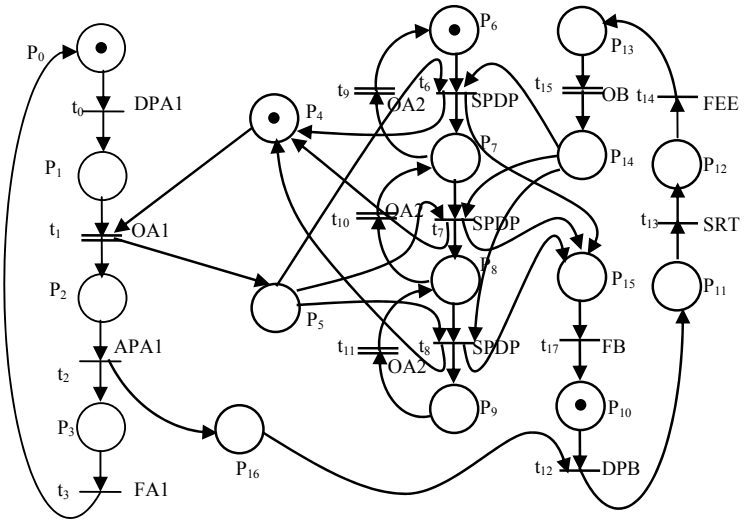


Fig. 4.11. Modèle RdP de Système manufacturier en boucle fermée

marquées, alors la palette passera devant le détecteur (l'événement SPDP). Donc les spécifications seront violées. Pour empêcher ce problème, on doit empêcher d'avoir un état dans lequel la place P_{14} est marquée et une des places de l'ensemble $\{P_6, P_7, P_8\}$ et la place P_5 ne sont pas marquées. Il y a trois possibilités pour cette situation et donc nous avons trois états interdits :

$\mathcal{M}_{IF} = \{P_0P_5P_9P_{14}, P_1P_5P_9P_{14}, P_3P_5P_9P_{14}\}$

Par application de la première propriété de simplification, on peut trouver le sur-état P_9P_{14} qui n'existe pas dans les sur – états des états autorisés et couvre tous les états interdits. Donc si nous utilisons la contrainte équivalente à ce sur-état, on peut construire un contrôleur maximal permissif.

La contrainte correspondante à ce sur-état est :

$$m_9+m_{14} \leq 1$$

En utilisant la méthode présentée pour les RdP conservatif, nous obtenons le même résultat.

Par la méthode de Yamalidou nous pouvons calculer la place de contrôle. Le résultat final est présenté dans la figure 4.12.

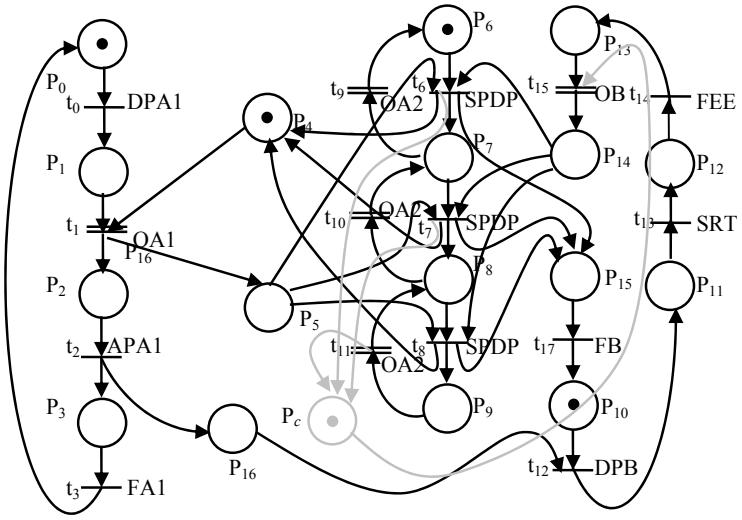


Fig. 4.12. Modèle RdP final de Système manufacturier

Le modèle RdP n'est pas directement applicable sur le API. Mais on peut transférer le modèle RdP à un modèle SFC ou Grafcet applicable sur le API. Cette transformation est présentée dans la figure 4.13. Les opérations *A, B, C, D, E* correspondent, respectivement, aux opérations dans les places P_0, P_2, P_{10} , P_{12}, P_{14} selon la description de ces places dans le tableau 4.4,.

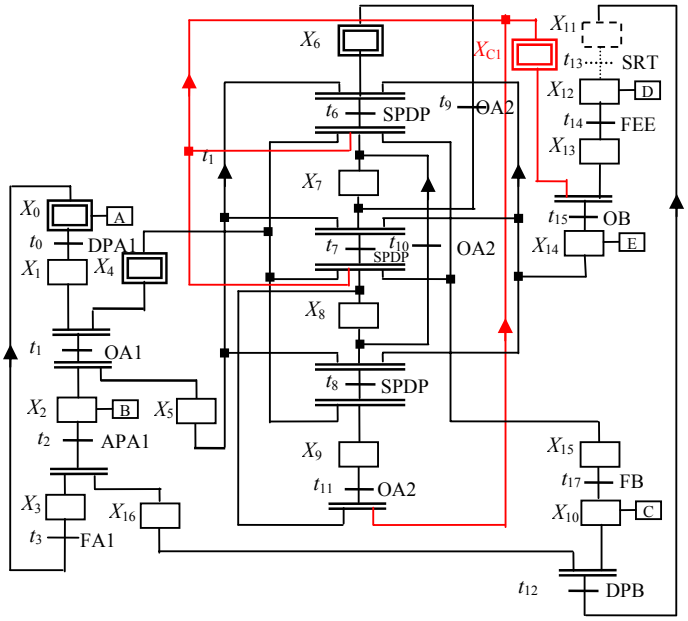


Fig. 4.13. Modèle Grafcet de RdP Figure 4.12

Dans le modèle Grafcet de la figure 4.13, il y a une transition et une étape que l'on peut supprimer. L'événement contrôlable *SRT* « début de lecture d'étiquette », n'est pas été utilisé pour le contrôle et donc on peut supprimer la transition t_{13} et l'étape X_{11}. Cela signifie que si un palet est détecté devant de butée (DPB), l'opération de lecture d'étiquette peut commencer immédiatement.

4.6 Conclusion

Nous avons présenté une méthode systématique pour réduire le nombre de contraintes linéaires correspondants aux états interdits. Ceci est réalisé en utilisant intuitivement les états non atteignables et en construisant de manière systématique des contraintes sur les marquages interdits par la détermination de sur-états. Ces sur-états correspondent à un ensemble de marquages ayant une même propriété (interdits ou autorisés). Des propriétés générales ont été établies et illustrées sur un exemple simple et ensuite sur un système d'assemblage. Les propriétés donnant des conditions nécessaires et suffisantes d'existence d'un contrôleur maximal permissif ont été établies. Grâce aux simplifications réalisées, ce contrôleur est de taille réduite. Lorsque celui-ci existe, il est possible de le déterminer de manière simple par la technique des invariants.

Dans ce chapitre nous avons présenté une deuxième méthode de simplification des contraintes linéaires. Alors que la première méthode introduite dans le chapitre précédent est applicable seulement sur les RdP saufs et conservatifs. La deuxième méthode est applicable pour les RdP saufs mais pas nécessairement conservatifs. Le premier exemple présenté dans ce chapitre, montre qu'il est possible d'obtenir avec cette méthode, un résultat plus réduit et plus simple. La comparaison des résultats obtenus par l'application de cette méthode sur l'ensemble des exemples présentés dans la section 3.5, montre qu'en moyenne, il n y a pas de différences importantes entre les résultats des deux méthodes (tableau 4.5).

CAS	Nombre d'états interdits	Nombre de contraintes finales par méthode 1	Nombre de contraintes finales par méthode 2	Borne de la contrainte primaire	Borne de la contrainte finale par méthode 1	Borne de la contrainte finale par méthode 2
1	5	3	1	2	1	1
2	5	2	2	2	1	1
3	6	1	3	3	2	2
4	5	5	5	1	1	1
5	12	1	1	1	1	1
6	3	1	1	3	1	1
Total	36	13	13	12	7	7

Tableau 4.5 : Réduction du nombre des contraintes pour les différents cas pour les méthodes

La méthode présentée dans ce chapitre reste néanmoins plus intéressante car elle en nécessite pas d'hypothèse de RdP conservatif.

Chapitre 5

Synthèse de contrôleur par retour d'état

Dans ce chapitre nous allons présenter une nouvelle méthode de synthèse de contrôleur par retour d'état. Dans cette méthode nous ajoutons des conditions aux transitions contrôlables pour empêcher l'atteignabilité des états interdits. Par les méthodes de réduction que nous avons présentées dans les chapitres précédents, nous pouvons simplifier ces conditions et construire un contrôleur maximal permissif.

Chapitre 5

Synthèse de contrôleur par retour d'état

5.1 Introduction

Dans les méthodes de synthèse de contrôleurs présentées dans les chapitres 3 et 4, le contrôleur a été déterminé par l'ajout de places. Dans ces approches le contrôleur est obtenu par une modification de la structure du modèle. Il est possible d'utiliser une autre méthode de synthèse de contrôleur s'inspirant des RdP interprétés de commande (RdPIC). Au lieu d'ajouter des places de contrôle, nous concevons une commande pour chaque transition contrôlable. Cette commande peut autoriser ou interdire le franchissement de cette transition. Nous verrons qu'il est parfois possible d'obtenir un résultat plus simple par cette méthode. La structure du modèle RdP contrôlé par cette méthode est présentée dans la figure 5.1.

