UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Automatique-Productique**

Arrêté ministérial : 7 août 2006

Présentée par

Andra - Ioana VASILIU

Thèse dirigée par **Hassane ALLA**

préparée au sein du **Laboratoire GIPSA-Lab**
et de **L'école doctorale EEATS**

Synthèse de contrôleurs des systèmes à événements discrets basée sur les réseaux de Petri.

Thèse soutenue publiquement le **3 février 2012**,
devant le jury composé de :

M. Jean - Marc FAURE
Professeur, SUPMECA Paris, Président

M. Jean - Louis BOIMOND
Professeur, Université d'Angers, Rapporteur

M. Jean - Marc FAURE
Professeur, SUPMECA Paris, Rapporteur

Mme. Simona CARAMIHAI
Professeur, Université Polytechnique de Bucarest, Examinateur

M. Eric RUTTEN
Chargé de recherche, INRIA Rhône - Alpes, Examinateur

M. Hassane ALLA
Professeur, UJF Grenoble, Directeur de thèse

"I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me."

Sir Isaac Newton

Remerciements

Je tiens à adresser mes plus sincères remerciements et toute ma reconnaissance à mon directeur de thèse, M. Hassane Alla, pour son soutien constant, sa patience et sa confiance.

Je remercie également les membres du Jury, qui ont accepté de juger mon travail : M. Jean - Marc Faure pour m'avoir fait l'honneur de présider la soutenance ainsi que d'examiner mon travail, M. Jean - Louis Boimond, Mme. Simona Caramihai et M. Eric Rutten pour avoir accepté d'examiner en profondeur ce travail, ainsi que pour tous leurs commentaires constructifs.

J'adresse ma profonde reconnaissance à Mme. Simona Caramihai pour ses conseils, son soutien et sa confiance pendant et après mes années à l'Université Polytechnique de Bucarest et pour m'avoir dirigé vers la recherche scientifique.

J'exprime ma reconnaissance aux professeurs de Gipsa-Lab, et particulièrement aux membres de l'équipe SYSCO.

Je voudrais aussi remercier mes amis et collègues du laboratoire, qui ont chacun apporté leur touche personnelle à mon expérience grenobloise, ainsi qu'au personnel qui rend la vie des thésards plus agréable : Joumana, Liz, Amine, Haithem, Hieu, Antoine, Irfan, Lam, Simona, Oumayma, Pham, Jennifer, Federico, Sara, Emilie, Valentina, Gabriel, Patricia, Marie - Thérèse, Virginie.

Un grand merci à Joumana pour avoir accepté de corriger mon rapport et pour son soutien constant pendant la rédaction.

Je remercie également mes amis de Grenoble pour leur encouragements et leur vitalité vitaminée : Michelle, Arnaud, Delphine, Arturo, Pancho, Maria, Lucian, Luis, Olga, Anca, Vicente, David, José Luis.

J'exprime ma plus profonde et sincère gratitude à ma famille, pour... tout ! Je remercie aussi de tout mon cœur mes plus proches et plus chers amis : toute l'Obidenia, et spécialement Laura, Miru et Liv, et aussi Ana, Anca et Andreea.

Résumé

La méthode des invariants est une des plus utilisées méthodes de synthèse pour les SED modélisés par des réseaux de Petri (RdP). Cependant, elle ne garantit pas en général une solution optimale dans la présence de l'incontrôlabilité. Dans ce travail nous proposons une solution à ce problème pour les RdP généralisés.

Premièrement, nous proposons une solution d'identification des contraintes admissibles pour les RdP saufs non-conservatifs. La méthode repose sur une définition des contraintes contenant des marquages complémentés. Ceux-ci sont après éliminés en exploitant les composants conservatifs des RdP.

Deuxièmement, nous avançons une technique de détermination des contraintes admissibles pour les RdP généralisés. La méthode est basée sur une vision spatiale de l'espace d'états du modèle. Les contraintes sont dérivées de l'équation d'hyperplan affine qui sépare les régions interdite- et autorisée- de cet espace. Nous proposons un algorithme pour le calcul du contrôleur optimal minimal.

Abstract

The place-invariants method is one of the most popular controller synthesis approaches for Petri net (PN) modeled DES. Unfortunately, the observance of the constraints is not certain in the presence of uncontrollable transitions. This thesis offers a solution to this problem for ordinary and generalized PNs.

We begin by studying safe non-conservative PNs, and devising a constraint-determination technique that will always provide a set of admissible constraints for this type of model. The approach stems from the general definition of forbidden states — that of marking vectors.

In the second part of our work, we present an admissible constraint-determination technique for generalized PNs. The method is based on a special view of the system's state space. The constraints are derived from the equation of the affine hyper-plane separating the authorized- and forbidden- regions of this space. We propose an algorithm that allows the identification of the minimal maximally permissive controller.

Table des matières

Table des figures

Liste des tableaux

Introduction

Le contrôle des systèmes automatisés est depuis longtemps un problème de grand intérêt pour le monde scientifique. Une littérature abondante a été développée pour la synthèse de contrôleurs pour les systèmes ayant des modèles continus. Cependant, les outils de base pour la commande des systèmes à événements discrets, tels que le Grafcet, consistaient à concevoir un modèle de la commande et à le valider par une série de tests. Malheureusement, cette méthode n'est pas satisfaisante pour les systèmes complexes et critiques, tels que les avions, les centrales nucléaires ou les raffineries de pétrole. Ce type de système, qui exige un grand niveau de sécurité et de précision, nécessite des méthodes formelles de contrôle, qui permettent de garantir *à priori* un fonctionnement conforme au spécifications. Les premiers travaux dans cette direction ont eu naissance au début des années 1980. Des nombreuses théories ont été depuis proposées, mais le problème reste d'actualité.

Les systèmes à événements discrets sont des systèmes dynamiques fondamentalement asynchrones pour lesquels l'espace d'états est discret. Leur évolution se fait conformément à l'arrivée des événements caractérisant le changement d'état du système. Au lieu de s'intéresser au déroulement continu des phénomènes, les systèmes à événements discrets ne se soucient que des débuts et des fins de ces phénomènes (les événements discrets) et de leur enchaînement logique, dynamique ou temporel. Leurs domaines principaux d'application incluent, parmi d'autres, la production manufacturière, la robotique, la circulation des véhicules, la logistique, les réseaux de communication et l'informatique.

Il y a maintenant une multitude d'outils permettant l'étude des systèmes à événements discrets, tels que la simulation sur ordinateur, les réseaux de files d'attente, les langages de programmation parallèle / temps réel, les modèles dynamiques algébriques, les chaînes de Markov, les automates et les réseaux de Petri. Nous allons rappeler dans cette thèse les approches de modélisation et d'analyse des systèmes à événements discrets principalement impliquées dans la recherche sur la synthèse de contrôleurs, qui est le sujet de ce travail ; c'est-à-dire les automates et les réseaux de Petri.

Étant donné un système manufacturier, nous nous intéressons à lui imposer un fonctionnement respectant un cahier des charges désiré. Pour les systèmes peu complexes, la manière la plus simple de contrôle est la conception directe du modèle supervisé du

système. Cependant, quand la complexité du système augmente, cette méthode devient très laborieuse et ne peut pas garantir le respect du cahier des charges. Il faut donc se tourner vers des méthodes formelles permettant la synthèse de contrôleurs.

Ce sont les travaux de P. J. Ramadge et W. M. Wonham qui constituent les bases de la théorie générale de la commande par supervision des systèmes à événements discrets ([Ramadge & Wonham, 1987*a*], [Ramadge & Wonham, 1987*b*]). L'objectif de l'approche de Ramadge et Wonham est de synthétiser un contrôleur pour un modèle donné, tel que le fonctionnement du procédé couplé au contrôleur reste toujours compris dans l'ensemble des comportements valides. De plus, ce fonctionnement doit être le plus permissif possible. Le comportement du système non-contrôlé, ainsi que l'ensemble des comportements désirés, sont chacun définis par un langage. À partir de ces langages, le but de la synthèse est de déterminer le sous-ensemble de comportements du procédé qui appartient aussi à la spécification. Ce sous-ensemble caractérise le langage du contrôleur, et décrit les comportements du système commandé.

L'approche de Ramadge et Wonham assure l'optimalité du contrôleur pour les systèmes modélisés par des automates à états finis. Cependant, bien que ce formalisme permette la modélisation d'une large classe de systèmes à événements discrets, il est peu adapté au systèmes réels à cause de sa grande sensibilité aux problème de l'explosion combinatoire du nombre d'états (même pour les petits systèmes). En outre, l'utilisation de cette approche a montré une difficulté dans l'implémentation du système contrôlé. C'est pour ces raisons que de nombreuses tentatives ont été faites pour adapter la théorie générale de la commande des systèmes à événements discrets à d'autres formalismes de modélisation ([Uzam & Wonham, 2006], [Li & Wonham, 1994], [Holloway & Krogh, 1990], [Holloway et al., 1996], [Kumar & Holloway, 1996], [Ghaffari et al., 2003*b*], [Ghaffari et al., 2003*a*], [Yamalidou et al., 1996], [Dideban & Alla, 2006], [Dideban & Alla, 2007], [Dideban & Alla, 2008], [Liao et al., 2010], [Nazeem et al., 2011]).

Une manière élégante de dépasser l'inconvénient de l'explosion combinatoire du nombre d'états est l'utilisation des réseaux de Petri. Ce formalisme est caractérisé par des structures de modélisation beaucoup plus riches et offre un net gain d'échelle. En outre, les réseaux de Petri sont aussi plus adaptés pour la modélisation des activités distribuées au sein des systèmes complexes, permettant une représentation beaucoup plus pratique et claire des différents sous-systèmes impliqués et de leurs interdépendances.

Cependant, si le réseau de Petri est choisi en tant que formalisme de modélisation, le gain en concision et richesse de représentation est contrebalancé par la perte de la généralité et / ou de l'optimalité des résultats. Toutefois, les nombreux avantages offerts par les réseaux de Petri ont motivé le développement de plusieurs théories de contrôle pour des classes de modèles de ce type. Nous rappelons, parmi d'autres, la méthode des invariants, la théorie des régions ou le contrôle par retour d'état.

La clarté, la simplicité et l'efficacité de calcul du contrôleur ont rendu la méthode des invariants une des méthodes de synthèse les plus utilisées pour les systèmes à événements discrets modélisés par des réseaux de Petri. La méthode impose un ensemble de contraintes linéaires sur le marquage du système. Le problème des états interdits est résolu par l'ajout judicieux de places et d'arcs de contrôle au modèle du procédé, le résultat étant un modèle réseau de Petri du contrôleur qui a le fonctionnement désiré en boucle fermée.

Malheureusement, cette approche ne garantit pas en général une solution optimale dans le cas où la synchronisation entre le procédé et la spécification est réalisée *via* des transitions incontrôlables. Dans ce travail nous proposons une solution à ce problème pour le cas des réseaux de Petri ordinaires et généralisés. L'idée centrale est celle de l'équivalence entre l'ensemble d'états interdits et celui des contraintes destinées à bloquer l'accès vers ces états. Cela revient a déterminer un ensemble de contraintes caractérisant de manière bijective le comportement autorisé du système. Le problème peut être résolu en utilisant la théorie de Ramadge et Wonham en conjonction avec le graphe de marquage du modèle. Notre objectif est de combler le fossé entre le graphe de marquage et le calcul du contrôleur optimal. Notre travail est donc situé en amont de la méthode des invariants, et concerne l'étape de conception des contraintes admissibles pour cette dernière.

Le premier chapitre de ce manuscrit donne une présentation générale des notions et formalismes de modélisation utilisés dans notre travail. Dans le 2-ème chapitre nous menons une brève étude de la théorie générale du contrôle des systèmes à événements discrets et de l'état de l'art dans ce domaine. La problématique spécifique de notre travail est détaillée dans le 3-ème chapitre.

Notre contribution originale est présentée dans les chapitres 4 et 5.

Dans le 4-ème chapitre nous proposons une solution pour la détermination des contraintes équivalentes aux états interdits dans le contexte des modèles réseaux de Petri binaires et non-conservatifs. Cette équivalence repose sur une première définition des contraintes contenant des marquages complémentés. Pour pouvoir appliquer la méthode des invariants, il a été nécessaire d'éliminer les marquages complémentés. Ceci est réalisé en exploitant les composants conservatifs des réseaux de Petri. Lorsqu'il existe des places non-conservatives, nous montrons qu'il est possible de construire un réseau de Petri conservatif isomorphe en ajoutant des places implicites au modèle. Celles-ci sont supprimées dans le modèle du contrôleur final. La méthode développée dans ce chapitre a donné lieu aussi à une présentation à l'IFAC World Congress 2011 ([Vasiliu & Alla, 2011*b*]) et à un article de revue ([Vasiliu & Alla, 2010]).

Dans le 5-ème chapitre nous nous sommes intéressés à la généralisation de l'idée pour la classe des réseaux de Petri généralisés. Comme la démarche pour les réseaux de Petri binaires non-conservatifs est basée sur le complément logique (une propriété

spécifique de ces réseaux de Petri), cette idée n'est pas valide ici. L'objectif est encore une fois de déterminer un ensemble de contraintes caractérisant de manière bijective le comportement autorisé du système. Nous allons ici développer une nouvelle technique de détermination des contraintes admissibles, ayant comme base une vision spatiale de l'espace d'états du modèle réseau de Petri généralisé ([Vasiliu & Alla, 2011*a*]).

Chapitre 1

Systèmes à événements discrets

Ce chapitre donne une succincte présentation des notions de base concernant les systèmes à événements discrets. Après une courte introduction sur les langages, nous passons à la description des deux outils de modélisation et d'analyse des systèmes à événements discrets les plus largement utilisés — les automates et les réseaux de Petri. L'accent est mis sur le formalisme de modélisation de base utilisé dans cette thèse : le réseau de Petri.

1.1 Introduction

Les systèmes à événements discrets (SED) sont des systèmes dynamiques fondamentalement asynchrones pour lesquels l'espace d'états est discret. Leur évolution se fait conformément à l'arrivée des événements caractérisant le changement d'état du système.

Au lieu de s'intéresser au déroulement continu des phénomènes, les modèles SED ne se soucient que des débuts et des fins de ces phénomènes (les événements discrets) et de leur enchaînement logique, dynamique ou temporel. Leurs principaux domaines d'application incluent, parmi d'autres, la production manufacturière, la robotique, la circulation des véhicules, la logistique, les réseaux de communication et l'informatique.

Formellement, le changement d'état d'un SED peut être décrit par un couple (événement, instant d'occurrence). Un ensemble ordonné de tels couples s'appelle une *séquence*.

Exemple 1.1.1
Considérons l'exemple d'une machine qui peut être dans trois états : arrêt, marche et panne. La figure 1.1 représente une évolution possible de ce SED.

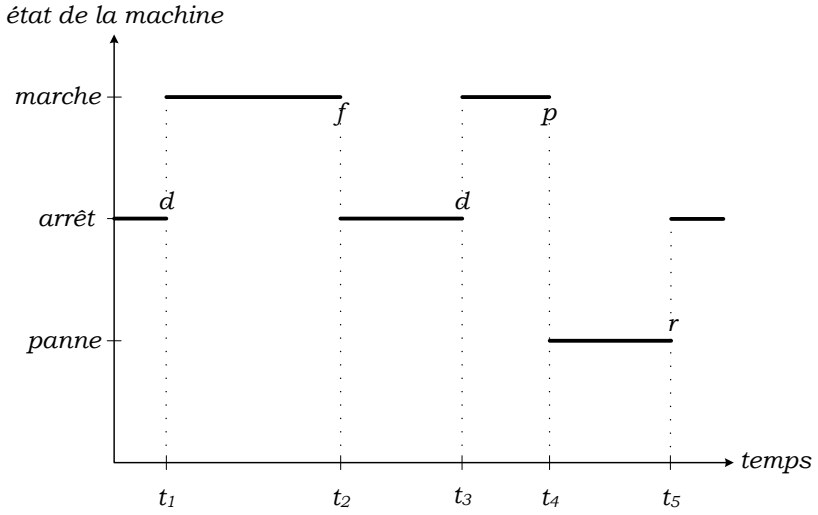


FIGURE 1.1 – Chronogramme de l'évolution de l'état d'une machine

Dans l'état initial la machine est supposée être en arrêt. Au moment t_1 une pièce est détectée à l'entrée, l'événement d = début de cycle d'usinage se produit, et la machine évolue dans l'état de marche. Il y a deux évolutions possibles, à partir de cet état. Soit le cycle d'usinage est accompli sans erreur et la machine dépose la pièce usinée à la sortie simultanément à l'occurrence de l'événement f = fin de cycle d'usinage, soit la machine tombe en panne – fait signalé par l'occurrence de l'événement p = panne. Depuis l'état de panne, la réparation de la machine – événement r – ramène celle-ci à son état initial. La pièce en cours de traitement est supposée perdue pendant la réparation.

Les deux scénarios de fonctionnement décrits précédemment sont présentés dans la figure 1.1. La machine mène à terme le premier cycle d'usinage au moment t_2 (événement f), commence un nouveau cycle au moment t_3 (événement d), tombe en panne au moment t_4 (événement p) et est réparée au moment t_5 (événement r). La séquence décrivant ce fonctionnement est comme suit : $\mathcal{S} = (d, t_1), (f, t_2), (d, t_3), (p, t_4), (r, t_5)$.

❑

Si le SED est décrit par un modèle logique, seul l'ordre des événements importe. Dans ce cas le temps est omis et nous parlons de séquences d'événements. Dans ce contexte, le fonctionnement de la machine de l'exemple 1.1.1 est donné par la séquence d'événements $\mathcal{S} = d, f, d, p, r$.

1.2 Langages

Généralement, les SED ont un comportement non-déterministe, au sens où, en fonction de l'état du système, un ou plusieurs événements différents sont susceptibles de se produire. Par conséquence, une seule séquence n'est plus suffisante pour décrire le fonctionnement du système. L'ensemble de séquences d'événements qui caractérise l'évolution d'un SED donné constitue un *langage* sur l'ensemble des événements possibles dans le système ([Hopcroft et al., 2007], [Cassandras & Lafortune, 2008]).

1.2.1 Définitions

Les langages sont les outils de base de la théorie de contrôle et de la supervision des SED. Ce formalisme s'appuie sur des opérations avec les symboles représentant les événements physiques qui entraînent l'évolution du système. L'ensemble (fini) de ces symboles constitue l'*alphabet* du système. Dans ce contexte, on appelle un *mot* (ou une *chaîne*, ou une *séquence*) défini sur un alphabet Σ toute suite finie d'éléments de Σ. On appelle ε le mot vide.

Chaque mot s est caractérisé par sa longueur : $|s|$ = le nombre des symboles de s. Par exemple, la longueur de la séquence définissant le fonctionnement présenté dans la figure 1.1 est $|\mathcal{S}| = |dfdpr| = 5$ et la longueur du mot vide est $|\varepsilon| = 0$.

Étant donné un alphabet, Σ, et un nombre entier naturel, n, la notation Σ^n désigne l'ensemble de tous les mots de longueur n sur Σ. Pour l'alphabet $\Sigma = \{d,\ f,\ p,\ r\}$ de l'exemple 1.1.1, nous avons : $\Sigma^0 = \{\varepsilon\}$, $\Sigma^1 = \{d,\ f,\ p,\ r\}$, $\Sigma^2 = \{dd,\ df,\ dp,\ dr,\ fd,\ ff,\ fp,\ fr,\ pd,\ pf,\ pp,\ pr,\ rd,\ rf,\ rp,\ rr\}$, etc.

Par extension, Σ^* définit l'ensemble des mots d'une longueur n quelconque qui peuvent être construits sur Σ :

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i,$$

le symbole "$\cup$" représentant l'opération d'union. En gardant le même exemple, nous avons : $\Sigma^* = \{\varepsilon,\ d,\ f,\ p,\ r,\ dd,\ df,\ dp,\ dr,\ fd,\ ff,\ fp,\ fr, ...\}$.

Définition 1.2.1. *On appelle langage défini sur un alphabet Σ tout sous-ensemble de Σ^*.*

❑

Remarques 1.2.1.

a) On appelle *langage infini* tout langage comportant un nombre infini de mots. $\mathcal{L} = \Sigma^*$ est un langage infini.

b) La notation $\mathcal{L}_\varnothing$ désigne le langage vide (le langage qui ne comporte aucun mot).

❑

Définition 1.2.2. *Étant donné un langage $\mathcal{L}$, on appelle la préfixe-clôture de $\mathcal{L}$ le langage $\bar{\mathcal{L}}$ contenant tous les préfixes des mots de $\mathcal{L}$:*

$$\bar{\mathcal{L}} = \{s_1 \in \Sigma^* \mid \exists s_2 \in \Sigma^* \textit{tel que } s_1 s_2 \in \mathcal{L}\}.$$

❑

Remarques 1.2.2.

a) Par définition, $\bar{\mathcal{L}} \supseteq \mathcal{L}$.

b) Un langage $\mathcal{L}$ est dit *préfixe-clos* si $\mathcal{L} = \bar{\mathcal{L}}$.

❑

1.2.2 Opérations avec les langages

Soit Σ un alphabet donné et soient $\mathcal{L}_1$ et $\mathcal{L}_2$ deux langages définis sur Σ. Les opérations élémentaires suivantes peuvent être établies :

- L'*union* des langages $\mathcal{L}_1$ et $\mathcal{L}_2$ est le langage contenant tout mot compris soit dans $\mathcal{L}_1$, soit dans $\mathcal{L}_2$: $\mathcal{L}_\cup = \mathcal{L}_1 \cup \mathcal{L}_2 = \mathcal{L}_1 + \mathcal{L}_2 = \{s \mid s \in \mathcal{L}_1 \text{ ou } s \in \mathcal{L}_2\}$.
- L'*intersection* des langages $\mathcal{L}_1$ et $\mathcal{L}_2$ est le langage contenant tout mot compris à la fois dans $\mathcal{L}_1$ et dans $\mathcal{L}_2$: $\mathcal{L}_\cap = \mathcal{L}_1 \cap \mathcal{L}_2 = \{s \mid s \in \mathcal{L}_1 \text{ et } s \in \mathcal{L}_2\}$.
- La *concaténation* des langages $\mathcal{L}_1$ et $\mathcal{L}_2$ est le langage contenant tout mot construit d'un mot de $\mathcal{L}_1$ suivi d'un mot de $\mathcal{L}_2$: $\mathcal{L}_\bullet = \mathcal{L}_1.\mathcal{L}2 = \{\ s \mid \exists\ s_1 \in \mathcal{L}_1 \text{ et } s_2 \in \mathcal{L}_2 \text{ tel que } s = s_1 s_2\ \}$.
- La *fermeture itérative* d'un langage $\mathcal{L}_1$ est l'ensemble des mots construits par une concaténation finie des mots de $\mathcal{L}_1$: $\mathcal{L}_1^* = \{\ s \mid \exists\ i \in \mathbb{N} \text{ et } s_1, s_2, ..., s_i \in \mathcal{L}_1 \text{ tel que } s = s_1 s_2 ... s_i\ \}$.

1.2.3 Langages réguliers

Les opérations d'union ("+"), de concaténation (".") et de fermeture itérative ("$*$") jouent un rôle important dans la définition d'une classe spéciale de langages — les *langages réguliers* — classe qui est d'un intérêt tout particulier pour la modélisation des SED. Les langages de ce type peuvent être représentés de façon concise par ce qu'on appelle des *expressions régulières* — des expressions dont les opérandes sont des symboles de l'alphabet de définition et dont les opérateurs sont compris dans l'ensemble $\{+, .\ , *\}$. Par exemple l'expression $\mathscr{E} = b + a.b$ est une expression régulière sur l'alphabet $\Sigma = \{a, b\}$.

Afin de faire correspondre un langage unique à chaque expression régulière, les opérateurs sont classés dans l'ordre de priorité suivant : $+$, $.$, $*$, du moins prioritaire au plus prioritaire. Une priorité d'ordre supérieur peut être définie à l'aide des parenthèses. Ainsi, les expressions régulières $\mathscr{E}_1 = a.b^*$ et $\mathscr{E}_2 = (a.b)^*$ définissent des langages différents : $\mathcal{L}_1 = \{\varepsilon, a, ab, abb, abbb...\}$ et, respectivement, $\mathcal{L}_2 = \{\varepsilon, ab, abab, ababab, ...\}$.

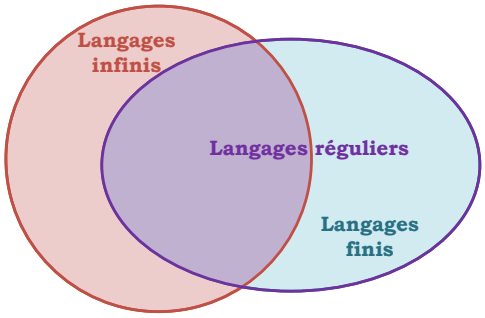


FIGURE 1.2 – Langages finis, infinis et réguliers

Remarques 1.2.3.

a) En général, nous pouvons faire correspondre plusieurs expressions régulières à un même langage régulier.

b) Tout langage fini peut être décrit par une expression régulière. Pourtant, la réciproque n'est pas toujours vraie. Étant donné un alphabet Σ, l'ensemble des langages finis sur Σ est inclus dans l'ensemble des langages réguliers (figure 1.2).

❑

1.3 Automates

Parmi la multitude des outils permettant l'étude des SED, nous allons présenter dans ce chapitre les automates et les réseaux de Petri. Ces deux approches de modélisation et d'analyse des SED sont fortement impliquées dans la recherche sur la synthèse de contrôleurs, qui est le sujet de cette thèse.

1.3.1 Notions fondamentales

L'automate est une machine à états — il est représenté par un ensemble d'états liés par des transitions associées à des événements. Un automate est dit *à états fini* si le nombre d'états est borné. Il peut être vu comme une machine ayant des entrées et des sorties discrètes et qui adapte ses sorties conformément à la modification de ses entrées. Si le prochain état de l'automate peut être déterminé sans équivoque à partir de son état

courant et du symbole d'entrée, alors l'automate est appelé *déterministe*. Autrement, l'automate est dit non-déterministe. Une relation d'équivalence existe entre les deux, dans le sens où pour tout automate fini non-déterministe, un automate déterministe équivalent peut être trouvé.

Le modèle automate permet de décrire le fonctionnement entrées / sorties des SED. Un automate à états fini et déterministe peut être défini formellement comme suit ([Hopcroft et al., 2007], [Cassandras & Lafortune, 2008]) :

Définition 1.3.1. *Un automate fini $\mathcal{A}$ est un 5-tuplet $\mathcal{A} = \{\mathcal{Q}, \Sigma, \delta, q_0, \mathcal{Q}_m\}$, où :*
- *$\mathcal{Q}$ est l'ensemble (fini) d'états,*
- *Σ est l'alphabet d'entrée (l'ensemble des événements),*
- *$\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ est la fonction de transition d'état, qui associe un état d'arrivée, q_k, à un état de départ, q_i, et à un symbole d'entrée, σ_j : $\delta(q_i, \sigma_j) = q_k$,*
- *$q_0 \in \mathcal{Q}$ est l'état initial, et*
- *$\mathcal{Q}_m \subseteq \mathcal{Q}$ est l'ensemble des états marqués (états finals).*

Un automate ne comportant pas de sorties est appelé accepteur : $\mathcal{A} = \{\mathcal{Q}, \Sigma, \delta, q_0\}$.