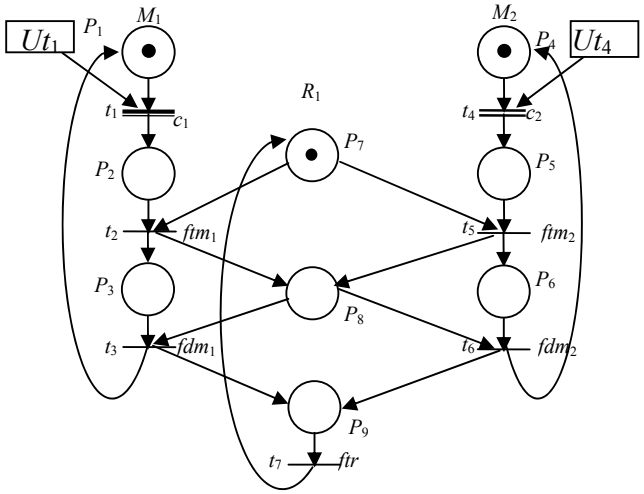


Fig. 5.1. Modèle RdP avec le contrôle

De nombreuses approches utilisent cette méthode pour empêcher des états interdits, en général en forme de Contraintes Généralisées d'Exclusion Mutuelles (CGEM) (Holloway et al. 1990), (Boel et al. 1995), (Holloway et al. 1996), (Ghaffari et al. 2003). L'avantage de ces approches, est qu'elles n'ont pas besoin de construire le graphe de marquage. Cependant l'inconvénient majeur de ces approches est que le temps de calcul du contrôle en temps réel est long et que le modèle final est complexe.

Dans ce chapitre nous présenterons une novelle méthode de synthèse de contrôleurs (Dideban et al. 2006a,b). Dans cette méthode nous ajoutons des conditions pour les transitions contrôlables. L'avantage de notre contribution comparée aux approches indiquées plus haut, c'est que nous obtenons un contrôleur plus simple et donc la complexité de modèle est diminué. L'inconvénient est de construire le graphe de marquages, cependant ce calcul se fera hors ligne. Pour présenter cette méthode de synthèse de contrôleur, nous présentons d'abord l'idée de base et ensuite nous calculons le contrôleur, puis nous le simplifions. Nous illustrerons cette idée sur l'exemple du système industriel qui a été présenté dans le chapitre 4. A la fin de ce chapitre, nous comparerons cette méthode de synthèse avec les méthodes présentées dans les chapitres précédents.

5.2 Contrôle par retour d'état

5.2.1. L'idée principale

Notre objectif est d'empêcher des états interdits par le contrôle des transitions contrôlables. Pour empêcher des états interdits, il faut connaître l'ensemble des états à partir desquels les états interdits sont atteignables. Cet ensemble a été déjà défini comme des états autorisés critiques dans la définition 2.7. Nous reprenons l'exemple 2.1 dont le modèle RdP est présenté dans la figure 5.1. L'ensemble des états critiques est présenté dans la figure 5.2.

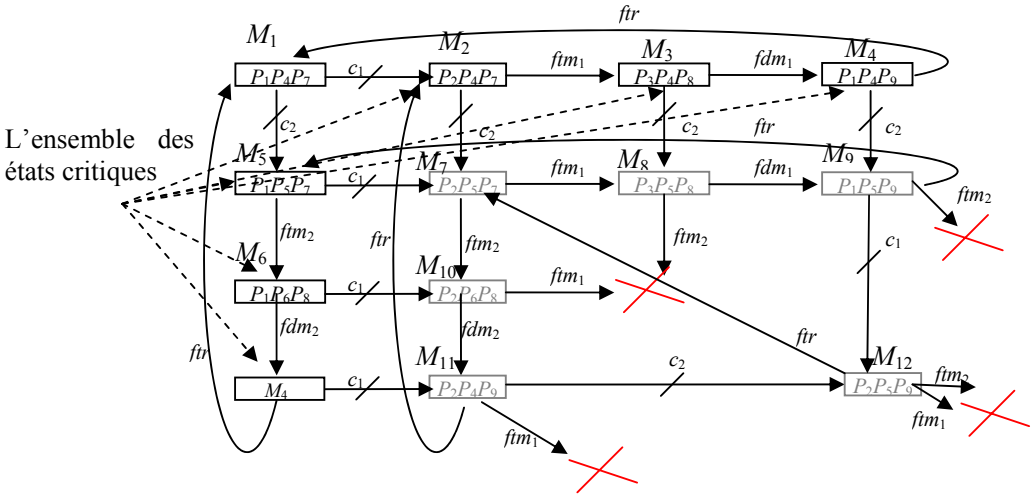


Fig. 5.2. L'ensemble des états critiques

Dans cet ensemble, pour chaque transition contrôlable nous avons un sous-ensemble des états critiques. Il y a deux transitions contrôlables et donc deux sous-ensembles :

$$\mathcal{M}_{t1}{}^{C} = \{P_1P_5P_7, P_1P_6P_8, P_1P_4P_9\}$$
$$\mathcal{M}_{t4}{}^{C} = \{P_2P_4P_7, P_3P_4P_8, P_1P_4P_9\}$$

Pour chaque transition contrôlable t_i, il faut créer un contrôle Ut_i, tel que le franchissement de la transition t_i dans l'ensemble des états critiques $\mathcal{M}_{AC}{}^{ti}$ soit interdit.

A partir de l'ensemble des états critiques, on peut calculer la condition de franchissement. Par exemple à partir de l'ensemble $\mathcal{M}_{t1}{}^{C} = \{P_1P_5P_7, P_1P_6P_8, P_1P_4P_9\}$, la condition de franchissement de t_1 exige que cette transition ne soit pas franchie à partir de tout marquage de cet ensemble. Cette condition peut être calculée comme ci – dessous :

Condition de franchissement $t_1 = (m_1m_5m_7 + m_1m_6m_8 + m_1m_4m_9)'$

Le symbole « ' » signifie le complément logique. Cette condition sera d'autant plus compliquée que le nombre d'états critiques pour chaque transition augmente.

Notre objectif dans ce chapitre est de présenter une méthode de synthèse de contrôleurs simple et efficace.

5.2.2 Calcul du contrôle

En utilisant l'algorithme de Kumar, on peut calculer l'ensemble des états interdits frontières $\mathcal{M}_{IF}$. A partir de cet ensemble il est possible de calculer l'ensemble des états autorisés critiques $\mathcal{M}_{AC}$ selon la définition 2.7. Dans l'ensemble des états autorisés critiques, on peut préciser les sous-ensembles $M_{t_i}^C$ pour toutes les transitions contrôlables t_i. Les états critiques pour une transition t_i ($M_{t_i}^C$) sont les états tels que le franchissement de cette transition conduit à un état interdit.

Définition 5.1 : Soit $\mathcal{M}_A$ l'ensemble des états autorisés et $\mathcal{M}_{IF}$ l'ensemble des états interdits frontières. L'ensemble des *états critiques* pour la transition contrôlable $t_i \in \sum_c$ est défini comme suit :

$$M_{t_i}^C = \{m_i \in \mathcal{M}_A \mid m_i \xrightarrow{t_i} m_j \quad , m_j \in \mathcal{M}_{IF}\}$$

❑

A l'opposé des états critiques, nous avons les états sains. Les états sains pour une transition contrôlable t_i ($M_{t_i}^S$) sont les états tels que le franchissement de cette transition conduit à un état autorisé.

Définition 5.2 : Soit $\mathcal{M}_A$ l'ensemble des états autorisés. L'ensemble des *états sains* pour la transition contrôlable $t_i \in \sum_c$ est défini comme suit :

$$M_{t_i}^S = \{m_i \in \mathcal{M}_A \mid m_i \xrightarrow{t_i} m_j, m_j \in \mathcal{M}_A\}$$

❑

Mais comment peut-on contrôler une transition pour qu'elle soit franchissable dans l'ensemble des états sains et ne le soit pas dans l'ensemble des états critiques ?

Il y a deux possibilités : soit on peut interdire le franchissement seulement à partir des états critiques ou soit on autorise le franchissement seulement dans les états sains.

Définition 5.3 : Soit $M_{t_i}^C$ l'ensemble des états critiques pour transition t_i et $M_{t_i}^S$ l'ensemble des états sains pour cette transition. Le contrôle $Ut_i(M_j) \rightarrow \{0,1\}$ est défini comme ci-dessous :

$$U_{t_i}(M_j) = \begin{cases} 0 & M_j \in M_{t_i}^C \\ 1 & \text{Si non} \end{cases} \quad \text{ou} \quad U_{t_i}(M_j) = \begin{cases} 1 & M_j \in M_{t_i}^S \\ 0 & \text{Si non} \end{cases}$$

❑

La relation entre le contrôle et le modèle du système est présentée dans la figure 5.3.

Soit $M_j = P_{j1}P_{j2}\ldots P_{jn}$ un état critique pour la transition t_i. Dans notre approche, nous considérons des RdP saufs. Donc le nombre de marques dans chaque place (m_{jr}) est une variable Booléenne. Par conséquent on peut calculer le contrôle $Ut_i(M_k)$ à l'aide de l'expression logique *con_*M_j :

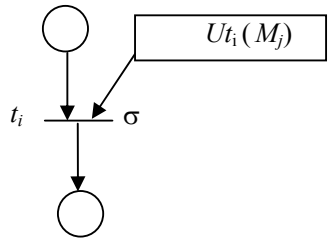


Fig. 5.3. Contrôle empêchant l'atteignabilité des états interdits

$$con\_M_j = m_{j1}m_{j2} \ \ldots \ m_{jr} \ \ldots \ .m_{jn} = \bigcap_{r=1}^{n} m_{jr}$$

Le variable Booléenne con_M_j est égal à 1 lorsque l'état actuel de système est l'état M_j. Le complémentaire de cette variable pour la transition t_i, donne le contrôle $Ut_i(M_k)$ pour cette transition comme présenté dans la figure 5.4. c_i est un événement contrôlable.