❑

Remarques 1.3.1.

a) La fonction de transition d'état δ peut être étendue pour associer un état d'arrivée, q_k, à tout état de départ, q_i, et à tout mot s_j sur Σ : $\delta : \mathcal{Q} \times \Sigma^* \to \mathcal{Q}$, $\delta(q_i, s_j) = q_k$.

b) Les *automates infinis* sont des automates où l'ensemble d'états $\mathcal{Q}$ est infini.

❑

Définition 1.3.2. *Le langage généré par $\mathcal{A}$ comprend tous les mots qui permettent de rejoindre un état quelconque de l'automate à partir de son état initial :*

$$\mathcal{L}(\mathcal{A}) = \{s \in \Sigma_{\mathcal{A}}^* \mid \exists\, q_k \in \mathcal{Q} \text{ tel que } \delta_{\mathcal{A}}(q_0, s) = q_k\}.$$

❑

Remarques 1.3.2.

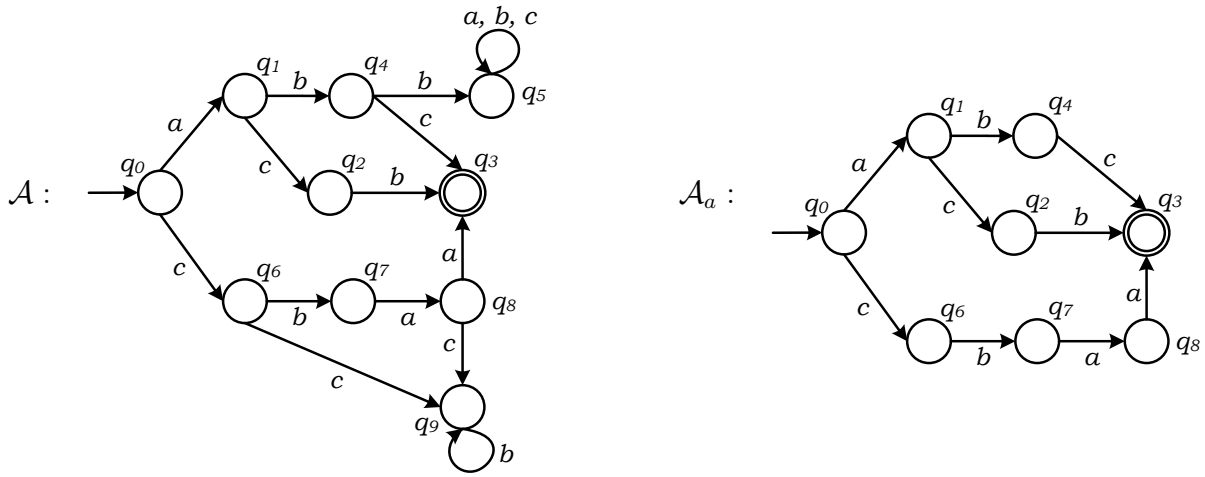
a) Le langage $\mathcal{L}(\mathcal{A})$ est un langage régulier.

b) Le langage généré par un accepteur est préfixe-clos.

❑

Un automate fini peut être décrit par son *graphe de transition d'états*. Dans ce graphe les états sont symbolisés par des cercles et la fonction de transition d'états est représentée par des arcs orientés associés aux événements de Σ. L'état initial est figuré par un cercle avec une flèche entrante et les états finaux sont indiqués par des doubles cercles. La

figure 1.3a présente le graphe de transition d'états d'un automate fini déterministe $\mathcal{A}$. Nous pouvons identifier $\mathcal{Q} = \{q_0, q_1, q_2, ..., q_9\}$, $\Sigma = \{a, b, c\}$, et $\mathcal{Q}_m = \{q_3\}$. La transition d'état $\delta(q_0, a) = q_1$ est figurée par un arc orienté de q_0 à q_1 et libellée par le symbole a. Plusieurs chemins sont possibles pour arriver de l'état initial q_0 à l'état final q_3, comme par exemple : $\delta(q_0, abc)$, ou $\delta(q_0, acb)$, ou bien $\delta(q_0, cbaa)$. On observe aussi que dans le cas de ce modèle la fonction de transition d'état est *partielle*, dès lors qu'elle n'est pas définie pour tout élément de produit cartésien $\mathcal{Q} \times \Sigma$. Par exemple : $\delta(q_0, b) = \varnothing$.



(a) (b)

FIGURE 1.3 – Graphe de transition d'états (a) et automate accepté (b) d'un automate fini

Soit l'automate fini déterministe $\mathcal{A} = \{\mathcal{Q}_\mathcal{A}, \Sigma_\mathcal{A}, \delta_\mathcal{A}, q_0, \mathcal{Q}_{m_\mathcal{A}}\}$. On appelle *état accessible* de $\mathcal{A}$ tout état $q_k \in \mathcal{Q}_\mathcal{A}$ qui peut être atteint par l'automate depuis son état initial et en franchissant les transitions d'état impliquées par un mot $s \in \Sigma^*_\mathcal{A}$. Par extension, l'automate $\mathcal{A}$ est dit accessible si tous ses états sont accessibles.

Un mot s est accepté (ou reconnu) par $\mathcal{A}$ si, en franchissant les transitions d'état impliquées par s depuis l'état initial de l'automate, un état final $q_k \in \mathcal{Q}_{m_\mathcal{A}}$ est atteint. L'ensemble des mots acceptés par un automate constitue le *langage accepté* par l'automate : $\mathcal{L}_a(\mathcal{A}) = \{s \in \Sigma^*_\mathcal{A} \mid \exists\, q_k \in \mathcal{Q}_{m_\mathcal{A}} \text{ tel que } \delta_\mathcal{A}(q_0, s) = q_k\}$ (figure 1.3b).

Remarque 1.3.3. La classe des langages acceptés par les automates finis construits sur un alphabet Σ est égale à la classe des langages réguliers sur Σ.

❑

À partir de maintenant, nous considérerons qu'il n'y a pas d'états privilégiés, et nous ne parlerons plus que des accepteurs.

1.3.2 Composition synchrone

L'opération de *composition* des accepteurs permet l'obtention du modèle global d'un système à partir des modèles de ses sous-systèmes. Si les alphabets associés aux automates considérés ont au moins un événement en commun, la composition est *synchrone*. Si les langages sont disjoints, la composition est *asynchrone*.

Soient deux accepteurs $\mathcal{A}_1 = \{\mathcal{Q}_1, \Sigma_1, \delta_1, q_{1_0}\}$ et $\mathcal{A}_2 = \{\mathcal{Q}_2, \Sigma_2, \delta_2, q_{2_0}\}$. On peut alors écrire :

Définition 1.3.3. *La composition synchrone entre* $\mathcal{A}_1$ *et* $\mathcal{A}_2$ *est l'accepteur* $\mathcal{A}_S = \mathcal{A}_1 \parallel_s \mathcal{A}_2 = \{\mathcal{Q}, \Sigma, \delta, q_0\}$*, où :*

- $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$ *est l'ensemble d'états,*
- $\Sigma = \Sigma_1 \cup \Sigma_2$ *est l'alphabet,*
- $q_0 = (q_{1_0}, q_{2_0})$ *est l'état initial et*
- $\delta : \mathcal{Q} \times \Sigma$ *est la fonction de transition d'état, qui associe à chaque état d'entrée* $q_i = (q_{1_i}, q_{2_i}) \in \mathcal{Q}$ *et à tout événement* $\sigma_j \in \Sigma$ *un état de sortie* q_k *calculé de la façon suivante :*
  - *pour* $\sigma_j \notin \Sigma_1 \cap \Sigma_2$ *:*
    - $\delta\left((q_{1_i}, q_{2_i}), \sigma_j\right) = (q_{1_k}, q_{2_i})$ *si* $\sigma_j \in \Sigma_1$ *et* $\delta_1(q_{1_i}, \sigma_j) = q_{1_k}$,
    - $\delta\left((q_{1_i}, q_{2_i}), \sigma_j\right) = (q_{1_i}, q_{2_k})$ *si* $\sigma_j \in \Sigma_2$ *et* $\delta_2(q_{2_i}, \sigma_j) = q_{2_k}$,
  - *pour* $\sigma_j \in \Sigma_1 \cap \Sigma_2$ *:*
    - $\delta\left((q_{1_i}, q_{2_i}), \sigma_j\right) = (q_{1_k}, q_{2_k})$ *si* $\delta_1(q_{1_i}, \sigma_j) = q_{1_k}$ *et* $\delta_2(q_{2_i}, \sigma_j) = q_{2_k}$.

❑

Remarque 1.3.4. Dans le cas particulier où les alphabets Σ_1 et Σ_2 sont identiques, une séquence est reconnue par $\mathcal{A}_S$ si, et seulement si, elle est reconnue à la fois par $\mathcal{A}_1$ et par $\mathcal{A}_2$:

$$\mathcal{L}_a(\mathcal{A}_S) = \mathcal{L}_a(\mathcal{A}_1) \cap \mathcal{L}_a(\mathcal{A}_2).$$

❑

Exemple 1.3.1
Considérons l'exemple d'un distributeur de jetons : un jeton est libéré seulement si une pièce a préalablement été introduite dans la machine *et* si le bouton adéquat a été appuyé. Le fonctionnement de ce procédé peut être décrit par deux accepteurs modélisant ses deux sous-systèmes : les automates $\mathcal{A}_{\mathcal{D}_1}$ et $\mathcal{A}_{\mathcal{D}_2}$ de la figure 1.4. L'accepteur $\mathcal{A}_{\mathcal{D}_1}$ reconnaît l'ensemble des séquences telles que les événements p = introduction d'une pièce et j = libération du jeton se produisent alternativement, avec p se produisant en premier. De même, $\mathcal{A}_{\mathcal{D}_2}$ reconnaît les séquences où la pression du bouton (l'événement b) précède la libération du jeton (événement j).

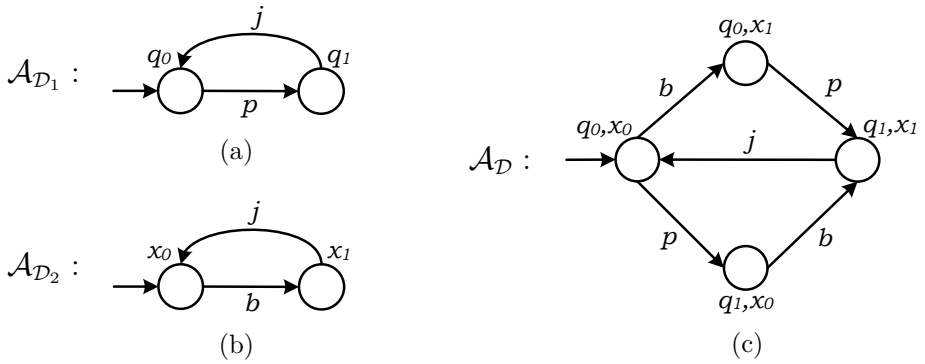


FIGURE 1.4 – Modèle accepteur d'un distributeur des jetons : a) sous-système "pièce → jeton" ; b) sous-système "bouton → jeton" ; c) modèle complet du système.

Le modèle global du distributeur (l'accepteur $\mathcal{A}_{\mathcal{D}}$ de la figure 1.4) est obtenu par la conjonction des deux contraintes de fonctionnement illustrées par $\mathcal{A}_{\mathcal{D}_1}$ et $\mathcal{A}_{\mathcal{D}_2}$, *i.e.* leur composé synchrone : $\mathcal{A}_{\mathcal{D}} = \mathcal{A}_{\mathcal{D}_1} \parallel_s \mathcal{A}_{\mathcal{D}_2}$. Chaque état de cet automate est un couple (q_i, x_j), où q_i est un état de $\mathcal{A}_{\mathcal{D}_1}$ et x_j un état de $\mathcal{A}_{\mathcal{D}_2}$. Son état initial, (q_0, x_0), décrit le distributeur au repos — c'est-à-dire qu'aucune pièce n'a pas été introduite dans la machine (q_0) et que le bouton n'a pas été appuyé (x_0). Le langage de $\mathcal{A}_{\mathcal{D}}$ est défini sur l'union des alphabets $\Sigma_{\mathcal{D}_1} = \{p, j\}$ et $\Sigma_{\mathcal{D}_2} = \{b, j\}$: $\Sigma_{\mathcal{D}} = \{p, b, j\}$, où j est le seul événement qui doit se produire de manière synchrone dans les deux sous-systèmes. Un premier changement d'état arrive alors au moment où une des conditions nécessaires pour la libération d'un jeton est satisfaite : soit une pièce est détectée à l'entrée du distributeur ($q_0 \xrightarrow{p} q_1$), soit le bouton est appuyé ($x_0 \xrightarrow{b} x_1$). Le système attend maintenant l'accomplissement de la deuxième condition pour démarrer le processus de libération du jeton (état (q_1, x_1)). Finalement, une fois le jeton libéré, le système retourne à son état initial.

❑

1.3.3 Machine de Moore

Une machine de Moore est un automate déterministe dans lequel la sortie peut prendre des valeurs dans un ensemble fini de grandeurs discrètes. Par conséquent, la sortie d'une machine de Moore n'est plus nécessairement binaire. Ceci confère à la machine de Moore un pouvoir de description supérieur à celui des modèles accepteurs. Formellement, une machine de Moore est définie comme ci-dessous :

Définition 1.3.4. *Une machine de Moore est un 6-tuplet $\mathcal{A}_{\mathcal{M}} = \{\mathcal{Q}, \Sigma, \delta, q_0, \Gamma, \lambda\}$, où :*

- *$\mathcal{Q}$, Σ, δ et q_0 gardent la même signification que dans la définition 1.3.1,*

- Γ *est l'alphabet (fini) de sortie, et*
- $\lambda : \mathcal{Q} \to \Gamma$ *est la fonction de transition d'affectation de sortie.*

❑

L'automate $\mathcal{A}_\mathcal{M}$ évolue, conformément à sa fonction de transition d'états δ, sur occurrence d'une modification de ses entrées (symboles de Σ). La fonction λ affecte une sortie de Γ pour chaque état de l'automate.

Un exemple de machine de Moore est donné dans la figure 1.5 : les entrées de $\mathcal{A}_\mathcal{M}$ sont des chaînes construites sur l'alphabet $\Sigma = \{a, b\}$, alors que ses sorties peuvent prendre des valeurs dans l'ensemble $\Gamma = \{0, 1, 2\}$. Le modèle comporte trois états, q_0, q_1 et q_2, chacun correspondant à une sortie de l'automate. Par exemple, la séquence d'entrée $\mathcal{S} = aaba$ conduit la machine depuis son état initial q_0, où la sortie de l'automate est égale à 0, vers l'état q_1, où la sortie a la valeur 1.

Dans cet exemple nous pouvons remarquer que la machine $\mathcal{A}_\mathcal{M}$ permet la séparation de ses chaînes d'entrée en trois classes — la sortie de $\mathcal{A}_\mathcal{M}$ est :

- 0 pour les chaînes se terminant par un b,
- 1 pour la chaîne a et pour les chaînes se terminant par ba, et
- 2 pour les chaînes se terminant par deux symboles a successifs, ou plus.

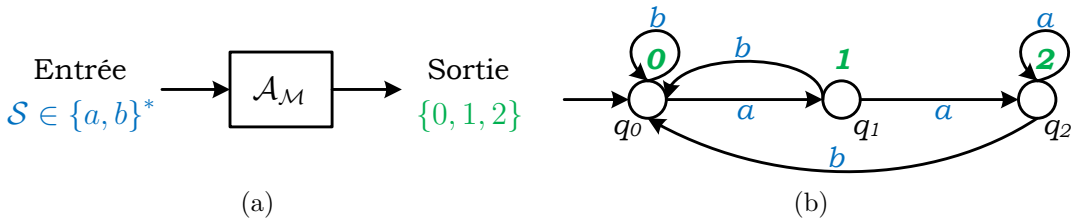


FIGURE 1.5 – Une machine de Moore : a) schéma entrées / sorties ; b) modèle automate.

1.4 Réseaux de Petri

Malgré le fait que les automates représentent un puissant formalisme de modélisation des SED, leurs avantages sont néanmoins atténués par leur grande sensibilité au problème de l'explosion combinatoire du nombre d'états (même dans le cas de systèmes simples). La modélisation par des réseaux de Petri (RdP) offre la possibilité de dépasser cet inconvénient, en fournissant des structures de modélisation beaucoup plus riches et un net gain d'échelle. En outre, les RdP sont aussi plus adaptés à la modélisation des activités distribuées au sein des systèmes complexes, permettant une représentation beaucoup plus concise et claire des différents sous-systèmes impliqués et de leurs interdépendances.

Cette section rappelle les notions de base sur les RdP. Pour plus de détails on pourra se reporter à [Giua & Seatzu, 2007], [David & Alla, 1992], [David & Alla, 2010], ou [Cassandras & Lafortune, 2008].

1.4.1 Définitions

Définition 1.4.1. *Un réseau de Petri est un graphe bipartite orienté.*

❑

Le réseau de Petri est un graphe orienté comportant un ensemble fini de places et un ensemble fini de transitions, avec des arcs orientés qui assurent le passage place $\rightleftarrows$ transition. Les places (symbolisées par des cercles) représentent des conditions spécifiques pour chaque état du système — *i.e.* des informations de type nombre de ressources, état d'une ressource, etc. Les transitions (symbolisées par des traits) représentent les actions qui peuvent provoquer le changement d'état du système (les événements discrets). Les arcs orientés indiquent les relations causales entre les états du système et les événements impliqués dans son évolution. Comme chaque changement d'état est causé par l'arrivée d'un événement, il s'ensuit logiquement qu'un arc ne peut jamais lier deux nœuds de même type.

Si le changement d'état du système est associé au moins une fois à un événement autre que l'événement toujours occurrent, ε, alors le réseau est appelé *synchronisé*. C'est aux réseaux synchronisés que nous nous intéressons dans la suite de ce travail.

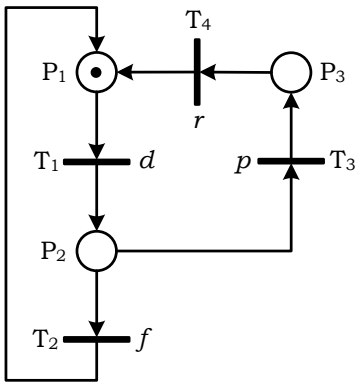


FIGURE 1.6 – Modèle RdP d'une machine à trois états (exemple 1.1.1)

Un exemple de réseau de Petri synchronisé est illustré dans la figure 1.6. C'est le modèle du système présenté dans l'exemple 1.1.1. L'ensemble des places est $\mathcal{P} = \{P_1, P_2, P_3\}$, avec P_1 représentant l'état d'arrêt, P_2 l'état de marche, et P_3 l'état de panne. Les transitions, $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$, correspondent, respectivement, au changements d'état

associés aux événements d = début de cycle d'usinage, f = fin de cycle d'usinage, p = panne et r = réparation.

Évidemment, la structure du graphe toute seule ne suffit pas pour complètement modéliser un système avec un réseau de Petri. Pour bien décrire la dynamique du système, il faut encore donner des informations sur son état. À cette fin, nous utilisons des jetons dans les places du graphe, pour signaler l'état de chaque ressource à un moment donné. Chaque place contiendra alors un nombre de jetons et l'état courant du système sera ainsi modélisé par le vecteur de marquage du réseau : $M_i = [m_{i_1}\ \ m_{i_2}\ \ ...\ \ m_{i_n}]^T$, où $m_{i_j} = M_i(P_j)$, $j = \overline{1,n}$, dénote le nombre de jetons (le *marquage*) contenu dans la place P_j quand le système est dans l'état M_i, et où $n \in \mathbb{N}^*$ est le nombre de places du réseau. Par exemple, pour le système de la figure 1.6, nous remarquons qu'il y a un seul jeton, dans la place P_1 — la machine est à l'arrêt. L'état du réseau est alors : $M_0 = [1\ \ 0\ \ 0]^T$.

L'ensemble de places marquées d'un état quelconque d'un RdP est donné par une fonction support définie comme suit :

Définition 1.4.2. *La fonction Support(X) d'un vecteur $X \in \mathbb{N}^*$ est telle que :*

$$Support(X) = \text{ L'ensemble de places marquées dans le vecteur } X.$$

❑

Par exemple, le support du vecteur $M_0 = [\ 1\ \ 0\ \ 3\ \ 0\ \ 0\ \ 7\ \ 0\ ]^T$ est $Support(M_0) = \{P_1, P_3, P_6\}$ ou, plus simplement, $Support(M_0) = P_1P_3P_6$.

Proposition 1.4.1. *Soit M_1 et M_2 deux marquages d'un RdP $\mathcal{R}$ ayant l'ensemble des places $\mathcal{P}$, de cardinalité* Card$[\mathcal{P}] = n$. *Les affirmations suivantes décrivent les relations possibles entre M_1 et M_2 :*

1. $M_1 > M_2 \Leftrightarrow \forall P_i \in \mathcal{P} : M_1(P_i) \geq M_2(P_i)$, *et* $\exists\ P_i \in \mathcal{P} : M_1(P_i) > M_2(P_i)$.
2. $M_1 < M_2 \Leftrightarrow \forall P_i \in \mathcal{P} : M_1(P_i) \leq M_2(P_i)$, *et* $\exists\ P_i \in \mathcal{P} : M_1(P_i) < M_2(P_i)$.
3. $M_1 = M_2 \Leftrightarrow \forall P_i \in \mathcal{P} : M_1(P_i) = M_2(P_i)$.

❑

Remarque 1.4.1. Si $Support(M_1) \subset Support(M_2)$ alors le marquage M_2 est couvert par le marquage M_1.

❑

Dans la suite de ce rapport, nous allons utiliser une écriture simplifiée, où l'état courant du système est symbolisé par une association entre son support et son marquage : $M_i = P_{i_1}^{m_{i_1}} P_{i_2}^{m_{i_2}} ... P_{i_t}^{m_{i_t}}$, où $t \leq n$ est le nombre de places marquées quand le système est dans l'état M_i. Nous pouvons alors dire que le réseau de la figure 1.6 est dans l'état $M_0 = P_1^1 = P_1$. Pour formaliser cette idée dans le cas général :

Notation 1.4.1. Soit $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ l'ensemble des places d'un RdP $\mathcal{R}$ et soit $M = [m_1 \ \ m_2 \ \ ... \ \ m_n]^T$, $m_i \in \mathbb{N}$, un marquage quelconque de $\mathcal{R}$. Le marquage M peut être représenté comme suit : $M = \{P_i^{m_i} \mid i = \overline{1,n}$, avec $m_i \geq 1\}$.

❑

Ayant présenté les éléments et les conventions de modélisation, nous pouvons maintenant définir de manière formelle le RdP synchronisé :

Définition 1.4.3. *Un réseau de Petri synchronisé est un 7-uplet* $\mathcal{R} = \{\mathcal{P}, \mathcal{T}, \Sigma, E, \mathcal{W}^+, \mathcal{W}^-, M_0\}$, *où :*

- $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ *est l'ensemble (fini) des places,*
- $\mathcal{T} = \{T_1, T_2, ..., T_k\}$ *est l'ensemble (fini) des transitions, avec* $\mathcal{P} \cap \mathcal{T} = \varnothing$,
- $\Sigma = \{\varepsilon, \sigma_1, \sigma_2, ..., \sigma_l\}$ *est l'alphabet (l'ensemble des événements), avec* ε *signifiant l'événement toujours occurrent,*
- $E : \mathcal{T} \to \Sigma$ *est la fonction qui associe à chaque transition* T_i *un événement,* σ_j,
- M_0 *est l'état initial,*
- $\mathcal{W}^+ : \mathcal{P} \times \mathcal{T} \to \{0, 1\}$ *est l'application d'incidence avant :*
  - *si* $\mathcal{W}^+(P_i, T_j) = 0$ *il n'y a pas d'arc de* P_i *à* T_j, *i.e. la condition* P_i *n'influence pas l'événement* T_j,
  - *autrement, si* $\mathcal{W}^+(P_i, T_j) = 1$, *il y a une causalité* $P_i \to T_j$,
- $\mathcal{W}^- : \mathcal{T} \times \mathcal{P} \to \{0, 1\}$ *est l'application d'incidence arrière :*
  - *si* $W^-(P_i, T_j) = 1$, *l'arrivée de l'événement* T_j *influence la variable d'état* P_i.

❑

Remarque 1.4.2. Un réseau de Petri est dit *pur* s'il ne présente pas de boucles, *i.e.* $\nexists\ (P_i, T_j) \in \mathcal{R}$ telles que $\mathcal{W}^+(P_i, T_j) = \mathcal{W}^-(P_i, T_j) \neq 0$. Pour le cas des RdP purs, la matrice d'incidence générale, $\mathcal{W} = \mathcal{W}^- - \mathcal{W}^+$, peut être utilisée au lieu de $\mathcal{W}^+$ et $\mathcal{W}^-$.

❑

Nous avons jusqu'ici parlé des RdP ordinaires. Cependant, pour alléger le graphisme et permettre l'utilisation des modèles RdP pour un plus grand nombre d'applications, le formalisme de modélisation a été enrichi avec des abréviations et des extensions. Une abréviation est une représentation graphique simplifiée de son RdP ordinaire correspondant. Il s'ensuit que toutes les propriétés des RdP ordinaires sont conservées par les abréviations. Par contre, les extensions sont des modèles ayant des règles de fonctionnement différentes des RdP ordinaires et qui ne conservent qu'une partie des propriétés de ces derniers.

Dans ce travail nous portons un intérêt particulier à une abréviation : le RdP généralisé. Un réseau de Petri généralisé est un RdP dans lequel des poids (nombres entiers strictement positifs) sont associés aux arcs. Les arcs dont le poids n'est pas explicitement spécifié sont égaux à 1.

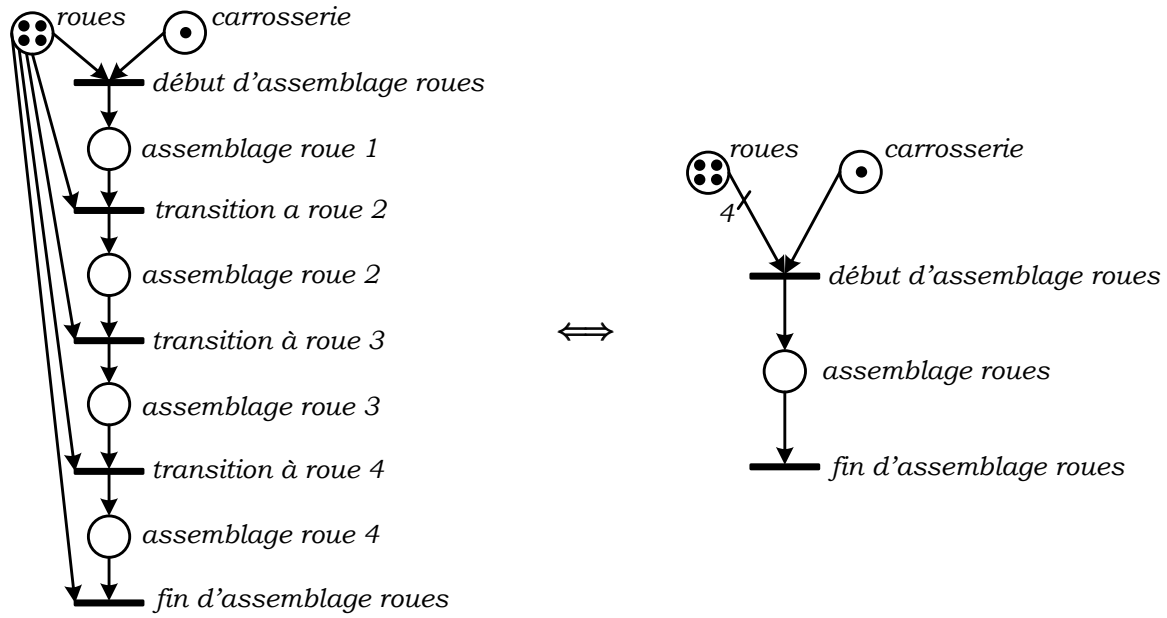


FIGURE 1.7 – Modèles RdP ordinaire et généralisé d'une chaîne d'assemblage de voitures : a) RdP ordinaire ; b) RdP généralisé.

Considérons l'exemple d'une machine qui exécute l'assemblage des roues d'une voiture. La figure 1.7 donne les modèles RdP ordinaire et généralisé de ce système. La signification des places et des transitions est explicitée sur cette même figure. La machine ne peut pas commencer l'assemblage avant que la carrosserie et toutes les roues soient disponibles. Il faut 4 roues, d'où le poids 4 de l'arc décrivant la causalité entre le nombre des roues disponibles et le début de l'assemblage dans la figure 1.7b.

Formellement, un RdP généralisé synchronisé est défini comme suit :

Définition 1.4.4. *Un réseau de Petri généralisé synchronisé est un 7-uplet* $\mathcal{R} = \{\mathcal{P}, \mathcal{T}, \Sigma, E, \mathcal{W}^+, \mathcal{W}^-, M_0\}$, *où :*

- $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ *est l'ensemble (fini) des places,*
- $\mathcal{T} = \{T_1, T_2, ..., T_k\}$ *est l'ensemble (fini) des transitions, avec* $\mathcal{P} \cap \mathcal{T} = \varnothing$,
- $\Sigma = \{\varepsilon, \sigma_1, \sigma_2, ..., \sigma_l\}$ *est l'alphabet d'entrée (l'ensemble des événements), avec* ε *signifiant l'événement toujours occurrent,*
- $E : \mathcal{T} \to \Sigma$ *est la fonction qui associe à chaque transition* T_i *un événement,* σ_j,
- M_0 *est l'état initial,*
- $\mathcal{W}^+ : \mathcal{P} \times \mathcal{T} \to \mathbb{N}$ *est l'application d'incidence avant :*
  - *si* $\mathcal{W}^+(P_i, T_j) = 0$, *il n'y a pas d'arc de* P_i *à* T_j, *i.e. la condition* P_i *n'influence pas l'événement* T_j,
  - *autrement, si* $\mathcal{W}^+(P_i, T_j) = w_{ij}$, *il y a une causalité* $P_i \xrightarrow{w_{ij}} T_j$, *et* w_{ij} *est le poids de l'arc décrivant cette causalité,*

- $\mathcal{W}^- : \mathcal{T} \times \mathcal{P} \to \mathbb{N}$ *est l'application d'incidence arrière :*
  - *si* $W^-(P_i, T_j) = w_{ij} > 0$*, l'arrivée de l'événement* T_j *influence la variable d'état* P_i*, et* w_{ij} *est le poids de l'arc décrivant cette causalité.*

❑

Remarque 1.4.3. Tout RdP généralisé a un RdP ordinaire correspondant.

❑

1.4.2 Franchissement des transitions

Comme dans un RdP les événements sont associés aux transitions, il s'ensuit naturellement que l'évolution du système est caractérisée par le franchissement des transitions. Cependant, pour qu'une transition puisse se produire, il faut que toutes les conditions nécessaires soient satisfaites. Les informations concernant ces conditions sont contenues dans le marquage de l'ensemble des places situées en amont de la transition concernée — les places d'entrée de la transition. L'évolution de l'état correspond donc à une évolution du marquage.

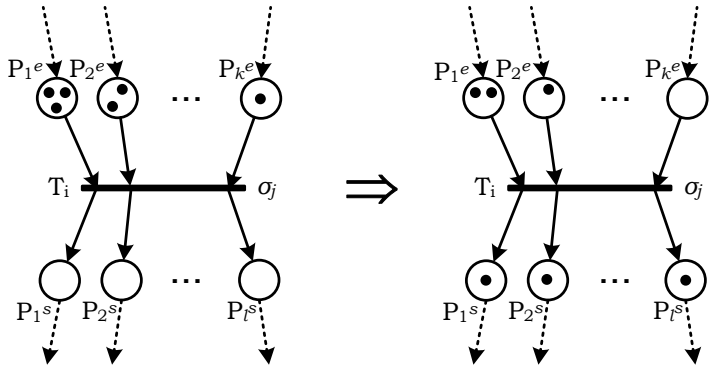


FIGURE 1.8 – Le franchissement d'une transition dans un RdP ordinaire

L'illustration d'un franchissement générique de transition est donnée dans la figure 1.8. Les places d'entrée sont numérotées avec l'indice supérieur "e". La transition ne peut être franchie que si chacune de ses places d'entrée contient au moins un jeton. La transition est alors appelé *franchissable* (ou *validée*). Supposant que les places d'entrée soient toutes marquées, la transition sera alors franchie au moment de l'occurrence de son événement associé, σ_j. L'occurrence de l'événement entraînera la modification des conditions d'état influencées par celui-ci. Les informations liées à ces conditions sont également contenues dans le marquage des places situées en aval de la transition — ses places de sortie, indiquées sur la figure par l'indice supérieur "s". L'effet du franchissement

sera alors d'enlever un jeton de chaque place d'entrée de la transition et d'ajouter un jeton dans chaque place de sortie.

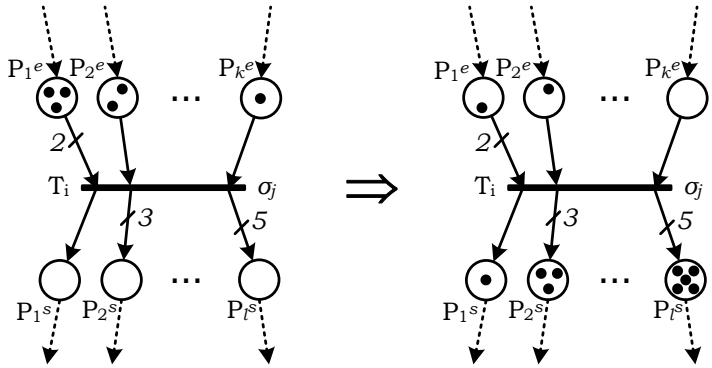


FIGURE 1.9 – Le franchissement d'une transition dans un RdP généralisé

Encore une fois, nous avons présenté la démarche pour le cas des RdP ordinaires. Si le réseau est généralisé, les conditions de franchissement dépendent du poids des arcs. Lorsqu'un arc $P_i \xrightarrow{w_{ij}} T_j$ a un poids w_{ij}, cela signifie que la transition T_j ne sera validée que si la place P_i contient au moins w_{ij} jetons. Lors du franchissement de cette transition, w_{ij} jetons seront retirés de la place P_i. Lorsqu'un arc $T_j \xrightarrow{w_{ij}} P_i$ a un poids w_{ij}, cela signifie que lors du franchissement de T_j, w_{ij} jetons seront ajoutés à la place P_i. Le concept est illustré dans la figure 1.9.

Remarque 1.4.4. Dans un RdP autonome il y a un seul franchissement à la fois et la durée de franchissement est nulle.

❑

Notation 1.4.2. On va noter avec $M_u \,|\sigma_j\, \rangle M_v$ le passage de l'état M_u vers l'état M_v en franchissant la transition associée à l'événement σ_j. Il est également possible de décrire de cette manière une séquence de franchissement $\mathcal{S}$: $M_u \,|\mathcal{S}\, \rangle M_v$.

❑

L'évolution d'état associée au franchissement d'une transition est caractérisée mathématiquement par l'*équation d'état* — l'équation vectorielle qui permet de déterminer l'état suivant du réseau à partir de son état courant et du *vecteur de franchissement*, φ. Le vecteur de franchissement est un vecteur colonne de dimension k (nombre de transitions) qui décrit le franchissement d'une transition T_j donnée, $j = \overline{1,k}$: $\varphi = [0 \quad ... \quad 0 \quad 1 \quad 0 \quad ... \quad 0]^T$, où le seul élément non nul est l'élément de la position j. L'équation d'état $M_v = M_u + \mathcal{W} \cdot \varphi$ décrit le franchissement $M_u \,|T_j\, \rangle M_v$.

En général, si une séquence de franchissement $\mathcal{S}$ est réalisable à partir d'un état M_u donné, le marquage atteint, M_v, est déterminé par l'équation d'état suivante :

$$M_v = M_u + \mathcal{W} \cdot \varphi_{\mathcal{S}},$$

où $\varphi_{\mathcal{S}}$ est le vecteur de franchissement de la séquence $\mathcal{S}$. Chaque composante j de $\varphi_{\mathcal{S}}$ correspond au nombre des franchissements de la transition T_j dans la séquence $\mathcal{S}$.

1.4.3 Graphe de marquages

Définition 1.4.5. *Étant donné un RdP $\mathcal{R}$ et son marquage initial M_0, son ensemble d'états accessibles, $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$, est défini comme l'ensemble de tous les marquages atteignables à partir de M_0 par le franchissement de séquences des transitions :*

$$\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} = \{ M_v \in \mathbb{N}^n | \; \exists \, \mathcal{S} \in \mathcal{T}^* \text{ tel que } M_0 | \mathcal{S} \rangle M_v \} \, . \tag{1.4.1}$$

❑

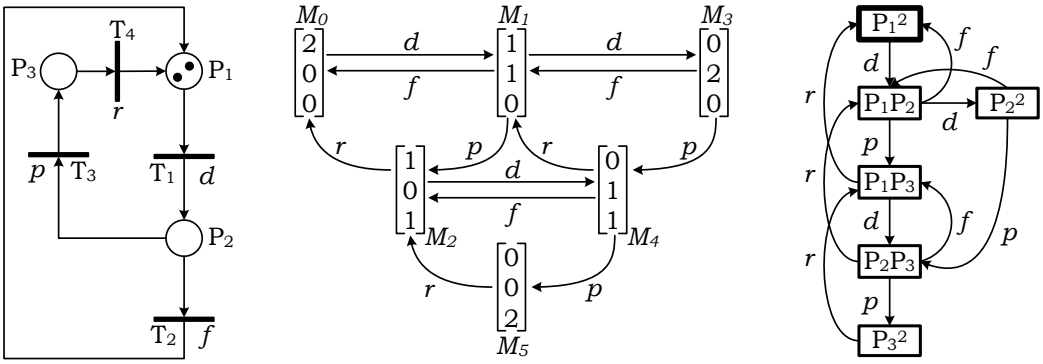
L'ensemble des marquages accessibles peut être représenté sous forme de graphe, ayant les marquages (états) accessibles correspondant aux nœuds et les transitions aux arcs. Ce graphe, appelé le graphe des marquages accessibles, ou simplement le graphe de marquage (GM), correspond au modèle automate (sans états marqués) du système. Formellement, le graphe de marquage d'un RdP $\mathcal{R}$ est défini comme suit :

Définition 1.4.6. *Un graphe de marquage est un 4-uplet $\mathcal{G} = \{\mathcal{M}, \Sigma, \delta, M_0\}$, où :*

- *$\mathcal{M}$ est l'ensemble (fini) des marquages,*
- *Σ est l'ensemble des événements associés aux transitions,*
- *$\delta : \mathcal{M} \times \Sigma \to \mathcal{M}$ est la fonction de transition d'état : $\delta(M_u, \sigma_j) = M_v$, et*
- *$M_0 \in \mathcal{M}$ est l'état initial.*

❑

Le graphe de marquage explore toutes les évolutions possibles du système, en examinant à chaque pas la liste complète des transitions franchissables. La figure 1.10 présente le graphe de marquage d'une machine à trois états ayant une capacité d'usinage de deux pièces à la fois. Le modèle RdP du système est illustré par la figure 1.10a. La signification des places et des transitions est la même que pour l'exemple 1.1.1. Il est supposé ici que la machine contient deux stations d'usinage indépendantes qui travaillent en parallèle. La gamme de fabrication de la pièce est identique sur chaque station de fonctionnement : une fois l'usinage commencé (T_1/d est franchie), la station peut soit finir son cycle et retourner vers son état initial d'arrêt (T_2/f se produise), soit tomber en panne (T_3/p survient). La seule possibilité d'évolution pour les stations en panne est la réparation (T_4/r est franchie), qui ramène la station en question à son état initial.



(a) (b) (c)

FIGURE 1.10 – Modèle RdP et graphe de marquage pour une machine à trois états capable d'usiner deux pièces à la fois : a) modèle RdP ; b) GM, représentation vectorielle ; c) GM, représentation par état.

En partant de l'état initial, $M_0 = [2 \quad 0 \quad 0]^T$, la machine ne peut évoluer que vers l'état de marche. La transition T_1 (événement d) est alors franchie et le marquage évolue vers $M_1 = [1 \quad 1 \quad 0]^T$ — la première station d'usinage commence son cycle. La machine peut commencer l'usinage d'une deuxième pièce sur l'autre station à tout moment du cycle de fonctionnement de la première station. Chacune des boucles $\left(M_0 \overset{d}{\underset{f}{\rightleftharpoons}} M_1 \xrightarrow{p} M_2 \xrightarrow{r} M_0\right)$, $\left(M_1 \overset{d}{\underset{f}{\rightleftharpoons}} M_3 \xrightarrow{p} M_4 \xrightarrow{r} M_1\right)$ et $\left(M_2 \overset{d}{\underset{f}{\rightleftharpoons}} M_4 \xrightarrow{p} M_5 \xrightarrow{r} M_2\right)$ représente le scénario d'une pièce sur une station d'usinage.

1.4.4 Propriétés

Les RdP présentent un riche ensemble des propriétés intéressantes pour la modélisation des systèmes manufacturiers et du partage des ressources. Nous présentons, dans la suite, les propriétés qui nous intéressent en particulier dans le cadre de ce travail :

RdP borné :

Une place $P_i \in \mathcal{P}$ d'un RdP $\mathcal{R}$ est dite *b-bornée*, $b \in \mathbb{N}$, pour un état initial M_0 donné si $M(P_i) \leq b$ pour tout état accessible $M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$. Le réseau $\mathcal{R}$ est appelé borné si toutes ses places sont bornées.

Remarques 1.4.5.

a) Un RdP borné pour tout marquage initial (fini) est appelé *structurellement borné*.

b) Un RdP est appelé *binaire* (ou *sauf*) pour un marquage initial M_0 donné si toutes ses places sont 1-bornées (contiennent au plus un jeton).

c) Pour les RdP saufs, $M_1 < M_2 \Leftrightarrow M_1$ couvre M_2.

❑

Transition vivante :

Une transition T_j est dite vivante pour un marquage initial M_0 si, pour tout marquage accessible $M_u \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$, il existe une séquence de franchissement $\mathcal{S}$, avec $T_j \in \mathcal{S}$, telle que : $M_u | \mathcal{S} \rangle M_v$, $M_v \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$. Si T_j peut être franchie au moins une fois à partir de l'état initial, T_j est dite quasi-vivante. Autrement, elle est non-vivante.

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

Persistance :

Un RdP est appelé persistent si, pour n'importe quelle paire de transitions validées, le franchissement de l'une ne rend pas l'autre infranchissable.

Blocage :

Un blocage est un marquage tel qu'aucune transition n'est pas validée.

Conflit structurel :

Une place $P_i \in \mathcal{P}$ constitue un conflit structurel si elle est une place d'entrée pour au moins deux transitions.

Conflit effectif :

Un conflit structurel devient effectif quand son marquage est inférieur au nombre de ses transitions de sortie validées par ce marquage.

Conservation et invariants de marquage :

Le partage des ressources dans les systèmes manufacturiers constitue un paradigme usuel de modélisation SED. Il est nécessaire, dans ce cas là, de représenter la distribution d'un nombre fini et constant de ressources entre les différents sous-systèmes qui en ont besoin. Une situation similaire concerne l'illustration de l'alternance d'état d'une

ressource — un seul jeton, représentant la ressource, circule entre les places désignant ses états possibles. Dans les deux cas, le nombre de jetons dans l'ensemble des places modélisant un certain chemin de fonctionnement doit rester constant à tout moment. Ces jetons ne peuvent jamais être ni perdus, ni multipliés, ni envoyés vers des places décrivant d'autres chemins de fonctionnement du système. En effet, le réseau doit *conserver* un nombre fixe de jetons dans tous les états accessibles du chemin — d'où la propriété de *conservation*.

Définition 1.4.7. *Un ensemble de places exhibant la propriété de conservation du marquage est appelé composante conservative.*

❑

Exprimée de cette manière, la propriété est très contraignante et ne peut pas être associée à des chemins décrivant des fonctionnements complexes, comme entre les deux situations présentées auparavant. Pour prendre en compte ce type de chemins, il faut définir la propriété de conservation en se rapportant à la pondération des marquages concernés :

Définition 1.4.8. *Étant donné un RdP* $\mathcal{R} = \{\mathcal{P}, \mathcal{T}, \Sigma, E, \mathcal{W}^+, \mathcal{W}^-, M_0\}$*, un ensemble de places* $\mathcal{P}_{\circlearrowleft} = \{P_{\circlearrowleft_1}, P_{\circlearrowleft_2}, ..., P_{\circlearrowleft_r}\}$*,* $\mathcal{P}_{\circlearrowleft} \in \mathcal{P}$*,* $r \leq n$*, et un vecteur de nombres entiers* $x^T = [x_1 \quad x_2 \quad ... \quad x_r]$ *(le vecteur de pondération), on dit que* $\mathcal{P}_{\circlearrowleft}$ *est une composante conservative (par rapport à* x^T*) si :*

$$\sum_{i=1}^{r} x_i \cdot M(P_{\circlearrowleft_i}) = constante = \sum_{i=1}^{r} x_i \cdot M_0(P_{\circlearrowleft_i}) \tag{1.4.2}$$

pour tout marquage accessible $M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$*.*

La relation (1.4.2) *constitue ce qu'on nomme un invariant de marquage.*

❑

Par exemple, pour la machine à trois états modélisée dans la figure 1.10, nous pouvons identifier l'invariant de marquage $m_1 + m_2 + m_3 = 2$.

Remarques 1.4.6.

a) Généralement, la propriété de conservation correspond aux propriétés physiques du système modélisé. Pour cette raison, des nombres naturels sont normalement utilisés pour les poids x_i. Il est pourtant possible pour un invariant de marquage de contenir aussi des poids négatifs, $x_i \in \mathbb{Z}_-$.

b) Si $\mathcal{P}$ est une composante conservative, le RdP est dit conservatif.

c) L'invariant de marquage est une propriété structurelle du réseau. C'est juste la valeur de la constante qui est dépendante du marquage initial.

❑

Le vecteur des poids est déterminé à partir de l'équation : $x^T \cdot \mathcal{W} = 0$, où $\mathcal{W}$ est la matrice d'incidence du réseau et x est un P-semiflot de dimension $n = \text{Card}[\mathcal{P}]$. Nous obtenons alors l'invariant de marquage : $x^T \cdot M_0 = x^T \cdot M = \textit{constante}$, où M_0 est le marquage initial et M est un marquage accessible quelconque. Une fois les invariants minimaux obtenus, il est possible d'étudier la structure du réseau indépendamment de toute évolution dynamique.

1.4.5 Composition synchrone

Nous avons présenté auparavant la composition synchrone des automates. Une opération similaire existe aussi pour les RdP. La synchronisation des réseaux de Petri est une opération structurelle qui implique la fusion des transitions synchronisées sur le même événement :

Définition 1.4.9. *Soit deux réseaux de Petri,* $\mathcal{R}_1 = \left\{\mathcal{P}_1, \mathcal{T}_1, \Sigma_1, E_1, \mathcal{W}_1^+, \mathcal{W}_1^-, M_{0_1}\right\}$ *et* $\mathcal{R}_2 = \left\{\mathcal{P}_2, \mathcal{T}_2, \Sigma_2, E_2, \mathcal{W}_2^+, \mathcal{W}_2^-, M_{0_2}\right\}$*. La composition synchrone entre* $\mathcal{R}_1$ *et* $\mathcal{R}_2$ *est le RdP* $\mathcal{R}_S = \mathcal{R}_1 \parallel_s \mathcal{R}_2 = \left\{\mathcal{P}_S, \mathcal{T}_S, \Sigma_S, E_S, \mathcal{W}_S^+, \mathcal{W}_S^-, M_{0_S}\right\}$*, où :*

- $\mathcal{P}_S = \mathcal{P}_1 \cup \mathcal{P}_2$,
- $\Sigma_S = \Sigma_1 \cup \Sigma_2$,
- $\mathcal{T}_S = \mathcal{T}_1 \cup \mathcal{T}_2 - \mathcal{T}_{12}$, $\mathcal{T}_{12} = \{T_i \in \mathcal{T}_1 |\ \exists\ T_j \in \mathcal{T}_2$ *telle que* $E_1(T_i) = E_2(T_j)\}$,
- $E_S(T_j) = \begin{cases} E_1(T_j), & si\ T_j \in \mathcal{T}_1 \\ E_2(T_j), & si\ T_j \in \mathcal{T}_2 \end{cases}$,
- $\mathcal{W}_S^+(P_i, T_j) = \begin{cases} \mathcal{W}_1^+(P_i, T_j), & si\ P_i \in \mathcal{P}_1 \\ \mathcal{W}_2^+(P_i, T_j), & si\ P_i \in \mathcal{P}_2 \\ 0, & si((P_i \in \mathcal{P}_1) \wedge (T_j \notin \mathcal{T}_1)) \vee ((P_i \in \mathcal{P}_2) \wedge (T_j \notin \mathcal{T}_2)) \end{cases}$,
- $\mathcal{W}_S^-(P_i, T_j) = \begin{cases} \mathcal{W}_1^-(P_i, T_j), & si\ P_i \in \mathcal{P}_1 \\ \mathcal{W}_2^-(P_i, T_j), & si\ P_i \in \mathcal{P}_2 \\ 0, & si((P_i \in \mathcal{P}_1) \wedge (T_j \notin \mathcal{T}_1)) \vee ((P_i \in \mathcal{P}_2) \wedge (T_j \notin \mathcal{T}_2)) \end{cases}$,
- $M_{0_S}(P_i) = \begin{cases} M_{0_1}(P_i), & si\ P_i \in \mathcal{P}_1 \\ M_{0_2}(P_i), & si\ P_i \in \mathcal{P}_2 \end{cases}$.

❑

Remarques 1.4.7.

a) Contrairement aux automates, dont la composition provoque la croissance exponentielle de l'espace d'états ($\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$), la composition synchrone des réseaux de Petri est caractérisée par la croissance linéaire du nombre de places ($\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$). Elle nécessite donc un effort de calcul réduit par rapport à celui demandé par la composition d'automates. Cependant, le graphe de marquage du RdP obtenu après la composition est le même que l'automate obtenu *via* la composition synchrone des

graphes de marquage des composants. Cette propriété fait du RdP un excellent moyen pour la modélisation de manière intuitive et compacte des systèmes complexes.

b) La croissance du nombre de transitions du RdP composé est, elle aussi, plus faible que celle enregistrée pour la composition des automates. En fait, pour les RdP seule l'opération de synchronisations peut causer une augmentation non-linéaire du nombre de transitions par rapport à la taille des modèles composés.

❑

Exemple 1.4.1
Considérons, par exemple, deux machines, $\mathscr{M}_1$ et $\mathscr{M}_2$, de type décrit dans la figure 1.6, et supposons qu'elles travaillent en tandem : chaque pièce doit être premièrement usinée par $\mathscr{M}_1$ et puis par $\mathscr{M}_2$, avec un stock de capacité unitaire entre les deux. Le modèle complet du système peut être obtenu par la synchronisation des modèles correspondants aux deux sous-systèmes : les machines et le stock (figure 1.11).

❑

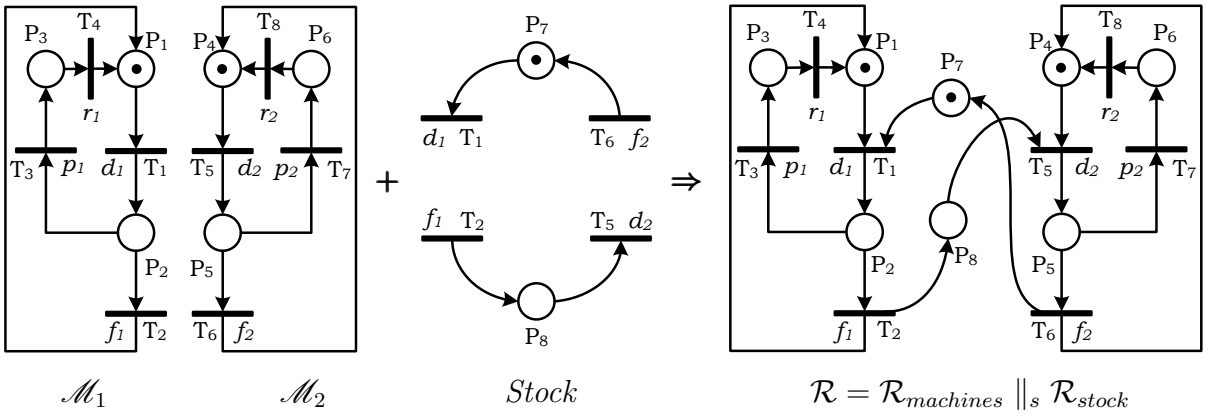


FIGURE 1.11 – Modèle RdP d'une chaine manufacturière obtenue en synchronisant deux machines et un stock de capacité unitaire

1.5 Conclusions

Nous avons présenté dans ce chapitre un bref synopsis de la théorie générale des SED, en nous focalisant sur les notions et les propriétés que nous allons utiliser dans la suite de ce mémoire. Nous avons vu quel type de système peut être considéré comme un SED et que ce sont les automates et les réseaux de Petri qui sont les plus importants formalismes mathématiques qui permettent leur modélisation et l'analyse de leur comportement. Nous avons discuté les avantages et inconvénients de point de vue modélisation de ces deux approches de base, et nous avons examiné les implications de

l'explosion combinatoire du nombre d'états. Nous avons également étudié les possibilités de composition de plusieurs modèles en vue d'obtenir une vision du comportement global d'un système complexe à travers l'analyse de ses composantes. Finalement, nous avons conclu que les réseaux de Petri, avec leurs structures de modélisation très riches et leur concision supérieure, sont les plus adaptés à la description des SED. De plus, ils permettent un passage systématique vers l'automate, ce qui rend possible de profiter totalement des propriétés des deux modèles.