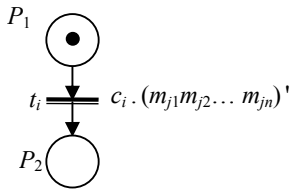


Fig. 5.4. Contrôle d'une transition à partir d'un état critique

Nous présentons cette idée par la propriété suivante :

Propriété 5.1: Soit $M_j = P_{j1}P_{j2}\ldots P_{jn}$ un état critique pour une transition t_i. En utilisant le contrôle $Ut_i(M_k)$, on peut interdire l'atteignabilité de cet état :

$$U_{t_i}(M_k) = (\bigcap_{r=1}^{n} m_{jr})'$$

Preuve :

Supposons que le système est dans l'état M_j.

$$Ut_i(M_j) = (m_{j1}m_{j2}m_{j3}\ldots m_{jn})` = (\bigcap_{r=1}^{n} m_{jr})' = 0$$

Donc la transition contrôlée par $Ut_i(M_k)$ n'est pas franchissable. En d'autres termes, les états interdits concernés ne sont pas atteignables.

❑

En général le nombre d'états critiques pour une transition est souvent supérieur à 1. Dans ce cas, on peut calculer le contrôle $Ut_i(M_k)$ selon la propriété 5.2.

Propriété 5.2 : Soit $M_{t_i}^{C}$, l'ensemble des états critiques de la transition t_i, le contrôle $Ut_i(M_k)$ est calculé comme suit :

$$Ut_i(M_k) = (\bigcup_{i=1}^{card(M_{t_i}^C)} (\bigcap_{r=1}^{card(support(M_j))} m_r))' \qquad (m_r \in \text{support}(M_j), M_j \in M_{t_i}^C)$$

❑

Les opérations $\bigcup$ et $\bigcap$ sont les opérations Booléennes sur les variables m_i.

Cette propriété correspond à l'union des conditions pour chaque état.

Par l'utilisation du contrôle $Ut_i(M_k)$, on peut interdire le franchissement de t_i à partir de l'ensemble $M_{t_i}^C$.

Le problème principal de cette méthode est qu'en général la condition calculée est complexe. Pour résoudre ce problème nous présentons une méthode de simplification proche de la méthode de simplification des contraintes linéaires. Nous présenterons ensuite une condition nécessaire et suffisante pour avoir un contrôleur maximal permissif. Une approche duale consiste à autoriser le franchissement de t_i dans les états sains. Elle sera présentée dans la section 5.4.

5.3 Simplification du contrôle

La simplification des contrôles est comme celle des contraintes. Nous pouvons utiliser deux méthodes proches des méthodes présentées pour les RdP conservatifs et pour les RdP saufs en cas général.

Les règles générales pour la simplification sont les règles de la simplification d'expressions logiques. Mais en plus de ces règles, nous utilisons également les équations d'invariants. Cela peut aider pour avoir plus de simplification.

Dans un RdP, pour le franchissement d'une transition, en plus de la condition logique qui doit être vraie, la transition doit être validée par les places en amont. Ceci peut aider pour plus de simplification en prenant en compte l'ensemble des états indifférents. La méthode de simplification que nous utilisons est proche de la méthode de simplification des sur-états.

5.3.1 Simplification du contrôle en utilisant la notion de sur-état

Comme nous avons dit, il est possible de simplifier le contrôle en utilisant les règles de simplification d'expressions logiques. Mais l'idée de sur-état, nous permet d'avoir une simplification plus efficace.

Soit $M_j = P_{j1}P_{j2}\ldots P_{jn}$ un état critique pour la transition t_i, le contrôle $Ut_i(M_k) = (m_{j1}m_{j2}m_{j3}\ldots m_{jn})'$ empêche le franchissement de cette transition pour l'état M_i. Il est évident que le contrôle $Ut_i(M_k) = (m_{j1})'$ peut également empêcher le franchissement de cette transition pour l'état M_j. Comme celui pour l'état on peut définie le contrôle équivalent à un ou un ensemble de sur-états.

Corollaire 5.1: Pour chaque sur-état $M_j = P_{j1}P_{j2} \ldots P_{jr} \ldots P_{jn}$ et un ensemble des sur-états $C_i = \{M_1, M_2, \ldots, M_m\}$, le contrôle concerné sera comme ci-dessous :

$$Ut(M_k) = (m_{j1}m_{j2}m_{j3}\ldots m_{jn})'$$

$$Ut(M_k) = (\bigcup_{j=1}^{card(C_i)} (\bigcap_{r=1}^{card(sup\,port(M_j))} m_r))' \qquad (m_r \in sup\,port(M_j), M_j \in C_i)$$

❑

$Ut(M_k)$ est calculé par le complémentaire de la condition d'interdiction.

En utilisant le contrôle concerné à chaque sur-état, on peut l'interdire, mais il est possible que par ce contrôle les états autorisés deviennent non accessibles. Le problème c'est que le contrôle déduit de

sur-état peut interdire le franchissement de cette transition même pour les états sains. Donc on peut utiliser un contrôle simplifié lorsqu'il n'interdit pas le franchissement de cette transition pour les états sains. C'est la même idée qui a été exploitée dans le chapitre 4. Pour résoudre ce problème on supprime de l'ensemble de sur-états des états critiques, des sur-états qui existent dans l'ensemble des sur-états des états sains.

Soit C_1^t l'ensemble des sur-états des états critiques et S_1^t l'ensemble des sur-états des états sains, le contrôle concerné par l'ensemble $C_2^t = C_1^t \setminus S_1^t$ interdit le franchissement de la transition t en excluant les états sains.

Dans cet ensemble C_2^t, il y a souvent des états redondants qu'il faut supprimer Cela arrive quand des sur-états sont couverts par d'autres.

En général le nombre d'états critiques pour une transition est supérieur à 1. Il faut donc choisir le sur-état de contrôle qui interdit le franchissement de la transition pour le maximum d'états. Une méthode de choix final semblable à celle déjà utilisée pour la simplification des contraintes sera appliquée.

L'algorithme 5.1 permet de faire la synthèse d'un contrôleur basée sur cette méthode de simplification.

Algorithme 5.1 :

Soit G un graphe de marquage représenté par un 4-tuplet (M [M_A, M_{IF}], $\Sigma[\Sigma_c, \Sigma_u]$, δ, M_0).

Pas 1 : Pour $\forall\, t \in \Sigma_c$ calculer l'ensemble M_t^C et M_t^S à partir de l'ensemble M_A, M_{IF}.

Pas 2 : Pour chaque transition t, calculer l'ensemble des sur-états C_1 et S_1 pour l'ensemble M_t^C et M_t^S.

Pas 3 : Calculer la condition d'interdiction, C_2 : $C_2 = C_1 \setminus S_1$

Pas 4 : Construire C_3 en supprimant les termes redondants de l'ensemble C_2.

Pas 5 : Construire C_4 en appliquant l'algorithme de choix final identique à celui présenté dans la section 3.4.2.

Pas 6 : Calculer les contrôles concernés pour chaque ensemble de sur-états selon la corollaire 5.1

Pas 7 : Fin

❑

Appliquons cet algorithme à notre exemple. Nous avons déjà calculé l'ensemble des états critiques pour les transitions contrôlables t_1 et t_4, ce qui correspond au pas 1 :

$$\mathcal{M}_{t1}{}^{C} = \{P_1P_5P_7,\ P_1P_6P_8,\ P_1P_4P_9\}$$
$$\mathcal{M}_{t4}{}^{C} = \{P_2P_4P_7,\ P_3P_4P_8,\ P_1P_4P_9\}$$

Les ensembles des états sains pour ces transitions sont :

$$\mathcal{M}_{t1}{}^{S} = \{P_1P_4P_7\}$$
$$\mathcal{M}_{t4}{}^{S} = \{P_1P_4P_7\}$$

Pas 2 : soit C_1^{t1}, l'ensemble des sur-états des états critiques de la transition t_1 et S_1^{t1}, l'ensemble des sur-états des états sains.

$C_1^{t1} = \{P_1, P_5, P_7, P_6, P_8, P_4, P_9, P_1P_5, P_1P_7, P_5P_7, P_1P_6, P_1P_8, P_6P_8, P_1P_4, P_1P_9, P_4P_9, P_1P_5P_7, P_1P_6P_8, P_1P_4P_9\}$

$S_1^{t1} = \{P_1, P_4, P_7, P_1P_4, P_1P_7, P_4P_7, P_1P_4P_7\}$

Pas 3 : à partir de l'ensemble $C_2^{t1} = C_1^{t1} \setminus S_1^{t1}$, nous pouvons calculer le contrôle qui garantit l'interdiction de cette transition dans les états critiques.

$C_2^{t1} = C_1^{t1} \setminus S_1^{t1} = \{P_5, P_6, P_8, P_9, P_1P_5, P_5P_7, P_1P_6, P_1P_8, P_6P_8, P_1P_9, P_4P_9, P_1P_5P_7, P_1P_6P_8, P_1P_4P_9\}$

Pas 4 : cependant dans l'ensemble C_2^{t1} il y a des sur-états redondants. Par exemple les sur-état P_5 et P_1P_5 constituent une redondance, parce que si la condition relative à P_1P_5 (m_1m_5) est égale à un, automatiquement la condition concernée à P_5 sera aussi égale à un. En supprimant ces termes nous obtenons :

$C_3^{t1} = \{P_5, P_6, P_8, P_9\}$

Deux questions principales demeurent: Est-ce que le contrôle sera maximal permissif ? Et deuxièmement est-ce qu'il est nécessaire d'utiliser tous l'ensemble C_3^{tj}, pour le contrôle de cette transition?