Toutes ces notions représentent la base qui nous permettra, d'un côté, d'expliquer la théorie du contrôle et de la supervision des SED et de l'autre côté de poursuivre notre travail sur la synthèse des contrôleurs discrets (ce qui est l'objectif des chapitres suivants).

Chapitre 2

Synthèse de contrôleurs

Ce chapitre présente les approches les plus connues de contrôle des SED. Pour commencer, nous clarifions les concepts fondamentaux et le schéma de supervision. Nous présentons ensuite les bases de la théorie générale de la commande par supervision des SED, développée par Ramadge et Wonham. Dans toute la suite de ce travail nous parlerons de commande et de contrôleur, et non pas de supervision et du superviseur. Pour finir, nous discutons les principales méthodes de synthèse de contrôleur basées sur les réseaux de Petri.

2.1 Introduction

Étant donné un système manufacturier, nous nous intéressons maintenant à lui imposer un fonctionnement respectant un cahier des charges désiré. Pour les systèmes peu complexes, la manière la plus facile de contrôle est la conception directe du modèle supervisé du système. Malheureusement, dès que la complexité du système augmente, cette méthode ne peut pas garantir le respect du cahier des charges. Pour pallier cet inconvénient, des méthodes formelles permettant la synthèse de contrôleurs pour les systèmes à événements discrets ont été développées. Ces méthodes reposent sur l'utilisation d'outils tels que les automates et les réseaux de Petri. Nous allons présenter, dans ce chapitre, les concepts et résultats fondamentaux nécessaires dans la suite de cette étude. Pour plus de détails sur le contrôle des SED on peut se rapporter à [Wonham, 2011], ou à [Cassandras & Lafortune, 2008].

Ce sont les travaux de P. J. Ramadge et W. M. Wonham (R&W) qui constituent les bases de la théorie générale de la supervision des SED ([Ramadge & Wonham, 1987*b*],

[Ramadge & Wonham, 1987*a*]). L'objectif de l'approche R&W est de synthétiser un contrôleur pour un modèle donné, tel que le fonctionnement du procédé couplé au contrôleur reste toujours compris dans l'ensemble des comportements valides. De plus, ce fonctionnement doit être le plus permissif possible. Le comportement du système non-contrôlé, ainsi que l'ensemble des comportements désirés, sont chacun définis par un langage. À partir de ces langages, le but de la synthèse est de déterminer le sous-ensemble des comportements du procédé qui appartient aussi à la spécification. Ce sous-ensemble caractérise le langage du contrôleur et décrit les comportements du système contrôlé. Il est important, à ce niveau, de mentionner qu'il y a certains événements qui ne peuvent pas être interdits par le superviseur : les événements incontrôlables. L'importance du concept de contrôlabilité pour la théorie de la supervision des SED sera clarifié dans les sections suivantes.

Un procédé peut être couplé à un ou à plusieurs contrôleurs. Si nous utilisons un contrôleur unique, la commande sera dite centralisée. S'il y a plusieurs contrôleurs couplés au même procédé, la commande sera modulaire.

L'approche de R&W assure l'optimalité du contrôleur pour les systèmes modélisés par des automates à états finis. Cependant, bien que ce formalisme permet la modélisation d'une large classe des SED, il est peu adapté au systèmes réels à cause de sa grande sensibilité au problème de l'explosion combinatoire du nombre d'états (même dans les petits systèmes). En outre, l'utilisation de cette approche a montré une difficulté dans l'implémentation du système contrôlé. C'est pour ces raisons que de nombreuses tentatives ont été faites pour adapter la théorie générale de la commande des SED à d'autres formalismes de modélisation ([Uzam & Wonham, 2006], [Li & Wonham, 1994], [Holloway & Krogh, 1990], [Holloway et al., 1996], [Kumar & Holloway, 1996], [Ghaffari et al., 2003*b*], [Ghaffari et al., 2003*a*], [Yamalidou et al., 1996], [Dideban & Alla, 2006], [Dideban & Alla, 2007], [Dideban & Alla, 2008], [Liao et al., 2010], [Nazeem et al., 2011]). Nous allons discuter, au sein de ce chapitre, quelques-unes des plus importantes approches basées sur les réseaux de Petri.

2.2 Le concept de la commande par supervision

Pour pouvoir construire un formalisme de contrôle pour un système donné, il est essentiel d'avoir une très bonne connaissance de deux notions de base : *le procédé* physique qui doit être contrôlé et *le cahier des charges* (ou *la spécification*) qu'on souhaite lui imposer. C'est à partir de ces deux éléments et du schéma de supervision qui décrit leur relation que le calcul du contrôleur est réalisé. Dans ce chapitre, les résultats fondamentaux concernant l'existence du contrôleur et d'autres propriétés clefs pour la synthèse de ce dernier sont énoncés dans le cadre des langages formels. Cette présentation, indépendante du formalisme de modélisation utilisé, atteste la généralité absolue de ces

résultats.

2.2.1 Le schéma de supervision

Toutes les informations utiles concernant le procédé physique à contrôler sont intégrées dans son modèle. Le procédé est considéré comme un *générateur d'événements*, c'est-à-dire un SED évoluant de façon spontanée en générant des événements (figure 2.2a). L'approche R&W se place au niveau d'abstraction logique (atemporel). Le procédé est modélisé par un automate $\mathscr{P} = (\mathcal{Q}, \Sigma, \delta, q_0)$. Son comportement est alors donné par son langage, $\mathcal{L}(\mathscr{P})$. Ce langage contient toutes les séquences d'événements qui peuvent être générées par $\mathscr{P}$. Il est nécessairement préfixe-clos.

Le comportement du procédé en isolation n'est en général pas satisfaisant, dans le sens où il ne respecte pas certaines propriétés désirables pour l'exploitation. Ces propriétés constituent les *objectifs de contrôle*, ou la *spécification*. Le but du contrôle est de restreindre le comportement du procédé afin d'assurer le respect de ces objectifs. Cette restriction est réalisée par le biais d'un contrôleur, $\mathscr{S}$, qui est lui même un SED. Dans ce contexte, le contrôleur se révèle comme une fonction $\mathscr{S} : \mathcal{L}(\mathscr{P}) \to 2^{\Sigma}$ qui, dans chaque état du procédé, envoie à celui-ci la liste d'événements interdits par la spécification — *la politique de contrôle*, Φ (figures 2.1 et 2.2c).

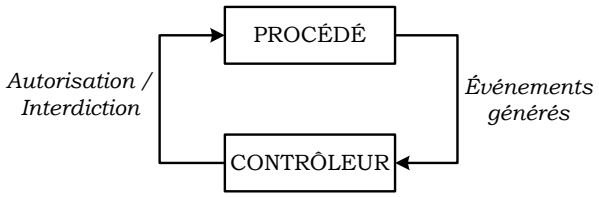


FIGURE 2.1 – Le schéma de supervision

Étant donné un état $q \in \mathcal{Q}$ du procédé, on note $\Sigma(q)$ l'ensemble des événements qui peuvent être générés par le procédé depuis cet état, et $\Phi(q)$ la liste des événements interdits par la spécification (*l'action de contrôle*) dans l'état q. Dans chaque état q, le procédé couplé au contrôleur reçoit en entrée la liste des événements interdits $\Phi(q)$ et émet en sortie l'ensemble des événements possibles et autorisés, $\Sigma(q)\backslash\Phi(q)$. Ce procédé, illustré dans la figure 2.2b, est appelé *procédé contrôlé*. Un événement σ est susceptible d'être généré par le procédé contrôlé $\mathscr{S}/\mathscr{P}$ depuis un état q si, et seulement si, il peut être généré par le procédé en isolation et s'il n'est pas interdit. C'est-à-dire : $\sigma \in \Sigma(q)$ et $\sigma \notin \Phi(q)$. Il va de soit que le procédé contrôlé peut être défini de façon analogue, en spécifiant en entrée la liste des événements autorisés, $\Sigma\backslash\Phi$.

Le fonctionnement du procédé couplé au contrôleur, appelé *fonctionnement en boucle fermée*, est décrit dans la figure 2.2c. Le contrôle ajoute au procédé des contraintes

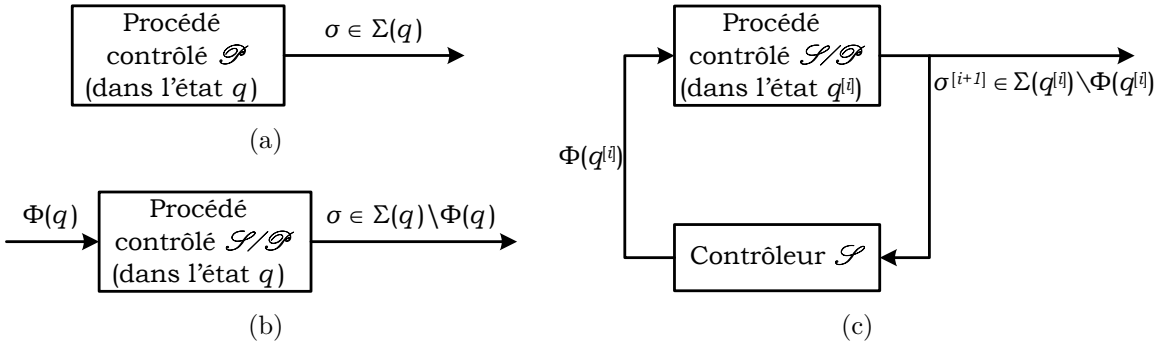


FIGURE 2.2 – Le concept de commande par supervision : a) Procédé en isolation ; b) Procédé supervisé ; c) Schéma de supervision avec indice de temps

supplémentaires, induisant un comportement souhaité pour le système en boucle fermée. En général, il y a plusieurs contrôleurs qui permettent de réaliser un fonctionnement en boucle fermée donné. Fondamentalement, l'observation du procédé par le contrôleur est asynchrone. A chaque instant logique $[i]$, le contrôleur construit, en se basant sur l'histoire du système, une liste d'événements interdits, $\Phi(q^{[i]})$, et la fournit au procédé. Après avoir reçu la liste, le procédé contrôlé peut générer à l'instant $[i+1]$ un événement $\sigma^{[i+1]} \in \Sigma(q^{[i]})\backslash\Phi(q^{[i]})$. Si l'occurrence de cet événement conduit le contrôleur vers un nouvel état, la liste des événements interdits est immédiatement actualisée, $\Phi(q^{[i+1]}) \neq \Phi(q^{[i]})$, et envoyée au procédé, qui réagit en conséquence. La notation $\mathscr{S}/\mathscr{P}$ sera dorénavant utilisée pour désigner le fonctionnement en boucle fermée du procédé $\mathscr{P}$ couplé au contrôleur $\mathscr{S}$.

Remarque 2.2.1. Le rôle du contrôleur consiste strictement à restreindre le fonctionnement du procédé.

❑

Pour bien garantir le fait que, après l'occurrence d'un événement quelconque, le contrôleur a le temps nécessaire pour interdire l'occurrence de tout événement indésirable, il est nécessaire de faire l'hypothèse suivante :

Hypothèse 2.2.1. *Deux événements ne peuvent pas être simultanément générés par le procédé.*

❑

Cette hypothèse est basée sur le fait que les événements ont une durée nulle. Les procédés considérés évoluent à des instants continus du temps, et la probabilité que deux événements soient simultanément générés par le procédé est aussi nulle.

Lors de toute discussion sur le contrôle des SED, il est important de réfléchir aussi au problème de l'optimalité. L'interdiction des comportements indésirables (illustrée dans le schéma de supervision) représente un aspect important du contrôle. Néanmoins, il existe un autre aspect, également important : la garantie du fait que le système n'est soumis à aucune restriction inutile. L'optimalité du contrôleur ne peut être assurée que lorsque cette deuxième exigence, *la condition de comportement maximal permissif*, est elle aussi garantie. Nous avons déjà mentionné le fait que l'approche R&W assume l'optimalité du contrôleur pour les modèles automates à états finis. Dans les chapitres suivants, nous allons traiter en détail les implications de cette condition essentielle pour notre travail dans le cas des modèles RdP généralisés synchronisés.

2.2.2 Contrôlabilité

Considérons maintenant le schéma de contrôle par supervision de la figure 2.2c. Il y a deux questions fondamentales qui doivent être prises en compte dans ce contexte : 1) le contrôleur est limité en termes de contrôlabilité des événements possibles dans le procédé, et 2) il est aussi limité en termes d'observabilité des événements exécutés par ce dernier. Dans ce qui suit, nous allons analyser seulement la question de la contrôlabilité, qui est essentielle pour les résultats présentés dans cette thèse. Des détails sur l'observabilité et ses implications peuvent être trouvés dans [Wonham, 2011] et [Cassandras & Lafortune, 2008].

L'existence des *événements incontrôlables* est une conjoncture usuelle dans la modélisation des systèmes automatisés réels. Ils peuvent représenter, soit des circonstances fondamentalement inévitables, telles que l'arrivée d'une panne ou la variation de la valeur repérée par un capteur, soit des actions qui ne peuvent pas être anticipées ou prévues à cause des limitations matérielles, soit des actions qu'on a choisi de désigner comme incontrôlables, telles que les événements ayant une très haute priorité ou le top d'un horloge. Quelque soit le cas de modélisation, le problème du point de vue du contrôle est le même : il faut ajouter des contrôles supplémentaires, car tout événement qui ne peut pas être interdit doit être toujours considéré comme faisant partie du comportement en boucle fermée. L'incontrôlabilité peut alors très rapidement compliquer la recherche d'un contrôleur garantissant un fonctionnement désiré pour le système supervisé. Si dans le cas des exemples très simples il est, malgré tout, possible de faire appel à l'intuition pour calculer un contrôleur satisfaisant même dans l'éventualité de l'incontrôlabilité, des méthodes formelles sont nécessaires si l'on souhaite résoudre ce problème pour des modèles plus complexes. Pour ces raisons, la contrôlabilité est un concept clef de la théorie R&W, constituant la base même de la synthèse de contrôleur.

Dans ce contexte, on peut définir une partition générale sur l'alphabet du procédé selon la contrôlabilité de ses événements :

Notation 2.2.1. Étant donné l'alphabet Σ d'événements d'un procédé $\mathscr{P}$, nous pouvons écrire :

$$\Sigma = \Sigma_C \cup \Sigma_U \ ,$$

où Σ_C et Σ_U désignent respectivement les ensembles d'événements contrôlables et incontrôlables sur Σ. Dorénavant, Σ_C sera appelé *l'alphabet des événements contrôlables*, et Σ_U *l'alphabet des événements incontrôlables.*

❑

Remarque 2.2.2. Seuls les événements contrôlables peuvent faire partie de la politique de contrôle. Le contrôleur est alors défini comme une fonction depuis le langage du procédé vers l'ensemble des sous-ensembles de Σ_C : $\mathscr{S} : \mathcal{L}(\mathscr{P}) \to 2^{\Sigma_C}$. Pour tout instant de temps logique $[i]$ et pour tout état $q \in \mathcal{Q}$ du procédé, nous avons : $\Phi(q^{[i]}) \cap \Sigma_U = \varnothing$.

❑

La théorie générale de la commande des SED dans l'approche R&W est basée sur les langages formels et les modèles automates à états finis. La contrôlabilité est une des propriétés de base pour cette approche, ayant des implications sur l'existence du contrôleur ; il faut alors l'étudier en se rapportant à ce cadre. Nous parlons alors de *la contrôlabilité d'un langage* :

Définition 2.2.1. *Un langage $\mathcal{K}$ est dit contrôlable par rapport à un langage $\mathcal{L}$ si :*

$$\overline{\mathcal{K}}\Sigma_U \cap \mathcal{L} \subseteq \overline{\mathcal{K}},$$

où $\overline{\mathcal{K}}$ est la préfixe-clôture de $\mathcal{K}$.

❑

Proposition 2.2.2 (Propriétés de la contrôlabilité). *Étant donnés deux langages $\mathcal{L}_1$ et $\mathcal{L}_2$, les affirmations suivantes peuvent être énoncés à l'égard de la contrôlabilité des opérations impliquant $\mathcal{L}_1$ et $\mathcal{L}_2$:*

1. *Si $\mathcal{L}_1$ et $\mathcal{L}_2$ sont contrôlables, alors leur union, $\mathcal{L}_\cup = \mathcal{L}_1 \cup \mathcal{L}_2$, est aussi contrôlable.*
2. *Si $\mathcal{L}_1$ et $\mathcal{L}_2$ sont contrôlables, leur intersection, $\mathcal{L}_\cap = \mathcal{L}_1 \cap \mathcal{L}_2$, n'est pas nécessairement contrôlable.*
3. *Si $\mathcal{L}_1$ et $\mathcal{L}_2$ sont contrôlables et si $\overline{\mathcal{L}_1} \cap \overline{\mathcal{L}_2} = \overline{\mathcal{L}_1 \cap \mathcal{L}_2}$ (i.e. $\mathcal{L}_1$ et $\mathcal{L}_2$ sont des langages non-conflictuels), alors l'intersection $\mathcal{L}_\cap = \mathcal{L}_1 \cap \mathcal{L}_2$ est aussi contrôlable.*
4. *Si $\mathcal{L}_1$ et $\mathcal{L}_2$ sont contrôlables et préfixe-clos, alors leur intersection, $\mathcal{L}_\cap = \mathcal{L}_1 \cap \mathcal{L}_2$, est aussi contrôlable et préfixe-close.*

❑

Remarques 2.2.3. a) Par définition, la contrôlabilité est une propriété de la préfixe-clôture des langages. Un langage $\mathcal{K}$ est contrôlable si, et seulement si, sa préfixe-clôture, $\overline{\mathcal{K}}$, est contrôlable.

b) [*La condition de contrôlabilité :*] Si dans la définition 2.2.1 on considère $\mathcal{K}$ comme le langage de la spécification et $\mathcal{L}$ comme le langage du procédé, avec $\mathcal{K} \subseteq \mathcal{L}, \mathcal{K} \neq \varnothing$, on dit que le langage de la spécification est contrôlable par rapport au langage du procédé si : $\forall$ mot $\omega \in \mathcal{K}$, $\forall \sigma \in \Sigma_U : \omega\sigma \in \mathcal{L} \Rightarrow \omega\sigma \in \mathcal{K}$.

❑

2.2.3 Définitions

Le contrôleur $\mathscr{S}$ peut être perçu comme une machine à états déterministe qui évolue conformément à la modification de son entrée (sur l'occurrence d'événements $\sigma \in \Sigma$ générés par le procédé) et qui change d'état selon une fonction de transition d'état, ξ. La sortie d'un contrôleur ne dépend que de l'entrée de celui-ci. On peut donc modéliser le contrôleur par une machine de Moore :

Définition 2.2.2. *Le contrôleur $\mathscr{S}$ est un 6-uplet $\mathscr{S} = (\mathcal{V}, \Sigma, \xi, v_0, 2^{\Sigma_C}, \theta)$, où :*

- $\mathcal{V}$ *est l'ensemble fini d'états,*
- Σ *est l'alphabet d'entrée,*
- $\xi : \mathcal{V} \times \Sigma \to \mathcal{V}$ *est la fonction de transition d'état,*
- v_0 *est l'état initial,*
- 2^{Σ_C} *est l'alphabet de sortie, et*
- $\theta : \mathcal{V} \to 2^{\Sigma_C}$ *est la fonction d'affectation de sortie.*

❑

Pour chaque état $v \in \mathcal{V}$, le contrôleur $\mathscr{S}$ fournit en sortie une liste d'événements interdits $\Phi = \theta(v)$. Comme seuls les événements contrôlables peuvent être interdits par le contrôle, chaque sortie de $\mathscr{S}$ est un élément de 2^{Σ_C}, où 2^{Σ_C} est l'ensemble de tous les sous-ensembles de Σ_C. Le modèle accepteur équivalent du contrôleur peut être obtenu en partant de la définition 2.2.2 et en supprimant depuis tout état $v \in \mathcal{V}$ toute transition de sortie associée à un événement interdit par le contrôleur dans l'état v.

L'ensemble des séquences d'événements autorisés par le contrôleur constitue le langage de supervision, $\mathcal{L}(\mathscr{S})$.

Le modèle accepteur du système contrôlé $\mathscr{S}/\mathscr{P}$ est construit en effectuant le composé synchrone des modèles accepteurs du procédé et du contrôleur : $\mathscr{S}/\mathscr{P} = \mathscr{S} \parallel_s \mathscr{P}$. Le langage $\mathcal{L}(\mathscr{S}/\mathscr{P})$, reconnu par la machine $\mathscr{S}/\mathscr{P}$, représente le fonctionnement en boucle fermée du système. Ce langage contient toute séquence d'événements $\omega \in \Sigma^*$ possible dans le procédé en isolation et autorisée par le contrôleur : $\mathcal{L}(\mathscr{S}/\mathscr{P}) = \mathcal{L}(\mathscr{P}) \cap \mathcal{L}(\mathscr{S})$. Un mot $\omega\sigma$ peut être généré par le procédé contrôlé si le mot ω a été généré par le procédé contrôlé et si l'événement σ est autorisé par le superviseur et le mot $\omega\sigma$ est physiquement possible. Le mot vide ε est compris dans le langage $\mathcal{L}(\mathscr{S}/\mathscr{P})$. Formellement :

Définition 2.2.3. *Le langage $\mathcal{L}(\mathscr{S}/\mathscr{P})$ généré par le procédé contrôlé en boucle fermée est défini par :*

- $\varepsilon \in \mathcal{L}(\mathscr{S}/\mathscr{P})$,
- $[\omega\sigma \in \mathcal{L}(\mathscr{S}/\mathscr{P})] \Leftrightarrow [(\omega \in \mathcal{L}(\mathscr{S}/\mathscr{P})) \wedge (\sigma \in \mathscr{S}(\omega)) \wedge (\omega\sigma \in \mathcal{L}(\mathscr{P}))]$,

où $\sigma \in \mathscr{S}(\omega)$ dénote le fait que l'occurrence de l'événement σ après le mot ω n'est pas interdit par le contrôleur.

❑

2.3 Synthèse du contrôle

La notion de contrôlabilité d'un SED a été introduite par Ramadge et Wonham ([Ramadge & Wonham, 1987*b*]) afin de caractériser les langages contrôlés d'un procédé dans le cadre de la théorie générale de la supervision. L'existence d'un contrôleur qui assure un comportement souhaité pour le système en boucle fermée est intimement liée au concept de contrôlabilité des langages, *i.e.* la possibilité d'imposer que le langage généré par le procédé couplé au contrôleur appartienne à un certain langage de spécification.

2.3.1 L'idée

Dans le cas général, pour déterminer un contrôleur permettant le respect d'un cahier des charges donné, il faut d'abord définir un modèle de spécification qui traduit fidèlement les contraintes imposées. Ce modèle peut être conçu de façon semblable à celui du contrôleur, en utilisant les machines de Moore ou les accepteurs. Étant donné le modèle $\mathscr{S}_{spec}$ d'une spécification, son langage associé, $\mathcal{L}(\mathscr{S}_{spec})$, contiendra l'ensemble des séquences tolérées par la spécification.

Remarque 2.3.1. Par définition, un contrôleur est une spécification. Cependant la réciproque n'est pas vraie.

❑

Le fonctionnement le plus permissif possible en boucle fermée, en respectant les conditions imposées par la spécification, est décrit par le langage $\mathcal{L}_{\mathscr{D}} = \mathcal{L}(\mathscr{P}) \cap \mathcal{L}(\mathscr{S}_{spec})$ — c'est le fonctionnement désiré. Par définition, $\mathcal{L}_{\mathscr{D}}$ contient les mêmes contraintes de fonctionnement que le langage de spécification, $\mathcal{L}(\mathscr{S}_{spec})$.

Remarque 2.3.2. En général, il n'est pas possible de restreindre, par la commande, le fonctionnement d'un procédé à n'importe quel sous-ensemble de celui-ci. L'existence d'un contrôleur $\mathscr{S}$ tel que $\mathcal{L}(\mathscr{S}/\mathscr{P}) = \mathcal{L}_{\mathscr{D}}$ réside dans le concept de contrôlabilité.

❑

Propriété 2.3.1. *Soit $\mathcal{L}(\mathscr{P})$ le langage d'un procédé. Le langage de spécification, $\mathcal{L}(\mathscr{S}_{spec})$, est contrôlable si, et seulement si, le fonctionnement désiré, $\mathcal{L}_{\mathscr{D}} = \mathcal{L}(\mathscr{P}) \cap \mathcal{L}(\mathscr{S}_{spec})$, est contrôlable.*

❑

Étant donné le langage $\mathcal{L}(\mathscr{P})$ d'un procédé, la condition nécessaire et suffisante pour l'existence d'un contrôleur $\mathscr{S}$ assurant un fonctionnement désiré $\mathcal{L}_{\mathscr{D}}$ inclus dans $\mathcal{L}(\mathscr{P})$ est donnée ci-dessous ([Ramadge & Wonham, 1989]) :

Théorème 2.3.2. *Pour un langage $\mathcal{L}_{\mathscr{D}}$ préfixe-clos, non vide et inclus dans le langage $\mathcal{L}(\mathscr{P})$ d'un procédé $\mathscr{P}$, il existe un contrôleur $\mathscr{S}$ tel que $\mathcal{L}(\mathscr{S}/\mathscr{P}) = \mathcal{L}_{\mathscr{D}}$ si $\mathcal{L}_{\mathscr{D}}$ est contrôlable par rapport à $\mathcal{L}(\mathscr{P})$.*

❑

Si le langage $\mathcal{L}_{\mathscr{D}}$ n'est pas contrôlable, il est toujours possible de trouver une solution de contrôle plus restrictive. Soit $\mathscr{C}(\mathcal{L}_{\mathscr{D}})$ la classe de tous sous-langages contrôlables et préfixe-clos de $\mathcal{L}_{\mathscr{D}}$:

$$\mathscr{C}(\mathcal{L}_{\mathscr{D}}) = \left\{ \mathcal{K} \in \mathcal{L}_{\mathscr{D}} \mid \mathcal{K} = \overline{\mathcal{K}} \text{ et } \mathcal{K}\Sigma_U \cap \mathcal{L}(\mathscr{P}) \subseteq \mathcal{K} \right\},$$

où $\mathcal{L}(\mathscr{P})$ est le langage du procédé. Cette classe est fermée sur l'union des langages. Il existe alors un plus grand élément de $\mathscr{C}(\mathcal{L}_{\mathscr{D}})$, *le langage suprême contrôlable* $\mathcal{K}^{\uparrow\mathscr{C}}$ ([Brandin & Wonham, 1994]) :

$$\mathcal{K}^{\uparrow\mathscr{C}} = \bigcup_{\mathcal{K} \in \mathscr{C}(\mathcal{L}_{\mathscr{D}})} \mathcal{K} .$$

Évidemment (théorème 2.3.2), le langage $\mathcal{L}(\mathscr{S}/\mathscr{P}) = \mathcal{K}^{\uparrow\mathscr{C}} \subseteq \mathcal{L}_{\mathscr{D}}$ décrit le contrôleur le plus permissif possible qui assure le respect de la spécification pour le système en boucle fermée (figure 2.3).