Nous donnons les réponses à ces questions dans la section suivante et nous pourrons ainsi exécuter les pas 5 et 6 de l'algorithme.

5.3.2 Le contrôle maximal permissif

Pour répondre aux questions posées dans la section précédente, nous présentons d'abord la définition de la relation de couverture ci-dessous :

Définition 5.4 : Soit c_k un sur-état de l'ensemble des sur-états simplifiées $C_3^{ti} = \{c_1, \ldots, c_k, \ldots, c_m\}$ et M_j un état de l'ensemble des états autorisés critiques de transition t_i $\mathcal{M}_{ti}^{C} = \{M_1, \ldots, M_j, \ldots, M_n\}$. Les relations Rt_i: $\mathcal{M}_{ti}^{C} \times C_3^{ti} \rightarrow \{0, 1\}$ et CVt_i: $\mathcal{M}_{ti}^{C} \times C_3^{ti} \rightarrow \{0, 1\}$ sont définies comme suit :

$$Rt_i(M_j, c_k) = \begin{cases} 1 & \text{Si } c_k \leq c_j \ (c_k \text{est sur-état de } c_j) \\ 0 & \text{Sinon} \end{cases}$$

Où c_j est la contrainte équivalente à M_j.

$$CVt_i(M_j, C_3^{t_i}) = \bigcup_{k=1}^{k=m} Rt_i(M_j, c_k)$$

❑

Les relations $Rt_1(M_j, c_k)$ et $CVt_1(M_j, C_3^{t1})$ pour notre exemple sont présentées dans le tableau 5.1.

c_k ▼ M_j ▶	$P_1P_5P_7$	$P_1P_6P_8$	$P_1P_4P_9$
P_8	0	1	0
P_5	1	0	0
P_6	0	1	0
P_9	0	0	1
$CVt_1(M_j, C_3^{t1})$	1	1	1

Tableau 5.1 Relations $Rt_1(M_j, c_k)$ et $CVt_1(M_j, C_3^{t1})$

Nous voyons que pour cet exemple :

$$\forall M_j \in \mathcal{M}_{t1}^{C} \Rightarrow CVt_1(M_j, C_3^{t1}) = 1$$

Nous montrons ci-dessous que cette condition est nécessaire et suffisante pour avoir un contrôle maximal permissif. Appelons $\mathcal{M}_{NI}$ l'ensemble des états accessibles (non interdits) par application ce contrôle.

Propriété 5.3 : L'ensemble des contrôles $Ut_i(C_3^{ti}, M_k)$ pour $\forall\ t_i \in \sum_c$ donne un contrôleur maximal permissif, ou ce qui est équivalent, l'ensemble des états non interdits $\mathcal{M}_{NI}$ est égal à l'ensemble des états autorisés $\mathcal{M}_A$, si et seulement si :

$\forall\ t_i \in \sum_c, \forall\ M_j \in \mathcal{M}_{ti}^{C}\ \ CVt_i(M_j, C_3^{ti}) = 1$

Preuve :

Condition nécessaire :

Si $\mathcal{M}_{NI} = \mathcal{M}_A$, montrons que cela implique : $\forall\ t_i \in \sum_c, \forall\ M_k \in \mathcal{M}_{ti}^{C}\ \ CVt_i(M_k, C_3^{ti}) = 1$.

Supposons qu'il existe un état M_k, tel que $CVt_i(M_k, C_3^{ti}) = 0 \Rightarrow$

Selon la corollaire 5.1 pour l'ensemble C_3^{ti} : $\quad Ut_i(M_k) = \left(\bigcup_{j=1}^{card(C_3^{ti})} \left(\bigcap_{r=1}^{card(support(M_j))} m_r\right)\right)' = 1 \Rightarrow$

$M_k\ (M_k \in \mathcal{M}_{ti}^{C}) \xrightarrow{t_i} M_k'\ \ (M_k' \in \mathcal{M}_{IF}) \Rightarrow$

Donc un état $M_k' \in \mathcal{M}_{IF}$ sera accessible et bien évidement l'ensemble des états non interdits par ce contrôle ($\mathcal{M}_{NI}$) n'est pas égal à l'ensemble des états autorisés $\mathcal{M}_A$.

Condition suffisante :

Si $\forall\ t_i \in \sum_c\ , \forall\ M_k \in \mathcal{M}_{AC}^{ti}\ \ CVt_i(M_k, C_3^{ti}) = 1$, montrons que cela implique $\mathcal{M}_{NI} = \mathcal{M}_A$.

Supposons qu'il existe un état M_k, tel que :

$M_k \in \mathcal{M}_{NI}$ et $M_k \notin \mathcal{M}_A \Rightarrow \exists\ M_k' \in \mathcal{M}_{ti}^{C}$ tel que $M_k' \xrightarrow{t_i} M_k \Rightarrow$

$Ut_i(M_k') = 1 \Rightarrow$

$Ut_i(M_k') = \left(\bigcup_{j=1}^{card(C_3^{ti})} \left(\bigcap_{r=1}^{card(support(M_j))} m_r\right)\right)' \Rightarrow \bigcup_{j=1}^{card(C_3^{ti})} \left(\bigcap_{r=1}^{card(support(M_j))} m_r\right) = 0 \Rightarrow$

Donc dans l'état M_k' : $\forall\ M_j \in C_3^{ti}\ \left(\bigcap_{r=1}^{card(support(M_j))} m_r\right) = 0 \Rightarrow$

M_k' n'a pas été couvert par aucune $M_j \in C_3^{ti}$ et :

$CVt_i(M_k', C_3^{ti}) = 0$, ce qui n'est pas.

❑

Par cette propriété, nous avons obtenu la condition nécessaire et suffisante pour avoir un contrôleur maximal permissif. Il reste à répondre à la deuxième question concernant la nécessité d'utiliser tout l'ensemble C_3^{ti}, pour le contrôle de la transition t_i.

La réponse de cette question sera donnée par l'algorithme de sélection final qui a déjà été présenté pour l'ensemble des contraintes dans la section 3.4.2. Par application de cet algorithme sur notre exemple (Pas 5 de l'algorithme 5.1), on obtient l'ensemble C_F^{ti} (voir tableau 5.2.).

L'ensemble C_F^{t1} est calculé comme suit :

$C_F^{t1} = \{P_5, P_6, P_9\}$

Selon le corollaire 5.1, le contrôle de cette transition sera (Pas 6 de l'algorithme 5.1):

$c_i \downarrow \quad M_j \rightarrow$	$P_1P_5P_7$	$P_1P_6P_8$	$P_1P_4P_9$	C_F^{t1}
P_8	0	1	0	-
P_5	1	0	0	√
P_6	0	1	0	√
P_9	0	0	1	√
$CVt_1(M_j, C_F^{t1})$	√	√	√	

Tableau 5.2 Sélection l'ensemble final C_F^{t1}

$Ut_1(M_k) = (m_5 + m_6 + m_9)'$

Pour répondre exactement aux mêmes objectifs, Il est possible de calculer la condition de franchissement par une méthode duale à celle présentée dans cette section.

5.4 Simplification en utilisant la méthode duale

5.4.1 Calcul de la condition de franchissement

Dans la méthode de synthèse de contrôle que nous avons présentée ci-dessus, nous avons calculé la condition nécessaire à l'interdiction de franchissement à partir des états critiques. Nous savons qu'une transition ne peut être franchie que dans les états sains. Il est alors possible de calculer la condition de franchissement de cette transition comme suit :

$$Ut_i(M_k) = \bigcup_{j=1}^{card(M_{t_i}^S)} \left(\bigcap_{r=1}^{card(support(M_j))} m_r \right) \qquad (m_r \in support(M_j), M_j \in M_{t_i}^S)$$

Comme pour la condition d'interdiction on peut définit le contrôle concerné pour chaque sur-état puis pour un ensemble des sur-états comme suit :

Corollaire 5.2 : Pour chaque sur-état $M_j = P_{j1}P_{j2} \ldots P_{jr} \ldots P_{jn}$ et un ensemble des sur-états $S_i = \{M_1, \ldots, M_j, \ldots, M_m\}$, le contrôle concerné sera :

$$Ut(M_k) = (m_{j1}m_{j2}m_{j3}\ldots m_{jn})$$

$$Ut(M_k) = \left(\bigcup_{j=1}^{card(S_i)} \left(\bigcap_{r=1}^{card(support(M_j))} m_r \right) \right)$$

❑

Il est possible que cette condition soit plus simple que la condition d'interdiction. Les méthodes de calcul et de simplification de cette condition sont identiques à celles présentées dans la section 5.3. Il suffit de permuter les deux concepts « critique » et « sains ».

L'algorithme 5.2 permet de faire la synthèse d'un contrôleur basée sur cette idée.

Algorithme 5.2 :

Soit G un graphe de marquage représenté par un 4-tuplet (M [M_A, M_{IF}], $\Sigma[\Sigma_c, \Sigma_u]$, δ, M_0).

Pas 1 : Pour $\forall\, t \in \Sigma_c$ calculer l'ensemble M_t^C et M_t^S à partir de l'ensemble M_A, M_{IF}.

Pas 2 : Pour chaque transition *t*, calculer l'ensemble des sur-états C_1 et S_1 pour l'ensemble M_t^C et M_t^S.

Pas 3 : Calculer la condition de franchissement, S_2 : $S_2 = S_1 \setminus C_1$

Pas 4 : Construire S_3 en supprimant les termes redondants de l'ensemble S_2.

Pas 5 : Construire S_4 en appliquant l'algorithme de choix final identique à celui présenté dans la section 3.4.2.