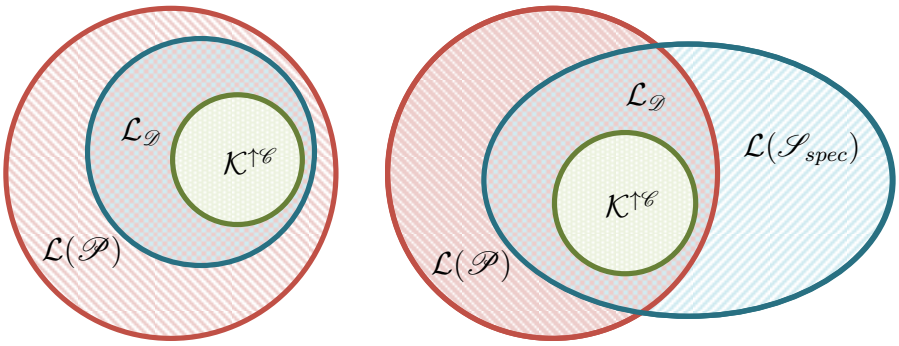
Remarque 2.3.3. Si le fonctionnement désiré $\mathcal{L}_{\mathscr{D}}$ est préfixe-clos, alors son langage suprême contrôlable $\mathcal{K}^{\uparrow\mathscr{C}}$ est aussi préfixe-clos.

❑

Propriété 2.3.3 (Propriétés de l'opération $\uparrow\mathscr{C}$). *La contrôlabilité et la "supremalité" du langage suprême contrôlable lui confèrent les propriétés suivantes :*

1. $(\mathcal{K}_1 \cap \mathcal{K}_2)^{\uparrow\mathscr{C}} \subseteq \mathcal{K}_1^{\uparrow\mathscr{C}} \cap \mathcal{K}_2^{\uparrow\mathscr{C}}$.
2. $(\mathcal{K}_1 \cap \mathcal{K}_2)^{\uparrow\mathscr{C}} = \left(\mathcal{K}_1^{\uparrow\mathscr{C}} \cap \mathcal{K}_2^{\uparrow\mathscr{C}}\right)^{\uparrow\mathscr{C}}$.
3. *Si $\mathcal{K}_1^{\uparrow\mathscr{C}}$ et $\mathcal{K}_2^{\uparrow\mathscr{C}}$ sont des langages non-conflictuels, alors :* $(\mathcal{K}_1 \cap \mathcal{K}_2)^{\uparrow\mathscr{C}} = \mathcal{K}_1^{\uparrow\mathscr{C}} \cap \mathcal{K}_2^{\uparrow\mathscr{C}}$.
4. $(\mathcal{K}_1 \cup \mathcal{K}_2)^{\uparrow\mathscr{C}} \supseteq \mathcal{K}_1^{\uparrow\mathscr{C}} \cup \mathcal{K}_2^{\uparrow\mathscr{C}}$.

❑



(a) (b)

FIGURE 2.3 – Relations entre les langages impliqués dans la synthèse du contrôleur : a) Langage suprême contrôlable d'un fonctionnement désiré ; b) Langage de spécification, fonctionnement désiré, et langage suprême contrôlable du fonctionnement désiré.

Remarque 2.3.4. Tout suffixe composé exclusivement des événements incontrôlables peut amener le système contrôlé hors du fonctionnement désiré $\mathcal{L}_{\mathscr{D}}$. Afin d'assurer le respect de la condition de contrôlabilité, il faut garantir le fait que la «frontière» entre $\mathcal{K}^{\uparrow\mathscr{C}}$ et son complément dans $\mathcal{L}_{\mathscr{D}}$ n'inclut que des chaînes qui peuvent être étendues hors de $\mathcal{K}^{\uparrow\mathscr{C}}$ uniquement *via* des événements contrôlables.

❑

2.3.2 L'algorithme de Kumar

Le théorème 2.3.2 rend explicite l'importance de la contrôlabilité pour la synthèse de la commande. Pour vérifier la contrôlabilité du langage de commande nous utilisons l'algorithme de Kumar ([Kumar, 1991]). Cette méthode permet l'évaluation de la contrôlabilité du langage de contrôle $\mathcal{L}(\mathscr{S}_{spec})$ à partir des modèles accepteurs d'un procédé, $\mathscr{P}$, et d'une spécification de fonctionnement, $\mathscr{S}_{spec}$. L'algorithme présente aussi l'avantage supplémentaire de permettre la synthèse du modèle accepteur du langage suprême contrôlable du fonctionnement désiré, $\mathcal{K}^{\uparrow\mathscr{C}}$. Il permet ainsi la résolution du cas où le langage $\mathcal{L}(\mathscr{S}_{spec})$ n'est pas contrôlable. L'approche est basée sur l'identification des différents types d'états interdits du modèle accepteur du fonctionnement désiré, types d'états que nous allons définir prochainement.

Soit $\mathscr{P} = (\mathcal{Q}, \Sigma, \delta, q_0)$ le modèle accepteur du procédé, et $\mathscr{S}_{spec} = (\mathcal{V}, \Sigma, \xi, v_0)$ celui de la spécification que nous souhaitons lui imposer. Soit $\mathscr{S}/\mathscr{P} = (\mathcal{X}, \Sigma, \psi, x_0)$ le modèle automate du système contrôlé ($\mathscr{S}/\mathscr{P} = \mathscr{P} \parallel_s \mathscr{S}_{spec}$), et soit $\mathcal{L}_{\mathscr{D}} = \mathcal{L}(\mathscr{S}/\mathscr{P})$ le fonctionnement désiré reconnu par ce modèle. Les notions suivantes peuvent être définies dans ce contexte (figure 2.4) :

Définition 2.3.1. *Tout état $x = (q, v)$ de $\mathscr{S}/\mathscr{P}$ tel qu'il existe un événement incontrôlable $\sigma \in \Sigma_U$ où $\delta(q, \sigma)$ est définie dans $\mathscr{P}$ et $\xi(v, \sigma)$ n'est pas définie dans $\mathscr{S}_{spec}$ est appelé état interdit. L'ensemble des états interdits est alors donné par la relation suivante :*

$$\mathcal{M}_{\mathcal{I}} = \{x = (q, v) \mid \sigma \in \Sigma_U \text{ avec } \delta(q, \sigma) \text{ défini et } \xi(v, \sigma) \text{ non défini}\}.$$

❑

Remarque 2.3.5. La définition 2.3.1 donne un type d'état interdit. L'autre type d'état interdit est donné par les états de blocage.

❑

Pour des raisons déjà présentées dans la remarque 2.3.4 et pour bien caractériser le comportement du système, il faut encore identifier les états permettant l'accessibilité des états interdits *via* l'occurrence des séquences d'événements incontrôlables. Cet ensemble est appelé *l'ensemble d'états faiblement interdits*, $\mathcal{M}_{\mathcal{F}}$:

Définition 2.3.2. *On appelle état faiblement interdit tout état $x = (q, v)$ de $\mathscr{S}/\mathscr{P}$ tel qu'il existe une séquence d'événements incontrôlables $\mathcal{S}_u \in \Sigma_U^*$ qui conduit le système depuis $x = (q, v)$ vers un état interdit $x' = (q', v')$ de $\mathscr{S}/\mathscr{P}$. Nous pouvons alors écrire :*

$$\mathcal{M}_{\mathcal{F}} = \left\{x_i \mid x_i \in \mathcal{M}_{\mathcal{A}_{spec}}, \exists\, x_j \in \mathcal{M}_{\mathcal{I}} \text{ ou } x_j \in \mathcal{M}_{\mathcal{F}} \text{ et } \exists\, \mathcal{S}_u \in \Sigma_U * \text{ tel que } x_i \xrightarrow{s_u} x_j\right\},$$

où $\mathcal{M}_{\mathcal{A}_{spec}}$ dénote l'ensemble des états possibles et autorisés par la spécification.

❑

Soit $\mathcal{M}_{\mathcal{A}}$ *l'ensemble des états autorisés par le système contrôlé*. Deux autres ensembles d'états peuvent être construits à partir de $\mathcal{M}_{\mathcal{A}}$, $\mathcal{M}_{\mathcal{I}}$ et $\mathcal{M}_{\mathcal{F}}$: *l'ensemble des états interdits frontières* et *l'ensemble des états autorisés critiques.*

Définition 2.3.3. *L'ensemble des états interdits frontières, $\mathcal{M}_{\mathcal{B}}$, correspond aux états interdits atteignables par occurrence d'événements contrôlables :*

$$\mathcal{M}_{\mathcal{B}} = \left\{x_j \mid x_j \in \mathcal{M}_{\mathcal{I}} \text{ ou } x_j \in \mathcal{M}_{\mathcal{F}}, \exists\, x_i \in \mathcal{M}_{\mathcal{A}} \text{ et } \exists\, \sigma \in \Sigma_C \text{ tel que } x_i \xrightarrow{\sigma} x_j\right\}.$$

La notation $\mathcal{M}_{\mathcal{B}}$ vient du fait que cet ensemble d'états constitue la «borne» contrôlable de $\mathcal{M}_{\mathcal{A}_{spec}}$.

❑

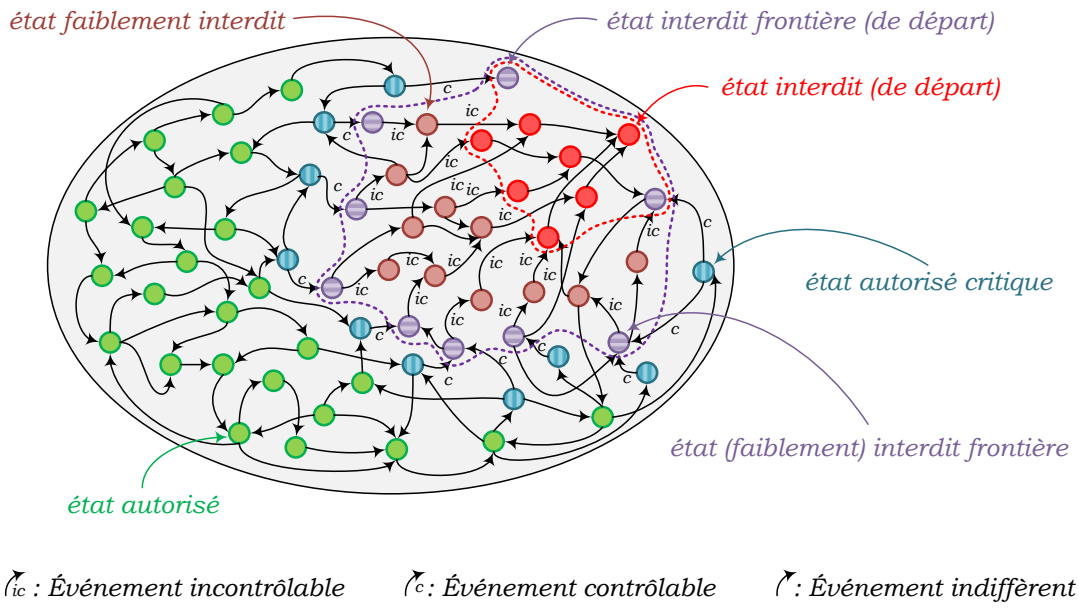


FIGURE 2.4 – Les ensembles d'états possibles dans le modèle accepteur d'un fonctionnement désiré

Définition 2.3.4. *L'ensemble des états autorisés critiques, $\mathcal{M}_C$, correspond aux états autorisés à partir desquels l'occurrence d'événements contrôlables mène vers un état interdit frontière :*

$$\mathcal{M}_C = \left\{ x_i \,|\, x_i \in \mathcal{M}_A, \exists\, x_j \in \mathcal{M}_B \text{ et } \exists\, \sigma \in \Sigma_C \text{ tel que } x_i \xrightarrow{\sigma} x_j \right\}.$$

❑

Tous ces états sont représentés dans la figure 2.4.

Remarque 2.3.6. Nous pouvons remarquer que, pour empêcher l'atteignabilité de tout état interdit, il faut et il suffit d'interdire le franchissement des événements contrôlables liant les états autorisés critiques aux états interdits frontières.

❑

Nous pouvons maintenant présenter l'algorithme de calcul du langage suprême contrôlable $\mathcal{K}^{\uparrow \mathscr{C}}$:

Algorithme 2.1 (Algorithme de Kumar - présentation formelle). Étant donnés les modèles accepteur $\mathscr{P}$ et $\mathscr{S}_{spec}$ d'un procédé et du cahier des charges à lui imposer, l'algorithme ci-dessous calcule l'automate $\mathscr{D}'$ qui accepte le langage suprême contrôlable du fonctionnement désiré pour le système en boucle fermée $\mathscr{S}/\mathscr{P} = \mathscr{P} \parallel_s \mathscr{S}_{spec}$:

Pas 1. Construction du composé synchrone $\mathscr{S}/\mathscr{P} = \mathscr{P} \parallel_s \mathscr{S}_{spec}$.

Pas 2. Détermination de l'ensemble d'états interdits, $\mathcal{M}_{\mathcal{I}}$.

Pas 3. Détermination de l'ensemble d'états faiblement interdits, $\mathcal{M}_{\mathcal{F}}$.

Pas 4. • Élimination des transitions liant les ensembles des états interdits et faiblement interdits dans $\mathscr{S}/\mathscr{P}$ ($\mathcal{M}_{\mathcal{I}}$ et $\mathcal{M}_{\mathcal{F}}$ sont supprimés du modèle).

• Élimination des états de $\mathscr{S}/\mathscr{P}$ devenus non-accessibles. Un état $x = (q, v)$ est dit *non-accessible* s'il n'existe pas de chemin permettant de rejoindre $x = (q, v)$ depuis l'état initial.

❑

Proposition 2.3.4. *L'automate $\mathscr{D}'$ obtenu comme résultat de l'algorithme de Kumar reconnaît le langage suprême contrôlable $\mathcal{K}^{\uparrow\mathscr{C}}$.*

❑

A partir du résultat obtenu par l'algorithme de Kumar, nous obtenons le contrôleur optimal.

2.3.3 Exemple d'un système manufacturier

Considérons l'exemple classique du système donné dans la figure 2.5a. Nous avons deux machines à trois états, $\mathscr{M}_1$ et $\mathscr{M}_2$, qui travaillent en tandem : chaque pièce doit être premièrement usinée par $\mathscr{M}_1$ et puis par $\mathscr{M}_2$, avec un stock de capacité unitaire entre les deux. Le schéma du système manufacturier, ainsi que les modèles automates de procédé et de la contrainte de stock (la spécification), sont présentés dans la figure 2.5).

Les fonctionnements des deux machines sont indépendants l'un de l'autre. Chaque machine est modélisée par un automate à trois états : arrêt (q_0 et q_3), marche (q_1 et q_4), et panne (q_2 et q_5). Dans l'état initial, les machines sont à l'arrêt. L'évolution de chaque machine est identique à celle décrite dans l'exemple 1.1.1.

Le modèle du procédé est obtenu en construisant le composé synchrone de $\mathscr{M}_1$ et $\mathscr{M}_2$: $\mathscr{P} = \mathscr{M}_1 \parallel_s \mathscr{M}_2$ (figure 2.5d). Le fonctionnement de $\mathscr{P}$ est défini sur l'alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$, où $\Sigma_i = \{d_i, f_i, p_i, r_i\}$ est l'alphabet de la machine $\mathscr{M}_i$. Nous avons, également : $\Sigma = \Sigma_C \cup \Sigma_U$, où Σ_C et Σ_U sont les alphabets des événements contrôlables et, respectivement, incontrôlables du système : $\Sigma_C = \{d_1, d_2, r_1, r_2\}$ et $\Sigma_U = \{f_1, f_2, p_1, p_2\}$.

La contrainte de stock imposée par la spécification a les implications suivantes :

1. la machine $\mathscr{M}_2$ ne peut commencer l'usinage que si elle peut prendre une pièce du stock, *i.e.* si le stock est plein,
2. la machine $\mathscr{M}_1$ ne peut déposer une pièce dans le stock que s'il y a de la place, *i.e.* si le stock est vide.

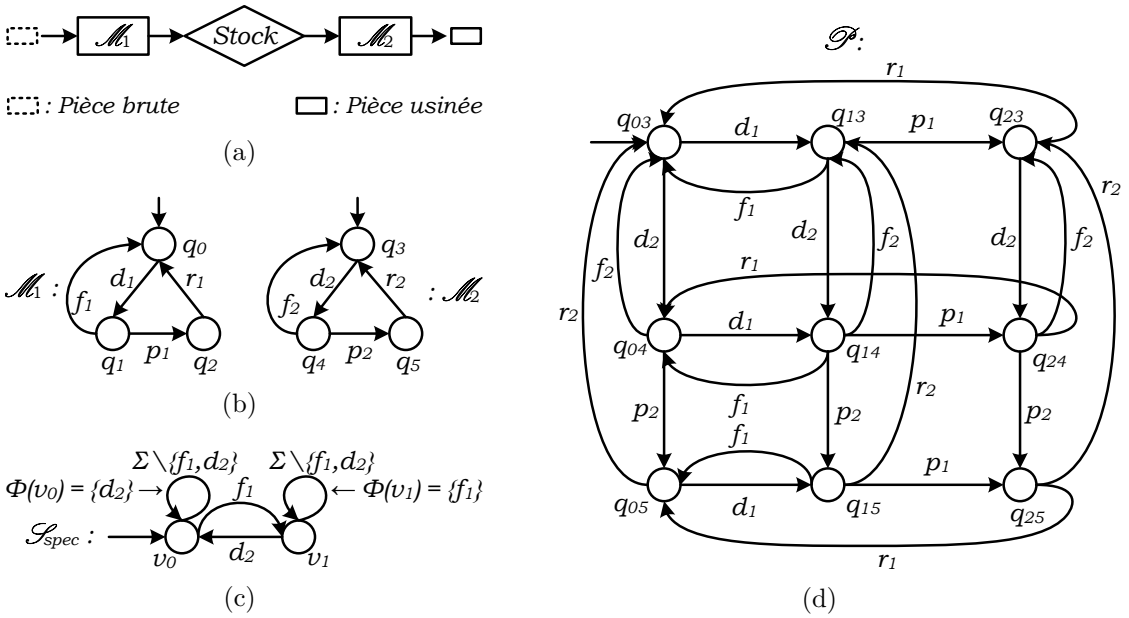


FIGURE 2.5 – Modèles automates du procédé et de la spécification : a) schéma du système manufacturier ; b) machines à trois états ; c) spécification ; d) procédé en isolation.

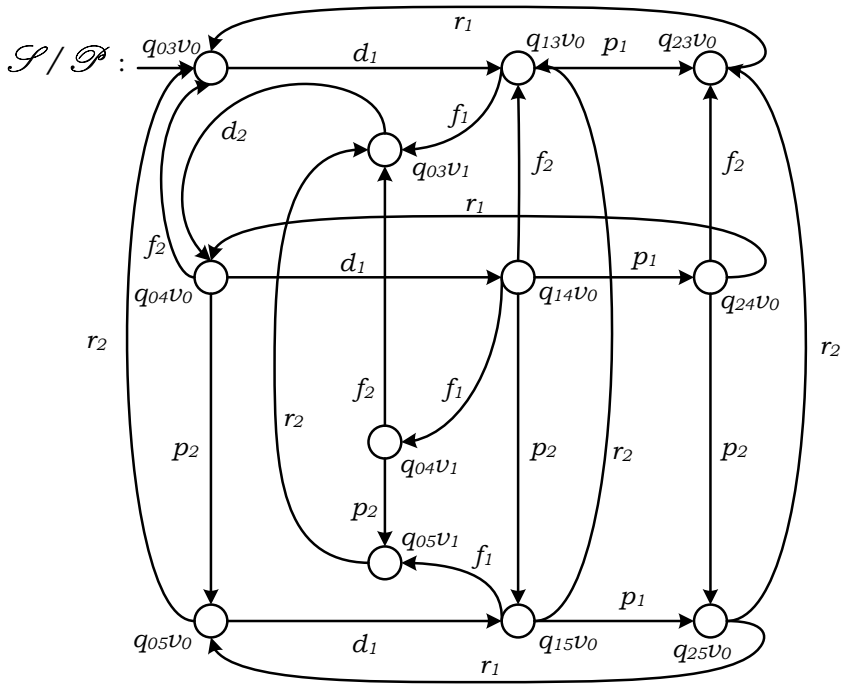


FIGURE 2.6 – Modèle du contrôleur pour le système manufacturier

Autrement dit, lorsque le stock est vide (état v_0 de la figure 2.5c), l'occurrence de l'événement contrôlable d_2 doit être interdite. Le stock évolue vers l'état v_1 (stock plein) au moment de la fin de cycle de $\mathscr{M}_1$ (événement incontrôlable f_1). Une fois le stock plein, l'occurrence de l'événement f_1 est interdite. C'est le faite de vouloir interdire cet événement incontrôlable qui va poser un problème. Le stock est supposé vide dans son état initial.

Le système supervisé (boucle fermée) est le composé synchrone des modèles du procédé et de la spécification : $\mathscr{S}/\mathscr{P} = \mathscr{P} \parallel_s \mathscr{S}_{spec}$. L'accepteur $\mathscr{S}/\mathscr{P}$ reconnaît le fonctionnement désiré : $\mathcal{L}(\mathscr{S}/\mathscr{P}) = \mathcal{L}(\mathscr{P}) \cap \mathcal{L}(\mathscr{S}_{spec})$.

Le contrôleur optimal est déterminé avec l'algorithme de Kumar. La spécification interdit l'occurrence de f_1 dans l'état v_1 de $\mathscr{S}_{spec}$. Mais, comme cet événement est incontrôlable, il s'ensuit qu'il faut en fait interdire le début du cycle de la machine $\mathscr{M}_1$ (l'événement contrôlable d_1) quand le stock est plein.

L'ensemble des états interdits est le suivant : $\mathcal{M}_{\mathcal{I}} = \{(q_{13}, v_1), (q_{14}, v_1), (q_{15}, v_1)\}$. Pour cet exemple, il n'y a pas d'états faiblement interdits. La figure 2.6 donne le modèle du contrôleur qui correspond au comportement maximal permissif.

2.3.4 Avantages et inconvénients de la théorie de Ramadge & Wonham

La théorie de R&W pour la commande des SED constitue actuellement un centre d'intérêt considérable pour la communauté scientifique concernée par ce problème. Sur le plan international, il y a plusieurs groupes de recherche s'intéressant à l'extension et l'optimisation de cette théorie. L'approche classique utilise les automates à états finis. Elle se confronte, cependant, au problème de l'explosion combinatoire d'états. 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 contrôleur qui comporte (au plus) $n \cdot m$ états. La taille du contrôleur est alors beaucoup plus grande que celle du procédé. Il s'ensuit que dans la plupart des cas l'explosion combinatoire due à l'utilisation de modèles automates rend impossible la synthèse de contrôleurs pour les systèmes de taille réelle.

Maîtriser la taille des contrôleurs est un objectif crucial pour l'applicabilité de l'approche R&W . Un grand nombre d'extensions sur la théorie de R&W visent à résoudre ce problème. Différents travaux proposent des méthodes pour réduire la taille des modèles obtenus, mais à ce point il y a encore beaucoup des difficultés de modélisation et d'implantation.

D'autres extensions visent l'incorporation des contraintes temporelles ([Gaubert,

1995], [Brandin & Wonham, 1994], [Sava, 2002]), ou des possibilités de modélisation plus générales, tels que les systèmes hybrides ([Sreenivas & Krogh, 1992], [Uzam & Wonham, 2006]), les SED à structure vectorielle ([Li & Wonham, 1994]), et les réseaux de Petri que nous allons étudier ci-dessous.

2.4 Commande par réseau de Petri

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 la manque de structure dans les modèles, limitent la possibilité de développer des algorithmes de calcul efficaces pour l'analyse et la synthèse des systèmes réels. Pour pallier ces difficultés, des approches basées sur les réseaux de Petri (RdP) ont été proposées.

Le problème qu'il faut se poser en premier est s'il est possible d'utiliser des modèles RdP à la place des automates dans la théorie générale du contrôle supervisé des SED. Nous avons montré que les notions fondamentales de cette théorie sont indépendantes du formalisme de modélisation, puisqu'elles sont des propriétés des langages. Cependant, l'algorithme de synthèse du contrôleur est basé sur la représentation automate des langages réguliers du procédé et de la spécification. Ça veut dire que, si le langage du procédé est régulier alors que celui de la spécification ne l'est pas, le contrôleur ne peut pas être réalisé par un automate à états finis. Toutefois, il est possible qu'il soit réalisable par un RdP. Il s'ensuit qu'une motivation importante pour l'utilisation des RdP est d'élargir l'air d'applicabilité des résultats fondamentaux de la théorie de R&W au-delà de la classe des systèmes décrits par des langages réguliers ([Giua & DiCesare, 1994*a*], [Sreenivas & Krogh, 1992]). Malheureusement, cette piste de recherche a rencontré de fortes difficultés liées aux particularités et aux propriétés spécifiques des langages RdP et aucun résultat général n'a encore été développé à cet égard.

Malgré cela, il est toutefois préférable d'utiliser les RdP à la place des automates quand les langages du procédé et de la spécification sont tous les deux réguliers, car ces modèles sont beaucoup plus compacts et offrent de nets avantages du point de vue complexité de calcul. De plus, la richesse d'information de nature structurelle captée par les RdP a donné lieu au développement d'approches de contrôle basées sur l'état. Ces méthodes changent la perspective du paradigme de contrôle du point de vue du langage au point de vue de l'état. Les spécifications à imposer au procédé sont données en termes d'états interdits et l'objectif du contrôle est de décider lesquelles des transitions contrôlables et validées peuvent être autorisées pour le franchissement à partir de chaque état du système. Le contrôleur doit garantir le fait qu'aucun état interdit n'est accessible dans le système en boucle fermée, tout en assurant au même temps la minimalité du degré d'intervention dans le fonctionnement du procédé. La synthèse de contrôleurs de

ce type exploite les propriétés structurelles intégrées dans les modèles RdP. Cette section réalise une présentation schématique de quelques-unes des méthodes de contrôle les plus connues pour les RdP.

2.4.1 La méthode des invariants

La méthode des invariants résout le problème des états interdits par l'ajout judicieux *de places et d'arcs «de contrôle»* au modèle du procédé, le résultat étant un modèle RdP contrôlé qui a le fonctionnement désiré en boucle fermée. L'approche utilise le concept des invariants de marquage pour calculer de manière très élégante les matrices d'incidence du système contrôlé et du contrôleur ([Yamalidou et al., 1996], [Moody & Antsaklis, 1998], [Moody & Antsaklis, 2000], [Iordache & Antsaklis, 2006]).

La question essentielle qu'il faut se poser est comment relier les places de contrôle aux transitions du procédé afin de garantir le fait que le RdP contrôlé satisfasse les spécifications données pour tous ses états accessibles. Le concept sous-jacent au cœur de la solution à ce problème est celui d'invariant de marquage. L'idée est d'utiliser la place de contrôle pour incorporer la spécification dans l'ensemble des composantes conservatives du système contrôlé. Supposons que les objectifs de contrôle peuvent être exprimés en termes d'un ensemble d'inégalités linéaires sur l'état M du réseau. Pour que la spécification soit satisfaite, cet ensemble d'inégalités doit être vrai pour tout état accessible du système en boucle fermée. Il faut alors utiliser les marquages des places de contrôle pour «transformer» les contraintes décrivant le fonctionnement désiré dans des invariants de marquage du système contrôlé. Pour chaque contrainte, une place de contrôle est ajoutée au modèle afin de compléter l'invariant et une colonne supplémentaire est apposée à la matrice d'incidence associé du système contrôlé. Cette colonne décrit les liens entre la place de contrôle qui vient d'être ajoutée et les transitions du procédé. La matrice d'incidence du système contrôlé correspond alors à la concaténation verticale des matrices d'incidence du procédé et du contrôleur.

Contraintes généralisées d'exclusion mutuelle :

La méthode des invariants est basée sur l'expression de la spécification sous la forme d'un système des *contraintes généralisées d'exclusion mutuelle* (GMEC) ([Giua et al., 1992], [Giua & DiCesare, 1994*b*]). Ce type de contrainte se présente comme une condition qui limite la somme pondérée des jetons dans un ensemble des places. Formellement, un GMEC est une inégalité linéaire de type :

$$c_{\mathscr{S}} : l^T \cdot M \leq b, \ \ \forall\, M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \tag{2.4.1}$$

où $l \in \mathbb{Z}^n$ est le vecteur (colonne) des poids de la contrainte, M est le vecteur de marquage générique, $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$ est l'ensemble d'états accessibles du réseau, et $b \in \mathbb{Z}$ est la borne de la contrainte. Une telle inégalité définit un ensemble des états autorisés : $\mathcal{M}(l, b) = \left\{ M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \mid l^T \cdot M \leq b \right\}$.

Normalement, une spécification comprend un ensemble des contraintes :

$$\mathcal{M}(L, b_v) = \bigcap_{i=1}^{r} \mathcal{M}(l_i, b_i) = \left\{ M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \mid L^T \cdot M \leq b_v \right\},$$

où $L = [l_1 \quad l_2 \quad ... \quad l_r]$, $b_v = [b_1 \quad b_2 \quad ... \quad b_r]^T$ et $\mathcal{M}(L, b_v)$ représente l'ensemble des marquages autorisés par l'ensemble des contraintes.

Remarque 2.4.1. Étant donné un RdP et son marquage initial, $\langle \mathcal{R}, M_0 \rangle$, les affirmations suivantes peuvent être faites concernant la redondance et l'équivalence des GMEC :

1. Un GMEC (l, b) est redondant par rapport à un ensemble d'états $\mathcal{M} \subseteq \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$ si $\mathcal{M} \subseteq \mathcal{M}(l, b)$.
2. Un GMEC (l, b) est redondant par rapport à un système $\langle \mathcal{R}, M_0 \rangle$ si $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \subseteq \mathcal{M}(l, b)$.
3. Un ensemble des GMEC (L, b_v), avec $L = [l_1 \quad l_2 \quad ... \quad l_r]$ et $b_v = [b_1 \quad b_2 \quad ... \quad b_r]^T$, est redondant par rapport à un système $\langle \mathcal{R}, M_0 \rangle$ si (l_i, b_i) est redondant $\forall i = \overline{1, r}$.
4. Deux ensembles des GMEC, (L_1, b_{v_1}) et (L_2, b_{v_2}), sont équivalents par rapport à un système $\langle \mathcal{R}, M_0 \rangle$ si $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \cap \mathcal{M}(L_1, b_{v_1}) = \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \cap \mathcal{M}(L_2, b_{v_2})$.

Il est possible de simplifier les GMEC, afin de trouver une plus simple solution de contrôle. Des résultats intéressants à ce sujet ont été proposés dans [Dideban, 2007].

❑

Si le vecteur de pondération est binaire, $l \in \{0, 1\}^n$, alors le GMEC est appelé *non-pondéré*. Dans le cas des RdP saufs et conservatifs, il est possible de trouver un ensemble des GMEC non-pondérés équivalent pour tout GMEC pondéré. Pour ce type de RdP il est aussi toujours possible de trouver un ensemble des GMEC qui décrit le fonctionnement désiré du système. Cependant, ces propriétés ne sont pas valables dans le cas général ! Une discussion plus détaillée à ce sujet est menée dans la suite de ce mémoire.

Calcul du contrôleur :

Soit $\mathcal{R}_{\mathscr{P}} = (\mathcal{P}_{\mathscr{P}}, \mathcal{T}, \Sigma, E, \mathcal{W}_{\mathscr{P}}, M_{\mathscr{P}_{ini}})$ le RdP du procédé à contrôler, avec $\text{Card}[\mathcal{P}_{\mathscr{P}}] = n$, $\text{Card}[\mathcal{T}] = k$, et $\mathcal{W}_{\mathscr{P}} \in \mathbb{Z}^{k \times n}$. Et soit $\mathcal{R} = (\mathcal{P}, \mathcal{T}, \Sigma, E, \mathcal{W}, M_0)$, avec $\mathcal{P} = \mathcal{P}_{\mathscr{P}} \cup \mathcal{P}_{\mathscr{C}}$ et $\mathcal{W} = [\mathcal{W}_{\mathscr{P}}^T \quad \mathcal{W}_{\mathscr{C}}^T]^T$ et $M_0 = [M_{\mathscr{P}_{ini}}^T \quad M_{\mathscr{C}_{ini}}^T]^T$, le RdP du système contrôlé, où $\mathcal{P}_{\mathscr{C}}$ est l'ensemble des places de contrôle et $\mathcal{W}_{\mathscr{C}}$ et $M_{\mathscr{C}_{ini}}$ sont la matrice d'incidence de contrôleur et, respectivement, son marquage initial.

Pour simplifier le raisonnement, mais sans perte de généralité, supposons que la spécification de fonctionnement désiré pour $\mathcal{R}_{\mathscr{P}}$ soit décrite par une seule contrainte :

$$c_{\mathscr{S}} : l^T \cdot M \leq b, \ \forall\, M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}_{\mathscr{P}}, M_{\mathscr{P}_{ini}} \rangle}.$$

Nous savons (définition 1.4.8) qu'un invariant de marquage de $\mathcal{R}$ est défini par une relation de type : $x^T \cdot M = x^T \cdot M_0 = ct.$, $\forall\, M \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$, où x est un P-semiflot de dimension Card$[\mathcal{P}]$. Il s'ensuit que, pour incorporer la spécification dans l'ensemble des composantes conservatives de $\mathcal{R}$, il faut ajouter à l'inégalité de la contrainte une quantité $m_C = b - l^T \cdot M$. Cette quantité représente le marquage de la place de contrôle : $m_C = M(P_C)$, $P_C \in \mathcal{P}_{\mathscr{C}}$. L'invariant de marquage qui assure le respect de $c_{\mathscr{S}}$ dans $\mathcal{R}$ est alors donné par : $l^T \cdot M + M(P_C) = b \Leftrightarrow [l^T \ \ 1] \cdot M = b$.

Si le vecteur de pondération $x^T = [l^T \ \ 1]$ caractérise un invariant de marquage de $\mathcal{R}$, alors $x^T \cdot \mathcal{W} = 0 \Leftrightarrow [l^T \ \ 1] \cdot [\mathcal{W}_{\mathscr{P}}^T \ \ \mathcal{W}_{\mathscr{C}}^T]^T = 0$. Nous pouvons alors calculer la matrice d'incidence relative à P_C (ici c'est un vecteur) : $\mathcal{W}_{\mathscr{C}} = -l^T \cdot \mathcal{W}_{\mathscr{P}}$. Le marquage initial de P_C, $M_{\mathscr{C}_{ini}}$, est calculé à partir de l'invariant de marquage et de l'état initial du procédé, $M_{\mathscr{P}_{ini}}$: $[l^T \ \ 1] \cdot M_0 = b = l^T \cdot M_{\mathscr{P}_{ini}} + M_{\mathscr{C}_{ini}} \Rightarrow M_{\mathscr{C}_{ini}} = b - l^T \cdot M_{\mathscr{P}_{ini}}$, où $M_0 = [M_{\mathscr{P}_{ini}}^T \ \ M_{\mathscr{C}_{ini}}^T]^T$ est l'état initial du système contrôlé. Si $M_{\mathscr{C}_{ini}}$ est un nombre négatif, alors il n'existe pas de solution de contrôle qui peut imposer la spécification $c_{\mathscr{S}}$ au procédé $\mathscr{P}$.

En général, une spécification est décrite par un système d'inégalités de type (2.4.1) : $L^T \cdot M \leq b_v$. Dans ce contexte, $L = [l_1 \ \ l_2 \ \ \ldots \ \ l_{\text{Card}[\mathcal{P}_{\mathscr{C}}]}]$, $L \in \mathbb{Z}^{n \times \text{Card}[\mathcal{P}_{\mathscr{C}}]}$, représente la matrice de pondération des contraintes (la concaténation horizontale de tous les vecteurs de pondération $l_i \in \mathbb{Z}^n, i = \overline{1, \text{Card}[\mathcal{P}_{\mathscr{C}}]}$), et $b_v \in \mathbb{Z}^{\text{Card}[\mathcal{P}_{\mathscr{C}}]}$ est le vecteur des bornes. Tout P-semiflot X tel que $X^T = [L^T \ \ I]$ permet d'avoir l'invariant de marquage : $X^T \cdot M = b_v$, où I est la matrice d'identité de dimension Card$[\mathcal{P}_{\mathscr{C}}]$. La matrice d'incidence du contrôleur et son état initial sont donnés par les équations suivantes :

$$\mathcal{W}_{\mathscr{C}} = -L^T \cdot \mathcal{W}_{\mathscr{P}} \tag{2.4.2}$$

$$M_{\mathscr{C}_{ini}} = b_v - L^T \cdot M_{\mathscr{P}_{ini}}. \tag{2.4.3}$$

Remarque 2.4.2. Le modèle du contrôleur donné par l'expression (2.4.2) peut, dans le cas général être différent du modèle du procédé. Il peut être non-sauf ou généralisé, alors que le modèle du départ ne l'était pas.

❑

Exemple 2.4.1
Considérons le modèle de la figure 2.7a, et supposons que l'objectif du contrôle est de limiter la somme des marquages des places P_1 et P_4 à trois jetons. Nous avons donc la contrainte suivante sur le marquage du système :

$$c : m_1 + m_4 \leq 3.$$

La matrice d'incidence et le marquage initial du procédé sont déterminés à partir du modèle RdP :

$$\mathcal{W}_{\mathscr{P}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \quad ; \quad M_{\mathscr{P}_{ini}} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Il est maintenant possible de synthétiser le contrôleur par la méthode des invariants :

$$c : m_1 + m_4 \leq 3 \quad \Rightarrow \quad \begin{cases} L^T = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \\ b = \begin{bmatrix} 3 \end{bmatrix} \end{cases}$$

$$\mathcal{W}_{\mathscr{C}} = -L^T \cdot \mathcal{W}_{\mathscr{P}} = \begin{bmatrix} -1 & 1 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$M_{\mathscr{C}_{ini}} = b - L^T \cdot M_{\mathscr{P}_{ini}} = \begin{bmatrix} 1 \end{bmatrix}$$

La place de contrôle, P_C, gère les entrées et les sorties des places P_1 et P_4 et force le système à observer l'invariant $m_1 + m_4 + m_C = 3$. Les marquages des places P_1 et P_4 sont bornées dans le modèle contrôlé (figure 2.7b).

❑

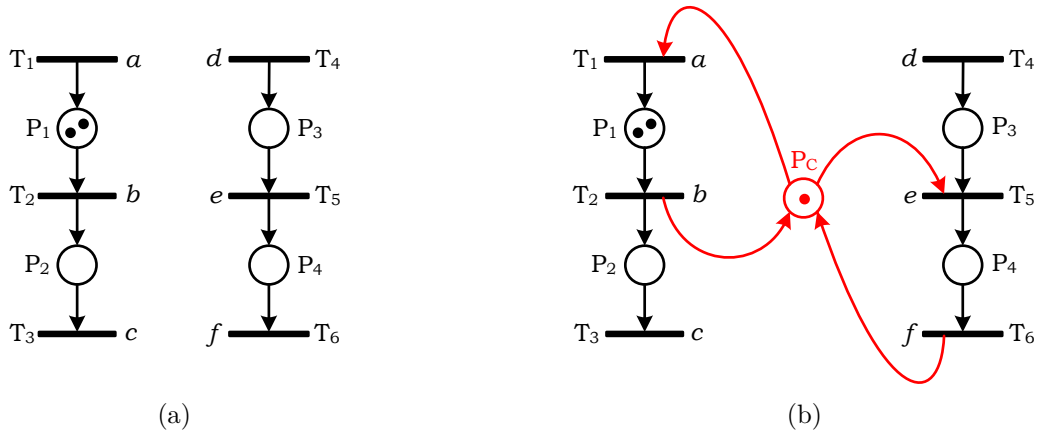


FIGURE 2.7 – Contrôle par la méthode des invariants : a) Procédé en isolation ; b) Procédé contrôlé

Problème des synchronisations incontrôlables :

Malheureusement, le contrôleur obtenu de cette manière ne peut garantir le respect de la spécification que si tous les événements impliqués dans l'évolution du procédé sont

contrôlables. Dans la situation où l'alphabet d'événements incontrôlables du procédé n'est pas vide, il est possible que des états interdits supplémentaires soient générés lors de la synchronisation des modèles du procédé et du contrôleur. Prenons le système de l'exemple de la figure 2.7. Si l'événement a est incontrôlable et si la place de contrôle P_C est vide, il est impossible d'interdire le franchissement de T_1, et $m_1 + m_4 = 4$!

En effet, si la synchronisation est réalisée *via* des événements incontrôlables, il est possible d'obtenir des arcs qui partent d'une place de contrôle vers une transition incontrôlable. Comme il est impossible d'influencer les événements incontrôlables, il n'y a pas de solution ! Pour résoudre ce problème, des modifications des contraintes admissibles sont apportées pour assurer qu'aucune place de contrôle ne soit une place d'entrée d'un événement incontrôlable ([Moody & Antsaklis, 2000]). Cependant, cette approche a l'inconvénient de ne pas offrir, en général, la solution optimale.

La méthode des invariants impose un ensemble de contraintes linéaires sur le marquage du système. Ces contraintes sont considérées connues *a priori*. Sous cette hypothèse, la méthode pourrait toutefois permettre de calculer un contrôleur optimal même dans l'éventualité d'un procédé qui n'est pas complètement contrôlable. Le problème est de trouver l'ensemble *complet* des contraintes admissibles, c'est-à-dire l'ensemble des contraintes généralisées d'exclusion mutuelle qui prend en compte aussi les états interdits causés par une éventuelle synchronisation incontrôlable. Nous allons discuter en détail ce problème dans la suite de ce mémoire et proposer une méthode pour déterminer un tel ensemble des contraintes, qui permet le calcul des contrôleurs maximaux permissifs pour les modèles RdP ordinaires et généralisés.

2.4.2 La théorie des régions

Une autre méthode qui permet la synthèse des contrôleurs maximaux permissifs pour les RdP, dans la présence des transitions incontrôlables et des possibles contraintes de vivacité, est la théorie des régions ([Badouel et al., 1994], [Ghaffari et al., 2003*a*]). Cette approche est plus puissante que la méthode des invariants, dans le sens où elle permet de trouver une solution de contrôle optimale (si une telle solution existe) indépendamment du niveau de contrôlabilité du procédé. Cependant, la théorie des régions est beaucoup plus laborieuse (du point de vue complexité de calcul) que la méthode des invariants. De plus, il n'existe pas d'optimisation significative au niveau structurel, et la taille du contrôleur final peut être importante.

La synthèse des contrôleurs par la théorie des régions est basée sur le graphe de marquage. Par opposition à la méthode des invariants, qui est une approche structurelle, la théorie des régions utilise l'approche de R&W pour calculer le contrôleur pour le graphe de marquages accessibles du système en boucle fermée. L'applicabilité de l'approche est générale pour des modèles RdP bornés pour lesquelles l'ensemble $\mathcal{M}_L$ des marquages

légaux (autorisés) est contrôlable et peut être décrit par un ensemble de contraintes généralisées d'exclusion mutuelle. Les spécifications du système sont données sous la forme d'une liste de marquages interdits (ou «dangereux»), $\mathcal{M}_D$. L'objectif est de déterminer un ensemble de places de contrôle qui, une fois ajoutées au modèle du procédé, interdisent l'accès à ces états.

Formellement, le contrôleur doit désactiver toutes les transitions de l'ensemble $\Omega = \{(M,T) \mid M \in \mathcal{M}_L \text{ et } \exists\, M' \notin \mathcal{M}_L \text{ tel que } M|T\rangle M'\}$, où (M,T) dénote un ensemble (état, transition_associée). L'idée est de trouver un RdP pur ayant un ensemble des transitions, $\mathcal{T}$, et un graphe de marquages, $\mathcal{G}$, donnés.

La démarche de synthèse se fait en deux grandes étapes :

1. détermination du comportement maximal permissif légal et vivant en utilisant la théorie de R&W ,
2. construction des places de contrôle en utilisant la théorie des régions.

Soit M_0 le marquage initial du modèle RdP à commander et soit $\mathcal{G}_C$ le graphe d'accessibilité du modèle RdP du système commandé, $\mathcal{R}$. L'opération de détermination de l'ensemble des marquages autorisés commence avec l'identification de l'ensemble des marquages interdits. À partir de là, le graphe des marquages partiels, $\mathcal{G}_p$, peut être généré et les marquages interdits frontières peuvent être trouvés. L'ensemble des marquages autorisés et le comportement autorisé $\mathcal{G}_C$, sont obtenus en inhibant toute transition contrôlable qui mène à un marquage bloquant ou interdit. Le graphe $\mathcal{G}_C$ obtenu correspond au comportement vivant maximal permissif du système commandé.

En ce qui concerne la deuxième étape de la synthèse, la théorie des régions stipule quatre conditions générales qui doivent être observées par un RdP $\mathcal{R}$ pour que son ensemble des états accessibles soit égal à un graphe des marquages $\mathcal{G}$ donné :

1. *L'équation de cycle* impose que la somme des équations d'état soit nulle pour les marquages de tout cycle ;
2. *La condition d'accessibilité* de tout marquage M à partir du marquage initial M_0 spécifie le fait que la somme des équations d'état doit être positive pour tous les marquages du chemin $M_0 \to M$; le chemin peut être choisi arbitrairement ;
3. *La condition de séparation d'événements* impose que pour qu'une place P puisse interdire le franchissement d'une transition T à partir d'un état M vers un état M', $M|T\rangle M'$, il est impératif (condition suffisante et nécessaire) que son marquage dans l'état M' soit négatif, $M'(P) < 0$;
4. *La condition de séparation d'états* précise le fait que les marquages doivent être différents l'un de l'autre.

Étant donné le modèle RdP du procédé, seules les places de contrôle doivent être construites pour implémenter la commande. La condition de séparation d'états est implicite et pour assurer la condition de séparation d'événements pour le système contrôlé

il suffit de l'assurer pour Ω. Il s'ensuit que tout marquage de contrôle doit nécessairement satisfaire :

- les équations de cycle (pour assurer la réversibilité du $\mathcal{R}$),
- la condition d'accessibilité (pour ne pas transgresser sur l'accessibilité du comportement autorisé $\mathcal{G}_C$),
- la condition de séparation d'événements pour tout marquage $M \in \Omega$ (pour bien interdire toutes les transitions indésirables).

Pour conclure, 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, ce qui a des conséquences défavorables pour l'implémentation.

2.4.3 Contrôle des blocages en utilisant les siphons

La notion structurelle de *siphon* peut être utilisée pour faire face au situations où les états interdits sont causés par des problèmes de vivacité et blocage ([Ezpeleta et al., 1995]). Un siphon est un ensemble de places, $\mathcal{P}_s$, tel que l'ensemble des transitions d'entrée de $\mathcal{P}_s$ est inclus dans son ensemble des transitions de sortie. Le blocage apparaît lorsque le siphon devient vide.

L'idée principale de la méthode de contrôle par des siphons est de trouver tous les siphons et de synthétiser un contrôleur de telle façon à ce qu'ils (les siphons) ne deviennent jamais vides. Cette condition peut être décrite par une contrainte linéaire et le contrôleur peut être calculé en utilisant la méthode des invariants présentée dans la section 2.4.1. Cependant, la solution n'est pas toujours facile. Il peut arriver que l'addition de la place de contrôle crée un nouveau siphon, qui repose le même problème.

Le nombre des siphons est en général grand, ce qui entraîne l'addition d'un grand nombre de places de contrôle. Pour résoudre ce problème, des méthodes basées sur les siphons élémentaires ont été proposées ([Iordache et al., 2001], [Wu & Zhou, 2004]). Le problème principal de ces approches est le manque de garantie de l'optimalité du contrôleur, car les transitions incontrôlables ne sont pas prises en compte.

2.4.4 Contrôle par retour d'état

L'idée de contrôle par retour d'état a été introduite par E. Holloway et H. Krogh dans ([Holloway & Krogh, 1990]). L'approche utilise les RdP pour contrôler les systèmes qui peuvent être modélisés par des graphes d'événements cycliques et saufs. Ces types de système constituent une classe particulière de RdP, où il n'y a pas de possibilités de conflit, de choix, ni de cumul. La méthode stipule qu'une transition T_k peut être franchie

seulement si elle est validée (*i.e.* il y a des jetons dans toutes les places d'entrée de T_k) et s'il y a des jetons fictifs dans les places de contrôle de T_k. Le contrôle peut être vu comme un commutateur binaire associé à la transition, *i.e.* il laisse ou ne laisse pas passer les jetons des ports d'entrée aux ports de sortie (figure 2.8).

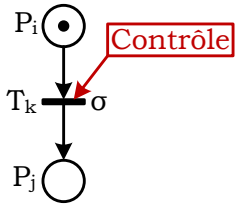


FIGURE 2.8 – Transition contrôlée par retour d'état

La méthode synthétise des contrôleurs maximaux permissifs en boucle fermée qui garantissent qu'aucun état interdit ne sera jamais atteint. La synthèse est effectuée sans générer ou analyser l'espace d'états du système, en utilisant la structure de modèle RdP de celui-ci. Le contrôle est réalisé par l'inhibition des événements appartenant au sous-ensemble Σ_C des événements contrôlables. Cet ensemble n'est pas spécifié directement, mais il est défini en munissant certaines transitions par des places de contrôle.

La résolution du problème d'états interdits est basée sur la structure de graphe cyclique et sauf d'événements. Un intérêt particulier est donné à l'effet du contrôle sur les marquages des places appartenant aux chemins commençant avec des places situées en aval des transitions contrôlées. Ces chemins, appelés *chemins d'influence*, sont utilisés pour identifier les places influencées par chaque contrôle. A partir de là, la méthode montre qu'il est possible de synthétiser un contrôle qui interdit l'accessibilité de n'importe quelle place appartenant à un chemin d'influence donné. L'approche utilise *des conditions d'interdiction* au niveau *d'une place*, *d'un ensemble de places* ou *d'une classe des places*. Toute condition d'interdiction peut être écrite comme une condition d'interdiction au niveau de classe.

Une extension ([Holloway et al., 1996]) a été ultérieurement apportée à la méthode de contrôle par retour d'état afin d'accroitre son applicabilité à une classe très large de RdP, y compris les réseaux non-saufs. Le problème majeur de cette approche est la complexité des expressions logiques associées aux transitions de contrôle, qui reste souvent très grande en dépit de toutes les techniques de simplification développées ([Kumar & Holloway, 1996], [Ghaffari et al., 2003*b*], [Dideban & Alla, 2006], [Vasiliu et al., 2009]).

2.5 Conclusions

Le but de la synthèse de contrôle est de construire un contrôleur $\mathscr{S}$ capable de restreindre le fonctionnement d'un procédé $\mathscr{P}$ donné de telle manière qu'il respecte à tout moment le cahier des charges imposé, $\mathscr{S}_{spec}$. De plus, le superviseur doit assurer le degré minimal nécessaire d'intrusion dans le fonctionnement du procédé. La contrôlabilité joue un rôle important dans la commande. Le contrôleur observe l'évolution du procédé et peut à tout moment interdire l'exécution des événements contrôlables de $\mathscr{P}$.

Le résultat principal pour le contrôle des SED est la théorie générale de la supervision, développée par Ramadgeg et Wonham. Cette approche calcule le contrôleur maximal permissif pour les systèmes modélisés par des automates à états finis. Malheureusement, cette méthode est très sensible à l'explosion combinatoire du nombre d'états et très souvent la taille du contrôleur dépasse celle du procédé en isolation. Par conséquent, cette approche peut être jugée comme peu convenable pour le contrôle des systèmes de taille moyenne ou grande.

Pour maîtriser la taille des contrôleurs, d'autres formalismes de modélisation ont été étudiés. Dans ce contexte, et grâce aux avantages discutés dans le chapitre 1, le réseau de Petri est devenu un des plus populaires formalismes alternatifs pour la synthèse de contrôle des SED. Même si une théorie unifiée applicable à tout modèle RdP n'a pas pu être conçue jusqu'à maintenant, il existe toutefois une variété de méthodes établies de synthèse de contrôleurs pour des différentes classes de modèles. Remarquable parmi elles est la méthode des invariants, qui exploite les propriétés structurelles des réseaux de Petri ; c'est celle-ci qui a été retenue comme méthode fondamentale de synthèse de contrôleur pour le travail présenté dans la suite de ce mémoire. Cependant, l'utilisation des RdP pour le contrôle des SED reste un sujet de recherche très prolifique, et de nombreuses nouvelles techniques, telles que la technique de déploiement présentée dans [Giua & Xie, 2005], sont encore en cours de développement. De nombreux travaux enquêtent aussi sur le problème de l'observabilité dans le contexte des RdP ([Giua & Seatzu, 2002], [Corona et al., 2004], [Giua, Seatzu & Júlvez, 2004], [Giua, Seatzu & Basile, 2004], [Giua et al., 2005]).