Pas 6 : Calculer les contrôles concernés pour chaque ensemble de sur-états selon la corollaire 5.2

Pas 7 : Fin

❑

- Application de l'algorithme 5.2 à l'exemple de la figure 5.1

Les ensembles C_1^{t1} et S_1^{t1}, respectivement ensembles des sur-états des états critiques et sains, ont été déjà calculés (*Pas* 1 et 2).

$C_1^{t1} = \{P_1, P_5, P_7, P_6, P_8, P_4, P_9, P_1P_5, P_1P_7, P_5P_7, P_1P_6, P_1P_8, P_6P_8, P_1P_4, P_1P_9, P_4P_9, P_1P_5P_7, P_1P_6P_8, P_1P_4P_9\}$

$S_1^{t1} = \{P_1, P_4, P_7, P_1P_4, P_1P_7, P_4P_7, P_1P_4P_7\}$

Pas 3 :

En utilisant l'ensemble $S_2^{t1} = S_1^{t1} \setminus C_1^{t1}$, nous pouvons calculer le contrôle qui garantit le franchissement excepté dans les états critiques.

$S_2^{t1} = S_1^{t1} \setminus C_1^{t1} = \{P_4P_7, P_1P_4P_7\}$

Pas 4 :

Dans l'ensemble S_2^{t1} il y a des sur-états redondants. En supprimant ces termes nous avons :

$S_3^{t1} = \{P_4P_7\}$

Pas 5 :

Par l'application de l'algorithme de choix final, nous avons l'ensemble :

$S_F^{t1} = \{P_4P_7\}$

Pas 6 :

Et le contrôle :

$Ut_1(M_i) = m_4m_7$

La condition nécessaire et suffisante pour avoir un contrôleur maximal permissif est exactement la même que celle présentée dans la propriété 5.3.

5.4.2 Le modèle final

De manière duale nous avons calculé deux contrôleurs pour la transition t_1. Il est possible que l'un des deux ne soit pas maximal permissif. Si les deux le sont, nous avons la possibilité de choisir la solution la plus simple. Pour notre exemple, la condition de franchissement est plus simple :

$$Ut_1(M_k) = m_4m_7$$

L'algorithme complet pour calculer un contrôleur est donné dans l'algorithme 5.3. Il correspond à la fusion des algorithmes 5.1 et 5.2.

Algorithme 5.3 :

Soit G un graphe de marquage représenté par un 4-tuplet (M [M_A, M_{IF}], $\sum[\sum_c, \sum_u]$, δ, M_0).

Pas 1 : Pour $\forall\ t \in \sum_c$ calculer l'ensemble M_t^C et M_t^S à partir de l'ensemble M_A, M_{IF}.

Pas 2 : Pour chaque transition t, calculer l'ensemble des sur-états C_1 et S_1 pour l'ensemble M_t^C et M_t^S.

Pas 3 : Calculer la condition d'interdiction (franchissement) C_2 (S_2) :

$C_2 = C_1 \setminus S_1$ ($S_2 = S_1 \setminus C_1$)

Pas 4 : Construire C_3(S_3) en supprimant les termes redondants de l'ensemble C_2 (S_2).

Pas 5 : Appliquer l'algorithme de choix final comme celui présenté dans la section 3.4.2 pour construire : C_4 (interdiction) et S_4 (franchissement).

Pas 6 : Parmi les 2 contrôleurs calculés, combien sont-ils maximaux permissifs ?

Si 2 aller au pas 7.

Si 1 choisir celui qui est maximal permissif et aller au pas 8.

Si 0 pas de contrôleur maximal permissif, aller au pas 9.

Pas 7 : Choisir le contrôleur le plus simple donné par C_4 ou S_4.

Pas 8 : Calculer les contrôles concernés pour chaque ensemble de sur-états selon les corollaires 5.1 ou 5.2 selon le pas 6.

Pas 9 : Fin

❑

Pour avoir le contrôleur global nous devons calculer la condition associée à chaque transition contrôlable. Nous allons calculer cette condition pour la transition contrôlable t_4, de la même manière que pour t_1.

$C_1^{t4} = \{P_2, P_3, P_7, P_1, P_8, P_4, P_9, P_2P_4, P_2P_7, P_4P_7, P_3P_4, P_3P_8, P_4P_8, P_1P_4, P_1P_9, P_4P_9, P_2P_4P_7, P_3P_4P_8, P_1P_4P_9\}$

$S_1^{t4} = \{P_1, P_4, P_7, P_1P_4, P_1P_7, P_4P_7, P_1P_4P_7\}$

$C_2^{t4} = C_1^{t4} \setminus S_1^{t4} = \{P_2, P_3, P_8, P_9, P_2P_4, P_2P_7, P_3P_4, P_3P_8, P_4P_8, P_1P_9, P_4P_9, P_2P_4P_7, P_3P_4P_8, P_1P_4P_9\}$

$S_2^{t4} = S_1^{t4} \setminus C_1^{t4} = \{P_1P_7, P_1P_4P_7\}$

$C_3^{t4} = \{P_2, P_3, P_9, P_8\}$
$\mathcal{M}_{t4}^{C} = \{P_2P_4P_7, P_3P_4P_8, P_1P_4P_9\}$
$\Rightarrow C_F^{t4} = \{P_2, P_3, P_9\} \Rightarrow Ut_4(M_k) = (m_2 + m_3 + m_9)$ '

$S_3^{t4} = \{P_1P_7\}$
$\mathcal{M}_{t4}^{S} = \{P_1P_4P_7\}$
$\Rightarrow S_F^{t4} = \{P_1P_7\} \} \Rightarrow Ut_2(M_k) = (m_1m_7)$

Le modèle final du RdP contrôlé est présenté dans la figure 5.5.

Les règles pour la transformation d'un modèle RdP vers le modèle Grafcet sont les mêmes que celles présentées dans le chapitre 3. Le modèle Grafcet de l'exemple est présenté dans la figure 5.6

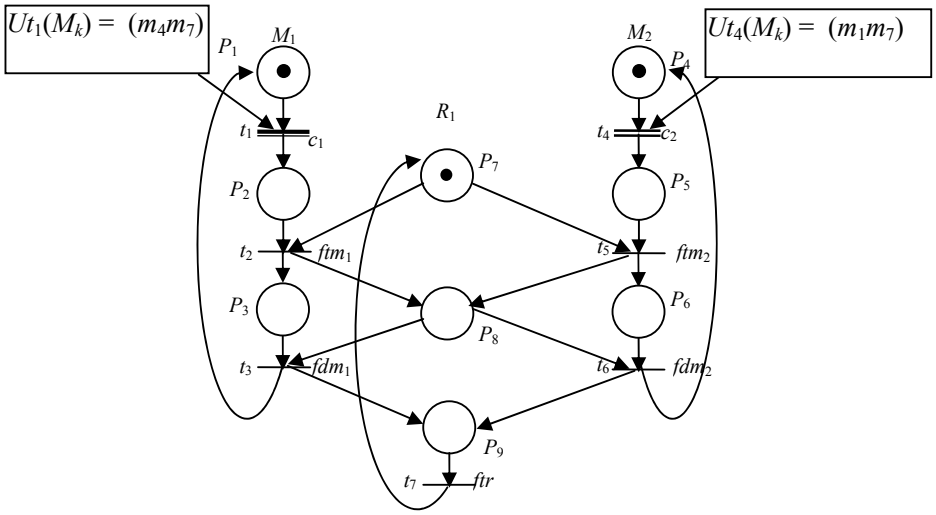


Fig. 5.5. Modèle RdP final avec le contrôle simplifié

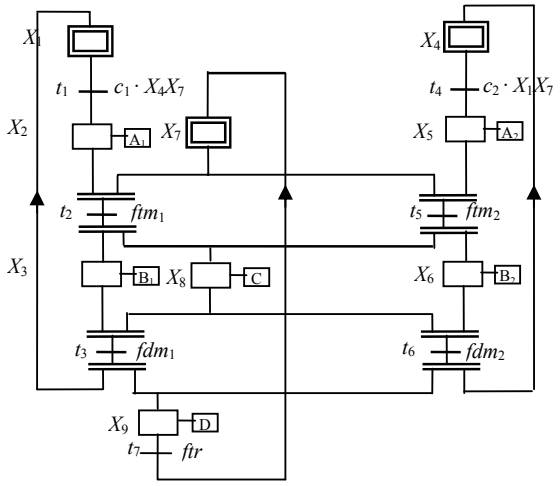


Fig. 5.6. Modèle Grafcet du RdP de la Figure 5.5

5.5 Application de la méthode de contrôle sur le système AIP

Nous appliquons cette méthode sur l'exemple de AIP qui a été présenté dans la section 4.5. Le modèle en boucle fermée de ce système est représenté dans la figure 5.7.

Pour cet exemple il y a 3 états interdits :

$\mathcal{M}_{IF} = \{P_0P_5P_9P_{14}, P_1P_5P_9P_{14}, P_3P_5P_9P_{14}\}$

A partir de cet ensemble d'états, on peut calculer l'ensemble des états critiques et sains pour toutes les transitions contrôlables. Dans cet exemple l'ensemble des états critiques pour les transitions contrôlables sont vides sauf pour la transition t_{15}.