Notre travail est situé en amont de la méthode des invariants, car celle-ci ne fournit pas de solution optimale dans le cas général. Notre recherche vise à déterminer des contraintes admissibles pour garantir l'optimalité. Nos objectifs principaux sont de combler le fossé entre le graphe de marquages et la méthode d'invariants et d'avancer une solution où cette méthode échoue. Dans les chapitres suivants nous allons débattre en détail autour de ces idées et proposer des techniques pour la détermination des contraintes permettant la synthèse de contrôleurs maximaux permissifs pour les RdP synchronisés saufs et non-saufs. Nous allons aussi traiter un objectif secondaire d'une haute importance pratique, qui est celle de l'optimalité structurelle des contrôleurs.

Chapitre 3

Contraintes et optimalité

Ce chapitre donne une vision générale sur la problématique traitée dans cette thèse — la détermination des contraintes permettant la synthèse, via la méthode des invariants, des contrôleurs maximaux permissifs pour les RdP synchronisés. Nous discutons les implications de l'optimalité dans le contexte des modèles RdP et nous investiguons le passage depuis un état interdit vers la contrainte qui le bloque. L'idée d'optimalité structurale est aussi brièvement examinée vers la fin du chapitre.

3.1 Introduction

Étant donné un système à événements discrets et le cahier des charges qui décrit son fonctionnement désiré, les deux objectifs fondamentaux du contrôle sont : (1) de limiter le comportement du système en boucle fermée aux contraintes imposées par la spécification, et (2) d'assurer le fait que le fonctionnement du système n'est pas soumis à aucune restriction inutile. Le contrôle optimal ne peut être accompli que si la condition de permissivité maximale est également assuré. Par ailleurs, nous nous intéressons aussi à un troisième aspect, plus pratique — la minimalité structurelle du contrôleur.

Nous avons discuté, dans le chapitre précédent, les différentes méthodes de synthèse des contrôleurs pour les SED. Nous avons vu que, malgré sa généralité et l'optimalité de la solution de contrôle qu'elle offre, la théorie générale de la supervision des SED est peu adaptée à l'implémentation des systèmes de taille réelle. Ce problème est majoritairement causé par l'utilisation des modèles automates pour décrire les comportements du procédé et de la spécification, modèles qui sont notoirement sensibles au problème de l'explosion combinatoire du nombre d'états. Cependant, si le réseau de Petri est choisi en tant

que formalisme de modélisation, le gain en concision et richesse de représentation est contrebalancé par la perte de la généralité et / ou de l'optimalité des résultats.

Toutefois, les nombreux avantages offerts par les réseaux de Petri ont motivé le développement de plusieurs théories de contrôle pour des classes de modèles de ce type. Une des plus générales et élégantes de ces théories est la méthode des invariants, comme nous l'avons vu dans le chapitre 2. Elle offre une solution de contrôle optimal et efficace (de point de vue complexité de calcul) pour tout RdP généralisé, à condition que la spécification soit décrite par un ensemble complet et approprié de contraintes généralisées d'exclusion mutuelle. De plus, l'approche est structurelle, ce qui permet d'effectuer tous les calculs hors ligne.

Malheureusement, le problème de la détermination d'un ensemble approprié et complet de contraintes qui permet le calcul de contrôleur maximal permissif se complique beaucoup avec l'intervention de l'incontrôlabilité dans les modèles. La cause principale des difficultés est la génération des états interdits supplémentaires suivant une synchronisation incontrôlable entre le modèle du procédé et celui de la spécification. Ces états ne peuvent pas être prédits à l'avance, ce qui rend la détermination des contraintes les interdisant très difficile.

Cependant, le problème peut être résolu en utilisant la théorie de R&W en conjonction avec le graphe de marquage du modèle (l'automate équivalent du RdP). En fait, une telle démarche nous permettra : 1) d'une part, de réunir les avantages des deux méthodes — la solution formelle, générale et optimale de R&W (basée sur les propriétés des langages) et l'élégance, l'efficacité de calcul et l'aisance d'implémentation de la méthode des invariants — 2) et d'autre part d'éviter leurs inconvénients — l'explosion combinatoire du nombre d'états et, respectivement, le problème des synchronisations incontrôlables. De plus, la démarche est exécutée entièrement hors ligne, le graphe de marquage est calculé une seule fois, et en conséquence le temps de réponse en temps réel du système commandé n'est pas influencé.

C'est pour ces raisons que nous considérons qu'une symbiose entre la méthode des invariants et la théorie de R&W est très rentable pour la synthèse des contrôleurs pour les SED modélisés par des RdP ordinaires ou généralisés. Ainsi, contrairement à la théorie des régions, nous évitons de reconstruire le modèle complet du système contrôlé à partir du graphe de marquage décrivant son fonctionnement désiré. De plus, la technique de transformation d'un automate fini dans un RdP équivalent ne donne pas d'habitude le modèle RdP le plus intuitif, ni le plus simple. Un modèle RdP beaucoup plus adapté peut généralement être construit à partir de l'information structurelle donnée par le système physique lui même. D'ailleurs, nous disposons, dès le départ, des modèles RdP du procédé et de la spécification. La boucle fermée peut donc être obtenue en réalisant le composé synchrone des deux modèles, et seul le modèle du contrôleur reste à être déterminé à l'aide de la méthode des invariants.

Notre objectif principal est de combler le fossé entre le graphe de marquages et le calcul du contrôleur optimal. Notre travail est donc situé en amont de la méthode des invariants, et concerne l'étape de conception des contraintes admissibles pour cette dernière. Ce chapitre présente de manière détaillée la problématique traitée : le passage depuis le graphe de marquage et les ensembles d'états interdits vers un ensemble complet de contraintes admissibles permettant le calcul (avec la méthode des invariants) d'un contrôleur maximal permissif, en dépit de l'existence des événements incontrôlables dans les modèles. Nous insisterons sur la nécessité d'une équivalence, d'un chemin bijectif entre l'état interdit et sa contrainte linéaire associée. En outre, nous allons également traiter un objectif secondaire d'une haute importance pratique : l'optimalité structurelle des contrôleurs.

3.2 États autorisés et interdits d'un RdP

Étant donnés un RdP $\mathcal{R}_{\mathscr{P}}$ et une spécification $\mathcal{R}_{\mathscr{S}}$, le modèle en boucle fermée du système à contrôler peut être obtenu en réalisant la composition synchrone des deux : $\mathcal{R} = \mathcal{R}_{\mathscr{P}} \parallel_s \mathcal{R}_{\mathscr{S}}$ (définition 1.4.9). Une fois ce modèle construit, les différents sous-ensembles de l'espace d'états accessibles peuvent être déterminés en appliquant l'algorithme de Kumar sur le graphe de marquage de $\mathcal{R}$.

Exemple 3.2.1
Soit un système manufacturier constitué de deux machines, $\mathscr{M}_1$ et $\mathscr{M}_2$, qui travaillent en parallèle pour produire deux variétés différentes du même type de pièces. La machine $\mathscr{M}_2$ a une capacité d'usinage d'une seule pièce à la fois, alors que la capacité d'usinage de la machine $\mathscr{M}_1$ est de deux pièces à la fois. Les pièces fabriquées par les deux machines sont recueillies dans un stock commun. Des robots assurent le transfert des pièces vers le stock. Dans l'intérêt de la clarté de présentation, nous supposons que le fonctionnement des machines est idéal (*i.e.* elles ne peuvent pas tomber en panne).

Le cahier des charges du système stipule qu'il n'y a que trois robots dédiés au transport des pièces entre les machines et le stock. Les débuts des opérations de fabrication (événements d_i) sont les seuls événements contrôlables du système. Les fins d'usinage (événements f_i) et le retour de chaque robot à l'état «disponible» (événement r) sont incontrôlables.

Évidemment, pour un système si simple il serait très facile de réaliser, dès le début, un modèle minimal du fonctionnement en boucle fermée, modèle qui assurerait au même temps le respect de la spécification et le comportement maximal permissif, tout en évitant les synchronisations incontrôlables. Cependant, pour un système de taille réelle une telle modélisation serait très difficile et impliquerait des coûts trop importants. En réalité, dans l'étape

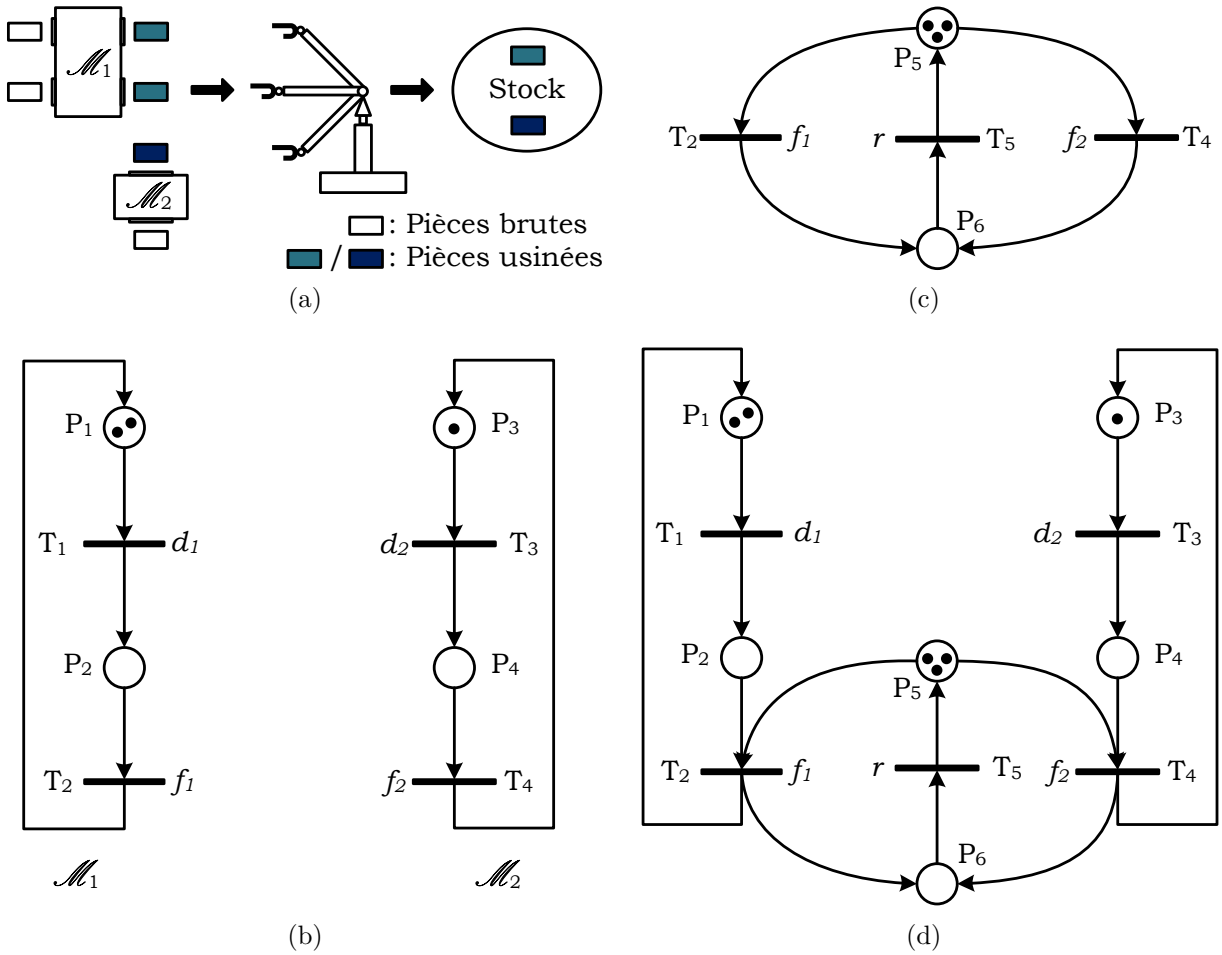


FIGURE 3.1 – Modèle du système manufacturier de l'exemple 3.2.1 : a) schéma du système manufacturier ; b) modèle RdP du procédé ; c) modèle RdP de la spécification ; d) modèle RdP du fonctionnement en boucle fermée.

de synthèse de contrôle nous sommes souvent obligés de travailler avec des modèles déjà existants des procédés et / ou des spécifications. Ces modèles ne sont pas toujours les meilleurs possibles. C'est pour ces raisons que nous avons choisi d'utiliser dans ce travail des modèles didactiques, comportant des synchronisations incontrôlables entre le procédé et la spécification. Cette représentation nous permettra d'illustrer de manière claire et explicite la problématique étudiée et, dans la suite de ce mémoire, de formuler et puis valider nos solutions sur des exemples simples et faciles à suivre. Les solutions seront aussi implémentées et validées sur des études de cas comportant des systèmes complexes d'exploitation.

La figure 3.1 présente le schéma du système en boucle fermée et les modèles RdP du procédé, de la spécification, et du fonctionnement désiré en boucle fermée. La signification des places est comme suit :

- P_1/P_3 : la machine $\mathscr{M}_1/\mathscr{M}_2$ est à l'arrêt ;
- P_2/P_4 : la machine $\mathscr{M}_1/\mathscr{M}_2$ est en cours d'usinage d'une pièce ;
- P_5 : nombre des robots disponibles ;
- P_6 : nombre des robots impliqués dans des opérations de transport.

❑

Les états interdits sont déterminés en exécutant la dynamique en boucle fermée, en concordance avec les règles de franchissement des RdP : tout marquage tel qu'il existe une transition validée dans $\mathcal{R}_{\mathscr{P}}$ mais non-validée par $\mathcal{R}_{\mathscr{S}}$ doit être interdit. En conséquence, chaque fois qu'un tel marquage est rencontré, la branche correspondante du graphe de marquage est coupée. Il est maintenant possible d'identifier les états interdits supplémentaires, générés par les synchronisations incontrôlables entre le modèle du procédé et celui de la spécification. Pour y arriver, il faut remonter les branches dans le graphe de marquage pour détecter les séquences d'événements incontrôlables qui mènent le système en boucle fermée vers les états interdits. Nous pouvons ainsi trouver l'ensemble d'états faiblement interdits et celui d'états interdits frontières (figure 3.2). Cette manière de déterminer les états interdits est formellement définie ci-dessous.

Soient $\mathcal{R}_{\mathscr{P}}$ le modèle RdP du procédé à contrôler, $\mathcal{R}_{\mathscr{S}}$ celui de la spécification à lui imposer, et $\mathcal{R} = (\mathcal{P}, \mathcal{T}, \Sigma, E, \mathcal{W}, M_0)$ le modèle RdP du système en boucle fermée correspondant. Soient $\mathcal{M}_{Acs}^{\langle \mathcal{R}_{\mathscr{P}}, M_{\mathscr{P}_0} \rangle}$, $\mathcal{M}_{Acs}^{\langle \mathcal{R}_{\mathscr{S}}, M_{\mathscr{S}_0} \rangle}$ et, respectivement, $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$, les espaces d'états accessibles de ces systèmes. Soient $\mathcal{T}_C$ et $\mathcal{T}_U$ les ensembles des transitions associés aux événements contrôlables et, respectivement, incontrôlables de $\mathcal{R}$, avec $\mathcal{T} = \mathcal{T}_U \cup \mathcal{T}_C$, et soient $\mathcal{T}_C^*$, et $\mathcal{T}_U^*$ les ensembles des mots pouvant être construits sur $\mathcal{T}_C$ et $\mathcal{T}_U$. Formellement, nous pouvons définir les sous-ensembles suivants de l'espace d'états accessibles de $\mathcal{R}$:

Définition 3.2.1. *L'ensemble $\mathcal{M}_{\mathcal{I}}$ d'états interdits de $\mathcal{R}$ comprend tout état $M_i \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$ pour lequel une transition incontrôlable $T_u \in \mathcal{T}_U$ peut être trouvée telle que $M_i | T_u \rangle M_j$, avec $M_j \in \mathcal{M}_{Acs}^{\langle \mathcal{R}_{\mathscr{P}}, M_{\mathscr{P}_0} \rangle}$ et $M_j \notin \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$:*

$$\mathcal{M}_{\mathcal{I}} = \left\{ M_i \,\middle|\, M_i \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}, \exists\, T_u \in \mathcal{T}_U : M_i | T_u \rangle M_j, M_j \in \mathcal{M}_{Acs}^{\langle \mathcal{R}_{\mathscr{P}}, M_{\mathscr{P}_0} \rangle} \text{ et } M_j \notin \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \right\}$$

❑

Définition 3.2.2. *L'ensemble $\mathcal{M}_{\mathcal{F}}$ des états faiblement interdits de $\mathcal{R}$ comprend tout état $M_i \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$ qui n'est pas lui même interdit mais à partir duquel un état interdit $M_j \in \mathcal{M}_{\mathcal{I}}$ peut être atteint via une séquence des transitions incontrôlables $\sigma_u \in \mathcal{T}_U^*$:*

$$\mathcal{M}_{\mathcal{F}} = \left\{ M_i \,\middle|\, M_i \in \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}, M_i \notin \mathcal{M}_{\mathcal{I}}, \exists\, \sigma_u \in \mathcal{T}_U^* : M_i | \sigma_u \rangle M_j, M_j \in \mathcal{M}_{\mathcal{I}} \right\}$$

❑

Définition 3.2.3. *L'ensemble $\mathcal{M}_{\mathcal{A}}$ des états autorisés de $\mathcal{R}$ est défini comme ci-dessous :*

$$\mathcal{M}_{\mathcal{A}} = \mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle} \setminus \{\mathcal{M}_{\mathcal{I}} \cup \mathcal{M}_{\mathcal{F}}\}$$

❑

Définition 3.2.4. *L'ensemble $\mathcal{M}_{\mathcal{B}}$ des états interdits frontières de $\mathcal{R}$ comprend tout état interdit ou faiblement interdit accessible depuis un état autorisé via une transition contrôlable :*

$$\mathcal{M}_{\mathcal{B}} = \left\{ M_{B_i} \,|\, M_{B_i} \in (\mathcal{M}_{\mathcal{I}} \cup \mathcal{M}_{\mathcal{F}}) \;\; et \; \exists \, M_{A_j} \in \mathcal{M}_{\mathcal{A}} \; et \; \exists \, T_c \in \mathcal{T}_C \; tels \; que \;\; M_{A_j} \xrightarrow{T_c} M_{B_i} \right\}$$

❑

Définition 3.2.5. *L'espace pertinent d'états accessibles du système, $\mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$, est défini comme suit :*

$$\mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle} = \mathcal{M}_{\mathcal{A}} \cup \mathcal{M}_{\mathcal{B}}$$

❑

Remarques 3.2.1.

a) L'ensemble $\mathcal{M}_{\mathcal{A}}$ correspond au contrôleur maximal permissif obtenu avec la théorie de Ramadge & Wonham.

b) L'espace pertinent d'états accessibles, $\mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$ contient toute l'information nécessaire pour la synthèse de contrôleur.

❑

Retournant au système de l'exemple 3.2.1, la figure 3.2 donne le graphe des marquages accessibles du fonctionnement en boucle fermée et montre les différents types d'états interdits. Nous voyons bien que, quand le système est dans l'état $P_1P_2P_3P_6^3$, la machine $\mathscr{M}_1$ est en cours d'usinage, alors qu'il n'y a pas de robot disponible pour prendre en charge la pièce éjectée par la machine. Comme la fin de l'opération de fabrication, ainsi que le retour du robot, sont des événements incontrôlables, il est alors impossible de savoir s'il y aura un robot disponible pour prendre en charge la pièce usinée au moment ou elle est éjecté par la machine. Il faut donc interdire cet état, afin d'assurer que la pièce n'est pas perdue.

En remontant les branches du graphe de marquage à partir de cet état, nous trouvons trois états faiblement interdits, qui peuvent donc mener le système vers $P_1P_2P_3P_6^3$ sous l'action d'événements incontrôlables :

– $P_2^2P_4P_5^2P_6$ — trois pièces sont en cours d'usinage, alors que juste deux robots sont disponibles,

– $P_1P_2P_4P_5P_6^2$ — une pièce est en cours de fabrication sur chaque machine, alors qu'il y a un seul robot disponible dans le système, et
– $P_2^2P_3P_5P_6^2$ — la machine $\mathscr{M}_1$ est en train de travailler sur deux pièces, alors qu'un seul robot est disponible pour les transporter vers le stock.

Dans ce cas là, tous ces états sont aussi des états interdits frontières.

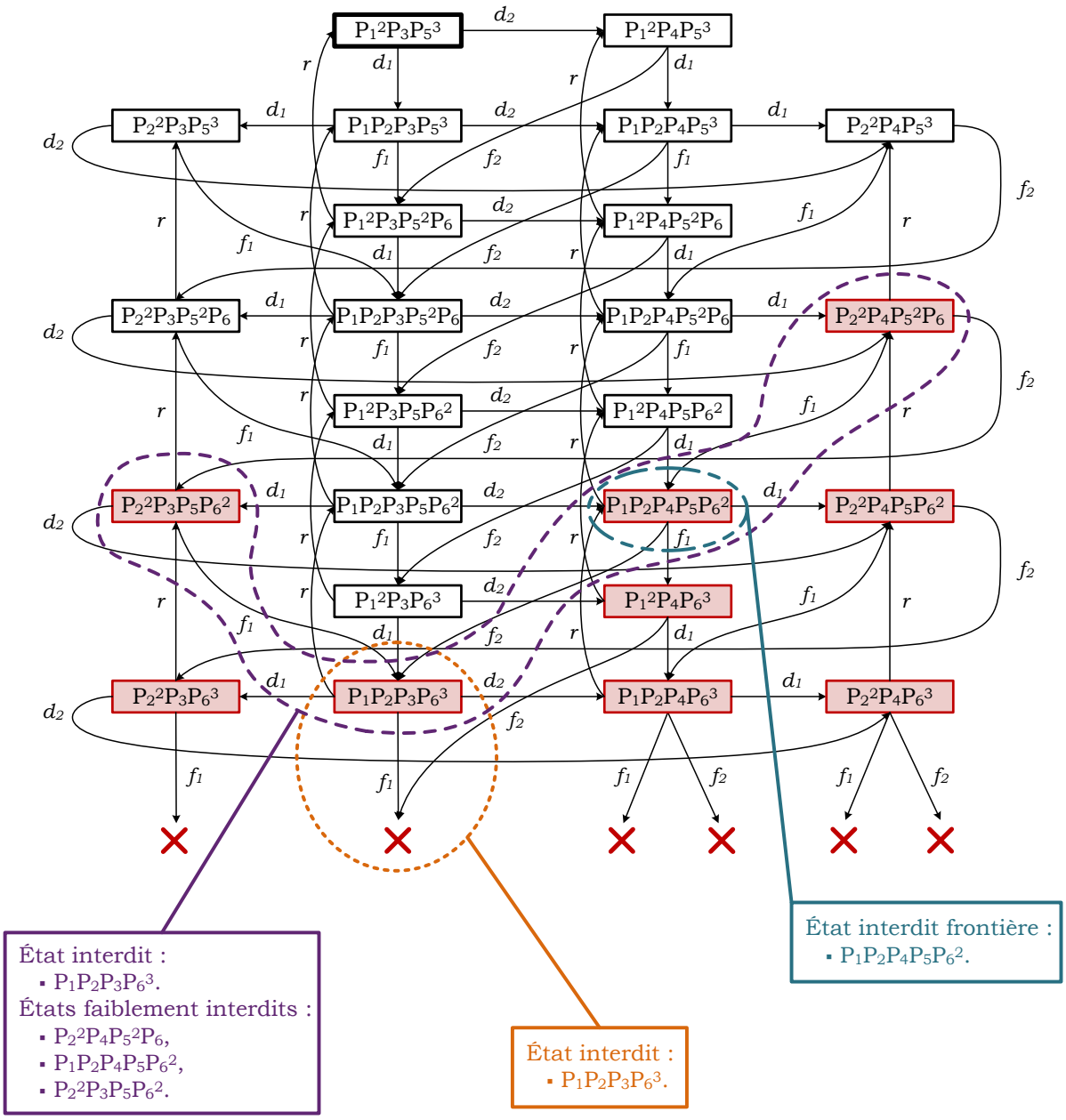


FIGURE 3.2 – Graphe de marquages accessibles du système de l'exemple 3.2.1

Les ensembles complets d'états interdits et d'états autorisés pour ce système, et son ensemble pertinent d'états accessibles sont le suivants :

- $\mathcal{M}_{\mathcal{I}} = \{P_1P_2P_3P_6^3\ ;\ P_2^2P_3P_6^3\ ;\ P_1^2P_4P_6^3\ ;\ P_1P_2P_4P_6^3\ ;\ P_2^2P_4P_6^3\ \}$,
- $\mathcal{M}_{\mathcal{F}} = \{P_2^2P_3P_5P_6^2\ ;\ P_1P_2P_4P_5P_6^2\ ;\ P_2^2P_4P_5^2P_6\ ;\ P_2^2P_4P_5P_6^2\ \}$,
- $\mathcal{M}_{\mathcal{B}} = \{P_1P_2P_3P_6^3;\ P_2^2P_3P_5P_6^2\ ;\ P_1P_2P_4P_5P_6^2\ ;\ P_1^2P_4P_6^3\ ;\ P_2^2P_4P_5^2P_6\ \}$,
- $\mathcal{M}_{\mathcal{A}} = \{P_1^2P_3P_5^3;\ P_1P_2P_3P_5^3\ ;\ P_1^2P_3P_5^2P_6\ ;\ P_1P_2P_3P_5^2P_6\ ;\ P_1^2P_3P_5P_6^2\ ;\ P_1P_2P_3P_5P_6^2\ ;\ P_1^2P_3P_6^3\ ;\ P_1^2P_4P_5^3\ ;\ P_1P_2P_4P_5^3\ ;\ P_1^2P_4P_5^2P_6\ ;\ P_1P_2P_4P_5^2P_6\ ;\ P_1^2P_4P_5P_6^2\ ;\ P_2^2P_3P_5^2P_6\ ;\ P_2^2P_3P_5^3\ ;\ P_2^2P_4P_5^3\ \}$,
- $\mathcal{M}_{Acs-l}^{\langle\mathcal{R},M_0\rangle} = \{P_1^2P_3P_5^3;\ P_1P_2P_3P_5^3\ ;\ P_1^2P_3P_5^2P_6\ ;\ P_1P_2P_3P_5^2P_6\ ;\ P_1^2P_3P_5P_6^2\ ;\ P_1P_2P_3P_5P_6^2\ ;\ P_1^2P_3P_6^3\ ;\ P_1^2P_4P_5^3\ ;\ P_1P_2P_4P_5^3\ ;\ P_1^2P_4P_5^2P_6\ ;\ P_1P_2P_4P_5^2P_6\ ;\ P_1^2P_4P_5P_6^2\ ;\ P_2^2P_3P_5^2P_6\ ;\ P_2^2P_3P_5^3\ ;\ P_2^2P_4P_5^3\ ;\ P_1P_2P_3P_6^3\ ;\ P_2^2P_3P_5P_6^2\ ;\ P_1P_2P_4P_5P_6^2\ ;\ P_1^2P_4P_6^3\ ;\ P_2^2P_4P_5^2P_6\ \}$.