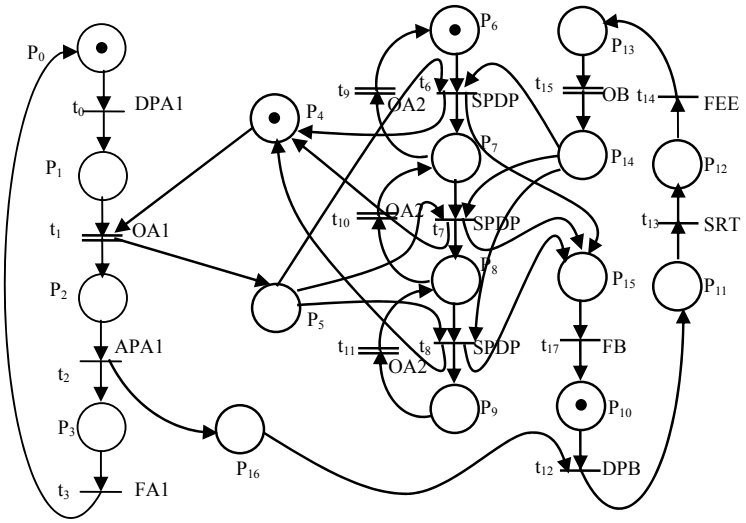


Fig. 5.7. Modèle RdP de l'exemple AIP en boucle fermé

Remarque 5.2 : Lorsque l'ensemble des états critiques pour une transition est vide, cela signifie que cette transition est toujours franchissable et il n'est pas besoin de calculer la condition de franchissement ou d'interdiction.

❑

Les ensembles des états critiques et sains pour la transition t_{15} sont :

$\mathcal{M}_{t15}{}^{C}= \{P_0P_5P_9P_{13}, P_1P_5P_9P_{13}, P_3P_5P_9P_{13}\}$

$\mathcal{M}_{t15}{}^{S}= \{P_0P_5P_6P_{13}, P_1P_5P_6P_{13}, P_3P_5P_6P_{13}, P_0P_5P_7P_{13}, P_1P_5P_7P_{13}, P_3P_5P_7P_{13}, P_0P_5P_8P_{13}, P_1P_5P_8P_{13}, P_3P_5P_8P_{13}\}$

L'ensemble des sur-états des états critiques ($C_1{}^{t15}$) et sains ($S_1{}^{t15}$) est calculé comme suit :

$C_1{}^{t15} = \{P_0, P_1, P_3, P_5, P_9, P_{13}, P_0P_5, P_0P_9, P_0P_{13}, P_5P_9, P_5P_{13}, P_9P_{13}, P_1P_5, P_1P_9, P_1P_{13}, P_3P_5, P_3P_9, P_3P_{13}, P_0P_5P_9, P_0P_5P_{13}, P_0P_9P_{13}, P_5P_9P_{13}, P_1P_5P_9, P_1P_5P_{13}, P_1P_9P_{13}, P_3P_5P_9, P_3P_5P_{13}, P_3P_9P_{13}, P_0P_5P_9P_{13}, P_1P_5P_9P_{13}, P_3P_5P_9P_{13}\}$

$S_1{}^{t15}= \{P_0, P_1, P_3, P_5, P_6, P_7, P_8, P_{13}, P_0P_5, P_0P_6, P_0P_{13}, P_5P_6, P_5P_{13}, P_6P_{13}, P_1P_5, P_1P_6, P_1P_{13}, P_3P_5, P_3P_6, P_3P_{13}, P_0P_7, P_5P_7, P_7P_{13}, P_1P_7, P_3P_7, P_0P_8, P_5P_8, P_8P_{13}, P_1P_8, P_3P_8, P_0P_5P_6, P_0P_5P_{13}, P_0P_6P_{13}, P_5P_6P_{13}, P_1P_5P_6, P_1P_5P_{13}, P_1P_6P_{13}, P_3P_5P_6, P_3P_6P_{13}, P_0P_5P_7, P_0P_7P_{13}, P_5P_7P_{13}, P_1P_5P_7, P_1P_7P_{13}, P_3P_5P_7, P_3P_7P_{13}, P_0P_5P_8, P_0P_8P_{13}, P_5P_8P_{13}, P_1P_5P_8, P_1P_8P_{13}, P_3P_5P_8, P_3P_8P_{13}, P_0P_5P_6P_{13}, P_1P_5P_6P_{13}, P_3P_5P_6P_{13}, P_0P_5P_7P_{13}, P_1P_5P_7P_{13}, P_3P_5P_7P_{13}, P_0P_5P_6P_{13}, P_1P_5P_8P_{13}, P_3P_5P_8P_{13}\}$

Les ensembles $C_2{}^{t15}$, $S_2{}^{t15}$, $C_3{}^{t15}$ et $S_3{}^{t15}$ sont :

$C_2{}^{t15}= C_1{}^{t15}\setminus S_1{}^{t15} = \{P_9, P_0P_9, P_5P_9, P_9P_{13}, P_1P_9, P_3P_9, P_0P_5P_9, P_0P_9P_{13}, P_5P_9P_{13}, P_1P_5P_9, P_1P_9P_{13}, P_3P_5P_9, P_3P_9P_{13}, P_0P_5P_9P_{13}, P_1P_5P_9P_{13}, P_3P_5P_9P_{13}\}$

$S_2{}^{t15}= S_1{}^{t15}\setminus C_1{}^{t15} =\{P_6, P_7, P_8, P_0P_6, P_5P_6, P_6P_{13}, P_1P_6, P_3P_6, P_0P_7, P_5P_7, P_7P_{13}, P_1P_7, P_3P_7, P_0P_8, P_5P_8, P_8P_{13}, P_1P_8, P_3P_8, P_0P_5P_6, P_0P_6P_{13}, P_5P_6P_{13}, P_1P_5P_6, P_1P_6P_{13}, P_3P_5P_6, P_3P_6P_{13}, P_0P_5P_7, P_0P_7P_{13}, P_5P_7P_{13}, P_1P_5P_7, P_1P_7P_{13}, P_3P_5P_7, P_3P_7P_{13}, P_0P_5P_8, P_0P_8P_{13}, P_5P_8P_{13}, P_1P_5P_8, P_1P_8P_{13}, P_3P_5P_8, P_3P_8P_{13}, P_0P_5P_6P_{13}, P_1P_5P_6P_{13}, P_3P_5P_6P_{13}, P_0P_5P_7P_{13}, P_1P_5P_7P_{13}, P_3P_5P_7P_{13}, P_0P_5P_6P_{13}, P_1P_5P_8P_{13}, P_3P_5P_8P_{13}\}$

En supprimant les termes redondants, nous obtenons :

$C_3{}^{t15}= \{P_9\} \Rightarrow C_F{}^{t15} = \{P_9\} \Rightarrow Ut_{15}(M_k) = m_9'$

$S_3{}^{t15}=\{P_6, P_7, P_8\} \Rightarrow S_F{}^{t15} = \{ \{P_6, P_7, P_8\}\} \Rightarrow Ut_{15}(M_k) = m_6+ m_7+ m_8$

Les deux conditions donnent le contrôleur maximal permissif. Mais il est évident que la condition d'interdiction est plus simple que la condition de franchissement de cette transition. Le modèle final du RdP contrôlé est présenté dans la figure 5.8.

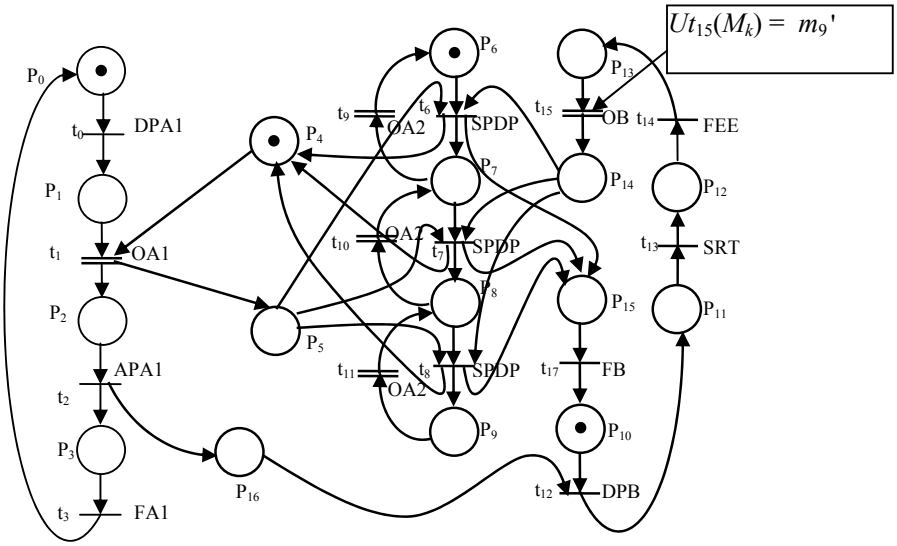


Fig. 5.8. Modèle RdP contrôlé de l'exemple AIP

5.6 Les différents méthodes de Simplification

5.6.1 Comparaisons des méthodes de synthèse de contrôleurs

Dans les chapitres 3, 4 et 5, nous avons présenté trois méthodes pour la synthèse de contrôleurs. Les résultats des méthodes présentées dans les chapitre 3 et 4 sont presque les mêmes. Dans les deux méthodes nous ajoutons des places de contrôle par l'approche basée sur les invariants. L'avantage important de la méthode présentée dans le chapitre 4 c'est qu'il est applicable pour les réseaux saufs non nécessairement conservatifs alors que la méthode présentée dans le chapitre 3 est applicable seulement sur les réseaux saufs et conservatifs. Il n'y a pas de résultat concernant les complexités des modèles contrôlés obtenus par les deux approches. Tout dépend du cas étudié.

La construction du graphe de marquage augmente la complexité de deux méthodes, mais celui-ci est construit hors ligne. Pour la méthode de simplification des RdP conservatifs, la construction de l'ensemble des états conservatifs est en général plus grand que l'ensemble des états accessibles. Dans la deuxième méthode, il est nécessaire de construire l'ensemble des sur-états pour les états interdits. Nous remarquons qu'en réalité la construction de l'ensemble des sur-états des états autorisés n'est pas nécessaire.

L'avantage des méthodes du chapitre 3 et 4 par rapport à la méthode présentée dans ce chapitre est que la dynamique du système est plus visible sur le modèle RdP. Mais l'avantage de la méthode présentée ici est qu'elle peut donner un résultat plus simple que les autres méthodes.

A cause de l'existence de deux possibilités pour le choix final, pour la méthode présentée dans ce chapitre (la condition d'interdiction et la condition de franchissement), il y a plus de chances d'avoir un contrôleur maximal permissif. S'il n'existe pas un contrôleur maximal permissif par la méthode présentée, on peut corriger la méthode comme on explique par un petit exemple. Par exemple soit $\mathcal{M}_{t1}= \{P_0P_9P_{13}, P_1P_5\}$ l'ensemble des états critiques et $\mathcal{M}_{t1}^{S}= \{P_0P_9, P_1P_5P_{13}\}$ l'ensemble des états

sains. Nous voyons qu'il y a un état critique qui couvre un état sain et il y a également un état sain qui couvre par un état critique. Donc par les méthodes présentées nous ne pouvons pas construire un contrôleur maximal permissif. Dans ce cas on peut changer la condition de franchissement comme suit :

Condition de franchissement $t_1 = m_0 m_9 m_{13}' + m_1 m_5 m_{13}$

Généraliser cette idée est un travail qu'on voudrait faire dans nos futurs travaux de recherche.

En conclusion, on peut dire que pour les trois méthodes présentées dans cette approche, il n'y a pas une méthode qui donne toujours le contrôleur le plus simple. Il faut exploiter les trois méthodes et choisir la solution la plus simple.

5.6.2 Comparaisons des méthodes de simplification avec la simplification des expressions logiques

L'idée de simplification que nous avons utilisée dans ces méthodes est proche de la simplification des expressions logiques. Il y a des points communs et des différences entre simplification des contraintes et simplification des expressions logiques. Le point commun important est que nous utilisons la ou les compléments d'une variable pour la simplification. Par exemple dans les expressions logiques nous avons toujours $c + c' = 1$ donc :

$abc + abc' = ab(c + c') = ab$

Dans la simplification des contraintes si

$m_3 + m_4 + m_5 = 1$ et $\mathcal{M}_{IF} = \{P_1P_2P_3, P_1P_2P_4, P_1P_2P_5\} \Rightarrow \mathcal{M}_{IF}' = \{P_1P_2\}$

Dans l'expression logique le complément d'une variable est toujours une seule variable et est présenté par le symbole de complément soit « ' » ou « ¯ ». Mais pour les places cela peut être plus d'une variable, comme nous avons vu dans l'exemple.

L'autre différence qui a été utilisée dans ce chapitre, c'est que dans les expressions logiques, lorsque celle-ci est égale à un, la sortie de la fonction associée devient vraie. Alors que pour le RdP, dans le franchissement d'une transition, en plus de la vérification de la condition, la transition doit être validée par les places en amont. Cette différence peut nous aider pour plus de simplification en augmentant l'ensemble des états indifférents.

5.7 Conclusion

Dans ce chapitre nous avons présenté une méthode efficace pour résoudre le problème des états interdits pour les RdP saufs. L'idée fondamentale est d'utiliser des conditions plus simples pour empêcher les états interdits à partir des états critiques. Cette condition est représentée par une expression logique associée aux transitions contrôlables. Pour calculer cette condition il y deux possibilité : soit on peut calculer la condition de franchissement, soit la condition d'interdiction. A la fin on peut choisir la plus simple condition. Nous avons établi une condition nécessaire et suffisante garantissant l'optimalité du contrôleur.

Chapitre 6

Conclusion et Perspectives

6.1 Conclusion

La synthèse de contrôleurs pour les systèmes à événements discrets a été abondamment étudiée dans la littérature à partir des outils automates et Réseaux de Petri. La théorie de la synthèse de contrôleurs a été introduite par Ramadge et Wonham en utilisant les automates. L'inconvénient principal de cette approche est l'explosion du nombre d'états pour les systèmes complexes. Il existe différentes approches qui utilisent les modèles RdP pour pallier ce problème. Parmi ces approches, les plus simples ne résolvent pas le problème de l'optimalité du contrôleur, et les plus complexes sont difficiles à mettre en œuvre.

Dans notre travail, nous proposons une méthode systématique et facile de mise en œuvre pour la synthèse du contrôle des systèmes à événements discrets. Nous modélisons les systèmes par des modèles RdP saufs. Deux méthodes distinctes sont utilisées. La première consiste à ajouter des places de contrôle pour empêcher l'atteignabilité des états interdits, et la deuxième revient à ajouter des conditions pour les transitions contrôlables. Nous avons alors été confrontés au problème des transitions incontrôlables pour garantir l'optimalité et à la complexité apportée par le nombre des places de contrôle qui peut être très grand.

Pour résoudre ces problèmes, nous avons utilisé l'approche principale de Ramadge et Wonham ainsi que le travail de Kumar en construisant le graphe de marquage pour déterminer les états interdits dans le cas général où il y a des événements incontrôlables. Ensuite par utilisation du théorème introduit par Giua, nous avons traduit l'ensemble des états interdits en un ensemble des contraintes linéaires. L'utilisation de méthodes de simplification des contraintes constitue la contribution fondamentale de cette thèse. Grâce à elles, il est possible de réduire le nombre et la borne des contraintes et ainsi construire un modèle contrôlé simple et facile de mise en ouvre.

La première méthode de simplification présentée est applicable sur les RdP saufs et conservatifs alors que la deuxième est applicable sur les RdP saufs dans le cas général. Ces méthodes sont basées sur deux propriétés fondamentales. Ces deux propriétés viennent des relations de l'invariant de marquage de places qui peut être total ou partiel. Par l'utilisation de la première propriété, la borne et le nombre des contraintes sont diminués alors que par l'utilisation de la deuxième propriété seul le nombre des contraintes est diminué.La méthode de synthèse de contrôle en appliquant la méthode de simplification pour les RdP conservatifs et saufs donne toujours le contrôleur maximal permissif.

L'équivalence entre les états interdits et les contraintes linéaires n'est pas toujours possible pour les RdP saufs. Nous avons alors déterminé les conditions nécessaires et suffisantes pour avoir un contrôleur maximal permissif en se basant sur notre méthode de simplification. Dans beaucoup de cas les résultats des deux méthodes sont comparables, néanmoins le résultat final peut être plus ou moins simple et dépend du cas étudié. Une comparaison a été faite au niveau du contrôleur obtenu pour un ensemble d'exemples. L'avantage principal de ces méthodes de synthèse de contrôleurs est que le modèle RdP contrôlé est très proche du modèle initial.

La deuxième idée qui a été utilisée pour la synthèse est l'utilisation des conditions pour le franchissement des transitions contrôlables. Cette méthode a été présentée dans le chapitre 5. Elle consiste à interdire le franchissement lorsqu'un état critique est atteint. Les méthodes qui utilisent cette idée, ont en général besoin d'un calcul long en temps réal. En appliquant notre méthode de simplification, nous arrivons à un contrôleur simple dans la plupart des cas.

Pour pouvoir implanter les contrôleurs obtenus, nous présentons une méthode pour la transformation d'un RdP vers un Grafcet. Si le modèle RdP est le RdP Interprété de Commande (RdPIC), la transformation est facile. Lorsque ce n'est pas le cas, nous avons résolu deux problèmes : le premier a lieu quand les places de contrôles sont non saufs et le deuxième lorsque le modèle RdP du contrôleur est non déterministe.

6.2 Perspectives

Dans notre thèse nous avons considéré les systèmes qui peuvent être modélisés par des RdP saufs avec des événements contrôlables et incontrôlables. Il est possible de développer une approche pour un cadre plus général :

- Existence des événements non observables :
- Prise en compte du temps dans les dates d'occurrence des événements. Cela peut permettre la construction de contrôleur maxiamal permissif. On sait que le temps peut rendre un système contrôalble.
- Déterminer un contrôleur maximal permissif alors que la propriété 5.3 n'est pas vérifiée. Une solution manuelle à ce problème est donnée dans la section 5.4.1.
- Appliquer la méthode de synthèse de contrôleur par retour d'état pour les RdP non saufs :

La méthode que nous avons utilisée pour la synthèse de contrôleurs par retour d'état dans le chapitre 5 peut être généralisée pour les RdP non saufs. Dans ce cas, pour une place non sauf on peut indiquer le marquage non unitaire par une puissance. Par exemple le marquage M_i = [0,0,1,0,2,1] peut être représenté par l'expression $m_3m_5^2m_6$. Elle sera vérifiée lorsque l'état actuel couvre cet état. Bien sûr que dans ce cas la complexité de méthode sera beaucoup augmentée.

- Développer un logiciel pour systématiser les étapes à partir de la synthèse du contrôleur.
- Application partielle de la méthode de simplification:

Pour les systèmes complexes, il est possible que le nombre des places et le nombre d'invariants devienne très grand. A cause de la croissance exponentielle du nombre des états conservatifs, l'espace mémoire et le temps de calcul deviennent prohibitifs. La question ouverte est de savoir si on peut casser cette complexité en divisant le problème global en sous-problèmes tout en maintenant l'optimalité de la solution.

Append A :

***Algorithme* 1 :** *Application de la propriété* 3.1 *pour la simplification* :

Soit m : *le nombre d'invariant,* n_i : *le nombre de place dans l'invariant i,* $\mathcal{P}_i$ *: l'ensemble des places d'invariant i.* P_{ij} : *j èm place de l'ensemble* $\mathcal{P}_i$.

Pas 1 : *Soit* $M := [j_m,...,j_i,...,j_1]$ *un état conservatif. Commencer par* [0,0,…,0]

Pas 2 : *Changer le vecteur J pour trouver un nouveau état, soit interdit soit non atteignable. Si il n'y a pas, aller au pas* 4 *si non soit* J_k *le vecteur d'un état concerné.*

Pas 3 : *pour toute les élément* j_{ki} *du vecteur* J_k*, si pour le* $0 \leq j_{ki} < n_i$ $M_k \notin \mathcal{M}_A$*, indiquer* j_{ki} *comme un invariant supprimé du vecteur* J_k*.(Par exemple remplacer par un caractère spécial comme* *). *Enregistrer le M qui peut être modifié dans l'ensemble des contraintes simplifiées* C_1. *Aller au pas* 2.

Pas 4 : *Comparer toutes les contraintes de* C_1 ;

S'il y a des sur-états $c_1, c_2,..., c_r$ *qu'ont la même invariant supprimé* j_i *et sont différent seulement dans un invariant* j_k *, et* $r=n_k$ *ajouter une nouvelle contrainte en supprimant l'invariant* j_k *et marquer toutes les contraintes* $c_1, c_2,..., c_r$ *comme des contraintes sélectionnées. Aller au pas* 4 .

En cas contraire supprimer tous les contraintes sélectionnées.

Pas 5 : *Fin*

***Algorithme* 2 :** *Application de la propriété 3.*2 *pour la simplification* :

Pas 1: *Soit C une contrainte simplifiée de l'ensemble*C_1, *représenté dans l'ensemble* C_2 *comme ci-après*: $C =: [\mathcal{J}_m,..., \mathcal{J}_i,..., \mathcal{J}_1]$ $\mathcal{J}_i \subseteq \{*, 0, 1,..., n_i - 1\}$

(Dan l'algorithme 1, j_i a été un nombre entier ou le symbole *, mais ici $\mathcal{J}_i$ est un ensemble de nombres entiers ou le symbole *. On peut utiliser le symbole * comme l'existence de tous les possibilités de $\mathcal{J}_i$)

Pas 2 : *Comparer les deux contraintes* C_L *et* C_K*, si toutes moins une élément* (*élément* $\mathcal{J}_{Lr}$ et J_{Kr}) *soit égal, soit un couvert par l'autre, calculer un nouveau contrainte* C_{LK} : $[\mathcal{J}_{LKm},... \mathcal{J}_{LKr},..., \mathcal{J}_{LKi},..., \mathcal{J}_{LK1}]$ $\mathcal{J}_{LKr} = \mathcal{J}_{Lr} \cup \mathcal{J}_{Kr}$, $\mathcal{J}_{LKi} = \mathcal{J}_{Li} \cap \mathcal{J}_{Ki}$ $(i \in \{1,..., m\} - \{r\})$ si *non aller au pas* 4.

Pas 3: *Enregistrer le contrainte* C_{LK} *dans l'ensemble des contraintes simplifiés* C_2 *si il n'existe pas une contrainte* C_I *qui couvre* C_{LK} :

Supprimer de l'ensemble C_2, *toutes les contraintes qui couvrent par* C_{LK}.

Aller au pas 2.

Pas 4 : *Fin*

Référence

AIP, 2001, Atelier Inter - établissement de Productique, Grenoble, France, http://gilco.inpg.fr/~tollenaere/aip-ds/

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

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

Basile F., Chiacchio P., Giua A., 2006, "Suboptimal supervisory control of Petri nets in presence of uncontrollable transitions via monitor places", *Automatica*, 42, 995-1004 .

Blanchard M., 1979, Comprendre, Maîtriser et appliquer le GRAFCET. *CEPADUES Editions, Toulouse*.

Boel, R. K., Ben-Naoum, L., and Van Breusegem, V. 1995. "On the forbidden state problems for a class of controlled Petri nets". *IEEE Trans. on Automatic Control* 40(10): 1717–1731.

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

Charbonnier F., 1996, Commande supervisée des systèmes à événements discrets, *thèse de doctorat, INPG, France.*

David R., Alla H., 2005, "Discrete, Continuous, and Hybrid Petri Nets", *Springer*, ch 1-3.

Dideban A., Alla H., 2004, « Des états interdits vers les contraintes linéaires », *note interne*, *Laboratoire d'automatique de Grenoble.*

Dideban A., Alla H., 2005, "From forbidden state to linear constraints for the optimal supervisory control", *CEAI,* vol.7, No. 3, pp 48-55.

Dideban A., 2005, "Synthèse optimale d'un contrôleur par construction de l'ensemble minimal de contraintes**"**, *RS- JESA,* Vol. 39 (1-2-3), *MSR05,* page 127-141.

Dideban A., Alla H., 2006a, "Synthesis of Feedback Control Logic for Safe and Controlled Petri Nets", *Iranian Conference on Electrical Engineering (ICEE* 2006), 16-18 May, Theran, IRAN.

Dideban A., Alla H., 2006b, "Solving the problem of forbidden states by feedback control logical synthesis", *The* 32*nd Annual Conference of the IEEE Industrial Electronics Society*,7-10 Nov, Paris, FRANCE.

Dideban A., Alla H., 2007, "Determination of Minimal Sets of Control Places for Safe Petri Nets", *American Control Conference*, 11-13 July, New York City, USA.

Ezpeletaa J., J. M. Colom, J. Martinez, 1995, "A Petri Nets Based deadlock prevention policy for flexible manufacturing systems", *IEEE Trans. On Robotics and Automation*, 11, pp. 173-184.

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

Gaudin B., 2004, « Synthèse de contrôleurs sur des systèmes à événement discrets structurés », *thèse de doctorat, Institut de formation supérieure en informatique et communication de l'université de Rennes, France*.

Ghaffari A., Rezg N. and Xie X, 2003a., "Feedback control logic for forbidden – state problems of Marked graphs: application to a real manufacturing system", *IEEE Trans. Automatic Control*, 48(1):18-29.

Ghaffari A., N. Rezg and X.-L. Xie, 2003b, "Design of Live and Maximally Permissive Petri Net Controller Using Theory of Regions", *IEEE Trans. On Robotics and Automation*, 19(1).

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

Giua A., F. DiCesare, 1993a, "Grafcet and Petri Nets in Manufacturing ", in *Intelligent Manufacturing: Programming Environments for CIM*, W.A. Gruver and J.C. Boudreaux (Eds.), pp. 153-76, Springer-Verlag,.

Giua A., F. DiCesare and M. Silva, 1993b, "Petir net supervisors for generalized mutual exclusion constraints", *Proc.* 12*th IFAC World Congresse, Sidney, Australia*.

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

Holloway L. E., X. Guan, L. Zhang,1996, "A Generalisation of state avoidance Policies for Controlled Petri Nets". *IEEE Trans. Autom. Control*, AC-41, 6, 804-816.

Iordache M. V., J. O. Moody, and P. J. Antsaklis, 2001, "A method for the synthesis liveness enforcing supervisors in Petri nets," in *Proc. Amer.Control Conf.*, pp. 4943–4948.

Kattan B. 2004, « Synthèse structurelle d'un contrôleur basée sur le Grafce », *Thèse de doctorat, UJF, France*.

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

Kumar R., L.E. Holloway, 1996, "Supervisory control of deterministic Petri nets with regular specification languages", *IEEE Trans. Automatic Control*, 41(2):245-249.

Leparc P., 1994, "Apports de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet " , *Thèse de doctorat de l'université de Rennes* 1.

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

Li Zhi Wu , MengChu Zhou, 2004, "Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems*", IEEE Transactions on systems, Man, And Cybernetics—Part A: Systems And Humans, Vol. 34, No.* 1.

Lin F., W. M. Wfonham, 1990, "Decentralized control and coordination of discretevent systems with partial observation". *IEEE Transactions of Automatic Control*, 35(12):1330-1337.

Moody J. O., Antsaklis P., 2000, "Petri net supervisors for DES with uncontrollable and unobservable transition", *IEEE Trans. Automatic Control*, 45(3):462-476.

Morris Mano M., 2001, "Digital Design"*, Prentice Hall* , ch 3.

Music G., MAtko D., Zupancic B., 2000, "Modelling, synthesis, and simulation of supervisory process control systems", *Mathematical and computer modelling of dynamical systems,* Vol. 6, No. 2 : 169-189

Ramadge P. J., Wonham W. M., 1987a, "Modular feedback logic for discrete event systems", *SIAM Journal of Control and Optimization*, 25 (5):1202-1218.

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

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

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

Sava A. T., 2002, "Sur la synthèse de la commande des systèmes à événements discrets temporisés", *thèse de doctorat, INPG, France*.

Takai S., 2005, "Supervisory Control of a Class of Concurrent Discrete Event Systems Under Partial Observation", *Discrete Event Dynamic Systems*, 15, 7-32.

Uzam M., Jones A. H., 1998, "Discrete event control System Design Using Automation Petri Nets and their Ladder Diagram Implementation" *Int J Adv Manuf Tech,* 14: 716-728.

Uzam M., Wonham W., 2006, "A hybrid approach to supervisory control of discrete event systems coupling RW supervisors to Petri Nets" *Int J Adv Manuf Tech,* 28: 747-760.

Wonham W. M., 2005, "Supervisory Control of Discrete-Event Systems", research report, http://www.control.toronto.edu/people/profs/wonham/wonham.html.

Yamalidou K., Moody J., Lemmon M.and Antsaklis P., 1996, "Feedback control of Petri Nets based on place invariants", *Automatica*, 32(1):15-28.

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