Le graphe de marquage correspondant à l'ensemble $\mathcal{M}_{Acs-l}^{\langle\mathcal{R},M_0\rangle}$ est donné dans la figure 3.3.

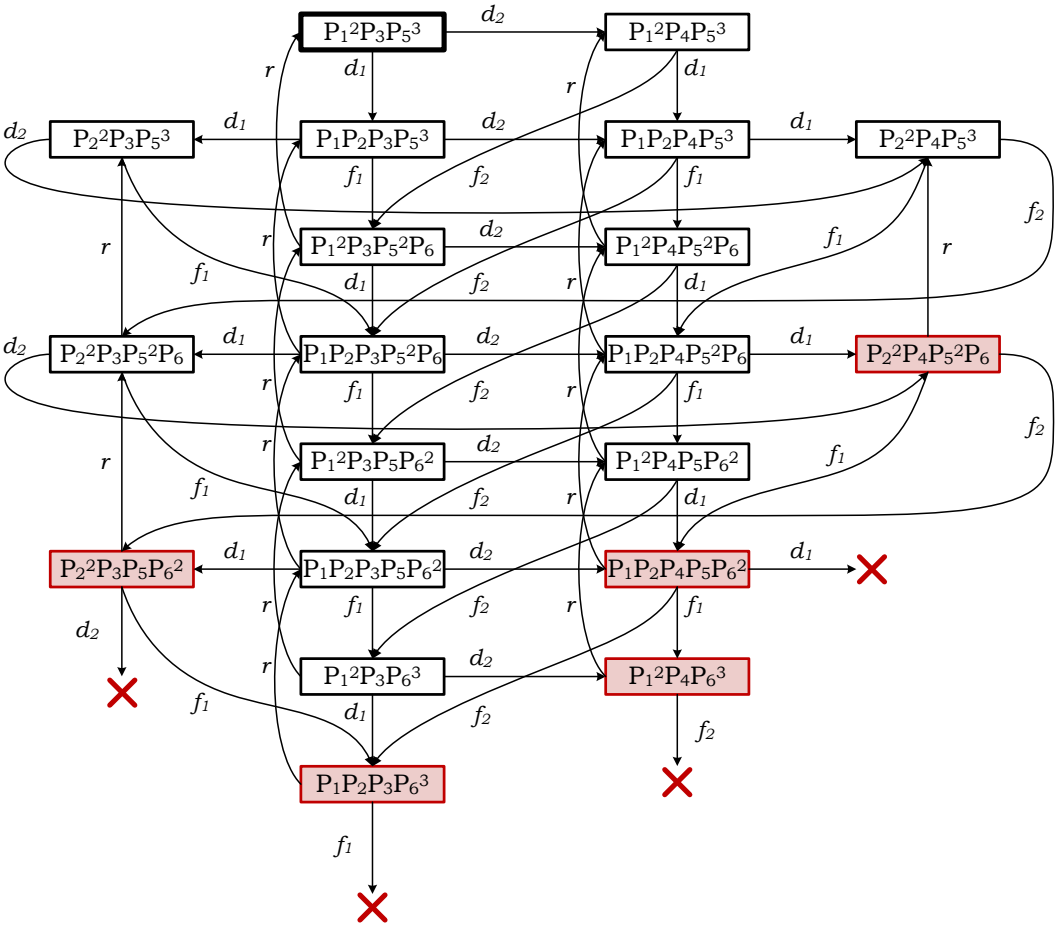


FIGURE 3.3 – Graphe pertinent de marquage du système de l'exemple 3.2.1

Une fois tous les ensembles d'états interdits et autorisés identifiés, il est possible de calculer le langage suprême contrôlable du système en boucle fermée en appliquant l'algorithme de Kumar sur son graphe de marquage. Malheureusement, ce type de synthèse de contrôleur implique la suppression du graphe de marquage des ensembles d'états interdits, faiblement interdits et interdits frontière. Or cette opération, évidente pour les automates, n'est pas du tout convenable pour les modèles RdP, car elle rendrait presque impossible le retour au modèle initial du système. Il faut donc prendre en compte les particularités du formalisme de modélisation dans le calcul du contrôleur et utiliser une approche de synthèse structurelle, telle que la méthode des invariants. Cependant, les ensembles d'états peuvent se montrer très utiles pour la détermination des contraintes admissibles pour la méthode des invariants.

3.3 Contraintes admissibles

Un contrôleur optimal est caractérisé par sa capacité d'assurer dans tout scénario de fonctionnement le respect de deux conditions : (1) tout comportement incompatible avec la spécification est interdit, et (2) tout autre comportement est permis. Dans le contexte des modèles RdP, cela signifie qu'il faut en même temps bloquer tous les états interdits frontières et garantir l'accessibilité de tous les états autorisés. Cela revient à assurer une équivalence entre chaque contrainte et son état interdit associé. Ceci est un problème difficile, car pendant que l'implication d'interdiction est triviale, assurer la réciproque est plus laborieux.

A présent, sur la base des résultats concernant les GMEC non-pondérés présentés dans [Giua et al., 1992], il est possible de déterminer un ensemble des contraintes admissibles permettant le calcul, avec la méthode des invariants, d'un contrôleur maximal permissif pour les RdP saufs et conservatifs. Un GMEC non-pondéré est alors associé à chaque état interdit frontière. Le nombre des contraintes est minimisé afin d'obtenir la solution de contrôle la plus simple possible. Pour assurer l'interdiction, l'ensemble d'éléments non-nuls du vecteur de pondération de chaque contrainte correspond à l'ensemble des places marquées dans son état associé, alors que sa borne est choisie égale au nombre des places marquées dans l'état concerné moins un. Par exemple, un état $M = P_1P_2P_5$ a trois places marquées : P_1, P_2 et P_5. Pour l'interdire il est donc suffisant d'imposer au système la condition $c : m_1 + m_2 + m_5 \leq 2$.

En général, étant donné un RdP sauf et conservatif $\mathcal{R}$ avec n places et un état interdit $M_{B_i} \in \mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$, $M_{B_i} = P_{i_1}P_{i_2}...P_{i_t}$, $t \leq n$, la contrainte interdisant M_{B_i} est :

$$c_i : \sum_{j=1}^{t} M(P_{i_j}) \leq b_i \tag{3.3.1}$$

où t est le nombre des places marquées dans l'état M_{B_i}, $M(P_{i_j}) = m_{i_j}$ est le marquage

de la place P_{i_j}, et $b_i = t - 1$ est la borne de la contrainte. Cette contrainte est prouvée grâce aux deux propriétés des RdP saufs et conservatifs :

1. dans un RdP sauf il est impossible d'avoir deux états (marquages) distincts avec le même support, où le support d'un marquage dénote l'ensemble des places marquées quand le réseau est dans l'état en question (définition 1.4.2), et
2. dans un RdP sauf et conservatif un marquage M_1 ne peut jamais être couvert par un autre marquage M_2 ($Support(M_1) \subset Support(M_2)$ jamais vrai).

Nous avons identifié deux problèmes auxquels nous allons proposer des solutions dans la suite de ce mémoire :

A) Dans le cas des RdP saufs mais non-conservatifs il est possible d'avoir des relations de couverture entre les états. Alors, si un état autorisé est couvert par un état interdit le contrôle calculé de la manière présenté précédemment ne peut pas être maximal permissif.

Considérons, par exemple, une variante non-conservative, du système présenté dans l'exemple 1.4.1. Dans ce scénario nous supposons que la spécification ne permet que le traitement d'une seule pièce par le système. Nous supposons aussi que le fonctionnement des machines est idéal (il y a pas de possibilité de panne). Ce type de comportement est rarement rencontré en pratique, mais nous l'utilisons pour mettre en évidence un contre-exemple. La nouvelle spécification et le système modifié, ainsi que son graphe pertinent de marquage, sont présentés dans la figure 3.4.

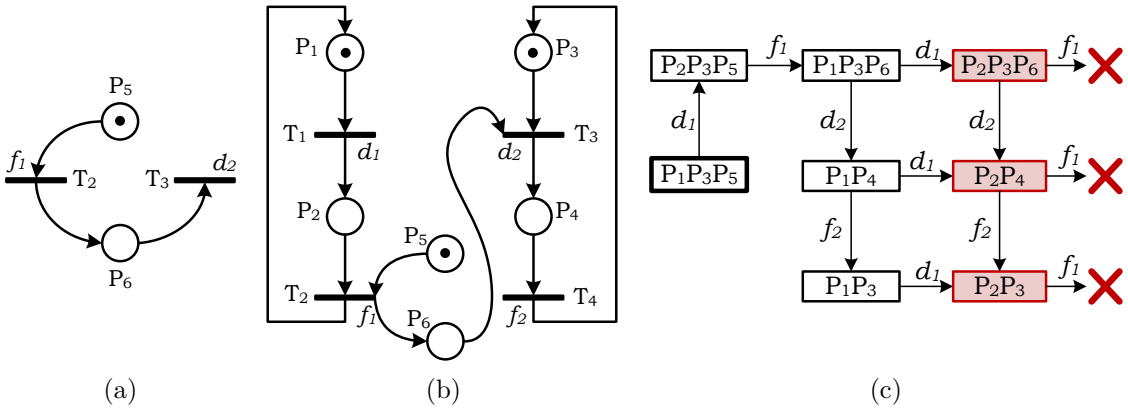


FIGURE 3.4 – Modèle RdP non-conservatif du système de l'exemple 1.4.1 : a) la spécification non-conservative ; b) le modèle du système en boucle fermée ; c) le graphe pertinent de marquage.

Les ensembles d'états autorisés et interdits frontière sont donc :
– $\mathcal{M}_{\mathcal{A}} = \{P_1P_3P_5, P_2P_3P_5, P_1P_3P_6, P_1P_4, P_1P_3\}$
– $\mathcal{M}_{\mathcal{B}} = \{P_2P_3P_6, P_2P_4, P_2P_3\}$
Si nous calculons le contrôleur avec la méthode des invariants nous obtenons le modèle contrôlé de la figure 3.5a. Il est évident (figure 3.5b) que ce contrôle est extrêmement

sous-optimal — il empêche le système de quitter même son état initial. Ce problème est causé par la relation de couverture entre l'état interdit P_2P_3 et l'état autorisé $P_2P_3P_5$. La solution habituelle pour dépasser cet inconvénient est de retourner à l'étape de modélisation du procédé et de la spécification, ce qui n'est pas toujours raisonnable. Nous allons proposer, dans le chapitre 4, une méthode systématique de détermination des contraintes admissibles permettant le calcul, avec la méthode des invariants, d'un contrôleur optimal maximal permissif sans retour à la modélisation.

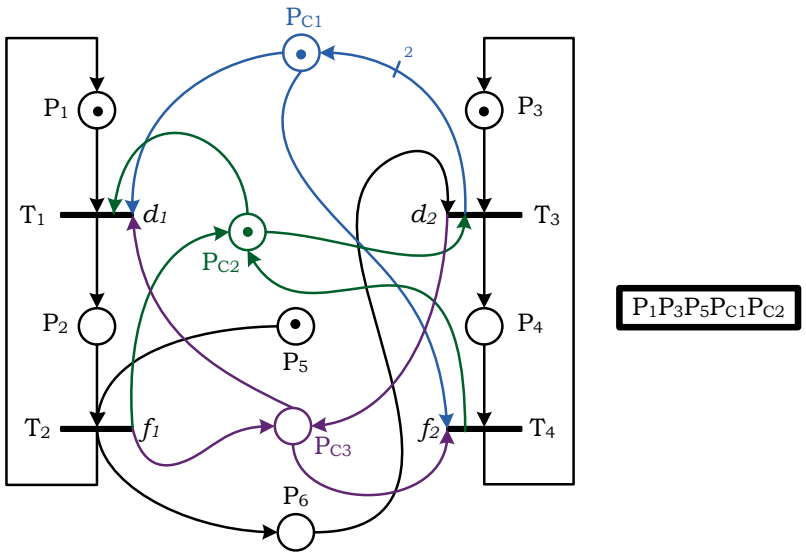


FIGURE 3.5 – Modèle contrôlé du RdP non-conservatif : a) le modèle du système en boucle fermée ; b) le graphe de marquage.

❑

B) Dans le cas des RdP non saufs en général le contrôle calculé en utilisant des contraintes de type (3.3.1) est sous-optimal.

Considérons l'exemple 3.2.1, traité dans la section précédente, et prenons, par exemple, l'état interdit $P_1P_2P_3P_6^3$. Cet état correspond au scénario où la machine $\mathscr{M}_1$ est en train de fabriquer une pièce, alors qu'aucun robot n'est disponible pour la transporter vers le stock. Il y a donc six jetons dans le système : trois jetons indiquant le nombre des robots engagés (P_6^3), et trois indiquant l'état des deux machines ($\mathscr{M}_1$ fabrique une pièce — un jeton dans chacune des places P_1 et P_2 — et $\mathscr{M}_2$ est en repos — un jeton dans P_3). Si nous calculons la contrainte associée à cet état en limitant la somme des jetons dans les places concernées, nous obtenons (relation (3.3.1)) : $M(P_1) + M(P_2) + M(P_3) + M(P_6) \leq 5$.

Cette contrainte n'est pas admissible, car une étude minutieuse des ensembles d'états du système révèle le fait que, même si elle interdit bien l'état $P_1P_2P_3P_6^3$, elle interdit aussi un état autorisé : $P_1^2P_3P_6^3 \in \mathcal{M}_{\mathcal{A}}$.

Ce problème est une conséquence de l'idée de l'unicité du support, qui est à la base des GMEC non-pondérés. Cependant, dans le cas des RdP non-saufs, cette hypothèse ne tient plus. Les superpositions (totales ou partielles) de support entre les états interdits et les états autorisés fait que la solution intuitive de contrôle — la limitation de la somme non-pondérée des jetons — devient trop restrictive pour cette classe de modèles.

Même si la transgression sur les comportements désirés du système n'est pas trop grave dans ce cas particulier, il faut toutefois tenir compte du fait que ceci est un exemple très simple. Rejeter les implications de cette hyper-restrictivité peut s'avérer très dangereux pour un système plus complexe, où elle peut entraîner l'interdiction d'un très grand pourcentage de comportements désirables du système. Il faut donc trouver une technique de détermination des contraintes admissibles qui assure un passage bijectif (une équivalence) entre l'ensemble des états interdits et celui des contraintes. Ceci est la problématique traitée dans le chapitre 5.

❑

En résumé, la méthode des invariants permet le calcul du contrôleur optimal maximal permissif pour tout RdP ordinaire ou généralisé même en présence de la non-conservativité et des synchronisations incontrôlables, à condition que l'ensemble adéquat des contraintes lui est fourni. Cependant, trouver un tel ensemble de contraintes reste un problème très délicat, qui implique de prendre en compte les (éventuels) états interdits générés par les synchronisations incontrôlables et les couvertures entre les états dans le cas des systèmes non-conservatifs. Actuellement, il n'y a pas de solution unifiée pour cette question. Dans les chapitres suivants de ce mémoire, nous allons proposer des solutions à ce sujet, à partir du concept de contrainte généralisée d'exclusion mutuelle et de l'aspect spatial du problème.

3.4 Optimalité structurelle

La discussion concernant la synthèse des contrôleurs optimaux pour les SED comporte deux aspects : celui de l'optimalité «formelle» du contrôleur — c'est-à-dire sa capacité d'assurer au système en boucle fermée un comportement maximal permissif et en concordance avec le cahier des charges imposé — et celui de son optimalité «structurelle» — le nombre d'arcs et de places de contrôle qui doivent être ajoutés au modèle.

L'optimalité structurelle est une considération pratique de haute importance, étant donnée la grande sensibilité que les modèles SED montrent au problème de l'explosion combinatoire du nombre d'états. Pour ces raisons nous avons dédié une partie de notre recherche à ce sujet, même si la problématique de base pour cette thèse est celle de l'optimalité «formelle».

Formellement, pour assurer l'optimalité structurelle du contrôleur, il est nécessaire de garantir les deux conditions suivantes :

1. un nombre minimal de places de contrôle est ajouté au système, et
2. le nombre d'arcs de contrôle est, lui aussi, minimal.

En ce qui concerne l'opération de synthèse du contrôleur, ces exigences se traduisent par deux problèmes d'optimisation :

1. Le nombre de places de contrôle est intimement lié au *nombre des contraintes*. Il s'ensuit que, dans une première étape, l'optimalité structurelle implique la minimisation du nombre des contraintes. Ceci est un problème de calcul assez épineux, car l'efficacité des algorithmes est souvent peu satisfaisante. Notamment le nombre d'opérations qui reste substantiel, et ce malgré la complexité (généralement) polynomiale des algorithmes.
2. Dans une deuxième étape, il est désirable de minimiser le nombre d'arcs de contrôle. Ce nombre détermine la complexité des liens entre le contrôleur et le procédé, et il est associé à *la complexité de chaque contrainte*. Il faut donc minimiser le nombre des places marquées impliquées dans chaque contrainte.

Parmi les résultats présentés dans le 5ème chapitre de ce mémoire, nous allons proposer également un algorithme pour obtenir le contrôleur maximal permissif *minimal* en termes de nombre de places et d'arcs de contrôle. L'optimalité de la contrainte sera prise en compte dans la technique de détermination des contraintes admissibles.

3.5 Conclusions

Ce chapitre fait un compte rendu de la problématique de base étudiée dans cette thèse — la détermination des contraintes permettant la synthèse des contrôleurs optimaux dans les deux sens (maximal permissivité et structurel) pour une large classe de modèles RdP en utilisant la méthode des invariants. Il positionne notre travail par rapport aux résultats déjà existants et aux questions qui restent à résoudre. Notamment, avant d'identifier les questions essentielles auxquelles nous allons répondre dans notre recherche, nous avons justifié les motivations de notre décision d'une part, de «marier» la méthode des invariants avec la théorie de R&W et d'autre part, d'abandonner les contraintes de type GMEC non-pondéré. Nous avons également défini les notions fondamentales pour notre démarche et mis en évidence notre objectif secondaire — l'optimalité structurelle.

Chapitre 4

Détermination des contraintes en vue de la synthèse du contrôleur optimal pour les réseaux de Petri saufs

Ce chapitre présente une technique de détermination des contraintes permettant la synthèse, avec la méthode des invariants, de contrôleur maximal permissif pour les réseaux de Petri saufs non nécessairement conservatifs. L'approche est basée sur l'idée d'équivalence entre les contraintes et leurs états interdits correspondants.

4.1 Introduction

La clarté, la simplicité et l'efficacité de calcul du contrôleur ont rendu la méthode des invariants une des méthodes de synthèse les plus utilisées pour les SED modélisés par des réseaux de Petri. Malheureusement, cette approche ne garantit pas en général une solution optimale dans le cas où la synchronisation entre le procédé et la spécification est réalisée *via* des transitions incontrôlables. Pour le cas spécifique des RdP saufs et conservatifs, une solution pour palier cet inconvénient a été avancée dans [Dideban, 2007].

Dans ce chapitre nous proposons une solution au même problème pour le cas des RdP saufs non-conservatifs. L'idée centrale est celle de l'équivalence entre l'ensemble d'états interdits et celui des contraintes destinées à bloquer l'accès vers ces états. L'importance de cette équivalence est étudiée dans le contexte des modèles non-conservatifs. Notre

travail se situe dans l'étape de construction de contraintes admissibles pour la méthode des invariants, en amont des méthodes de simplification proposées dans [Dideban, 2007] (figure 4.1).

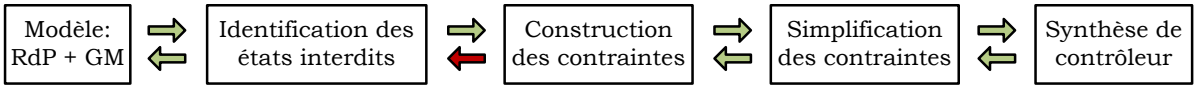


FIGURE 4.1 – Étapes de la procédure de synthèse de contrôleur, accentuant l'équivalence contrainte - état interdit associé.

4.2 L'importance de la conservativité

Dans ce chapitre seul le cas des RdP saufs est analysé. Un RdP est appelé sauf pour un marquage initial M_0 donné si toutes ses places sont 1-bornées (remarque 1.4.5). Cependant, il faut mentionner que même si le modèle du système à commander (procédé et spécification) est sauf, il est toutefois possible d'obtenir un modèle non-sauf pour le contrôleur.

Étant donné un RdP $\mathcal{R}$, un sous-ensemble $\mathcal{P}_\circlearrowleft \subset \mathcal{P}$ de ses places est appelé une composante conservative si la somme pondérée des jetons dans $\mathcal{P}_\circlearrowleft$ est constante. L'expression mathématique de cette relation constitue un invariant de marquage (définition 1.4.8). Le réseau lui-même est appelé conservatif si son ensemble complet des places, $\mathcal{P}$, est une composante conservative (remarque 1.4.6).

Considérons un système modélisé par un RdP sauf conservatif. La solution optimale de contrôle pour ce cas-là peut être déterminée avec les techniques présentées dans [Dideban, 2007]. Nous allons analyser dans la suite de cette section les implications de cette solution, ainsi que l'importance de la propriété de conservativité.

4.2.1 Synthèse de contrôleur pour les RdP saufs conservatifs

Nous allons commencer par présenter un exemple où le modèle est un RdP sauf conservatif et pour lequel nous allons calculer le contrôleur optimal. Cela va nous permettre d'introduire les questions posées dans le cas où le RdP est sauf mais non-conservatif.

Exemple 4.2.1
Prenons l'exemple classique d'un système à deux machines, $\mathscr{M}_1$ et $\mathscr{M}_2$, qui travaillent en tandem : chaque pièce doit être premièrement usinée par $\mathscr{M}_1$ et puis par $\mathscr{M}_2$, avec un stock de capacité unitaire entre les deux. La capacité

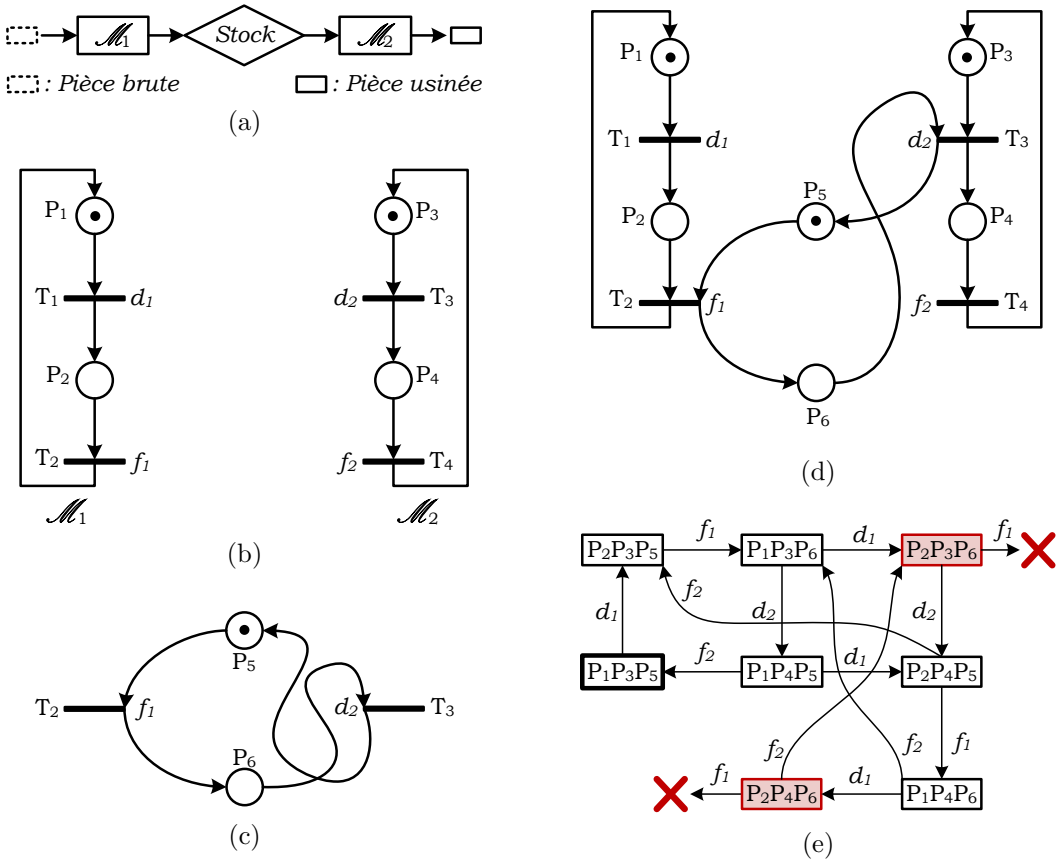


FIGURE 4.2 – Modèle du système manufacturier de l'exemple 4.2.1 : a) schéma du système manufacturier ; b) modèle RdP du procédé ; c) modèle RdP de la spécification ; d) modèle RdP du fonctionnement en boucle fermée ; e) graphe pertinent de marquage en boucle fermée.

d'usinage de chaque machine est d'une pièce à la fois. Seuls les débuts des tâches d'usinage sur chaque machine (événements d_i) sont contrôlables.

Le schéma du système manufacturier, ainsi que les modèles RdP de procédé, de la contrainte de stock (la spécification) et du fonctionnement désiré en boucle fermée sont présentés dans la figure 4.2), à côté du graphe pertinent de marquages accessibles en boucle fermée (figure 4.2e).

❑

À ce niveau il n'y a aucun moyen d'empêcher l'atteignabilité des états interdits générés par la synchronisation incontrôlable entre le procédé et la spécification. Un exemple d'un tel état est $P_2P_3P_6$. Il modélise un scénario de fonctionnement où la machine $\mathscr{M}_1$ a déjà commencé la fabrication d'une pièce, alors que $\mathscr{M}_2$ n'a pas encore fini le traitement de la pièce précédemment usinée par $\mathscr{M}_1$. Comme les fins d'opérations d'usinage sont incontrôlables (événements f_i), il s'ensuit qu'il est impossible de garantir que la pièce en

cours d'usinage sur $\mathscr{M}_1$ ne sera pas perdue.

Il faut donc appliquer un contrôleur au modèle en boucle fermée, afin de prévenir ce type de situations. La synthèse de la commande est réalisée avec la méthode des invariants :

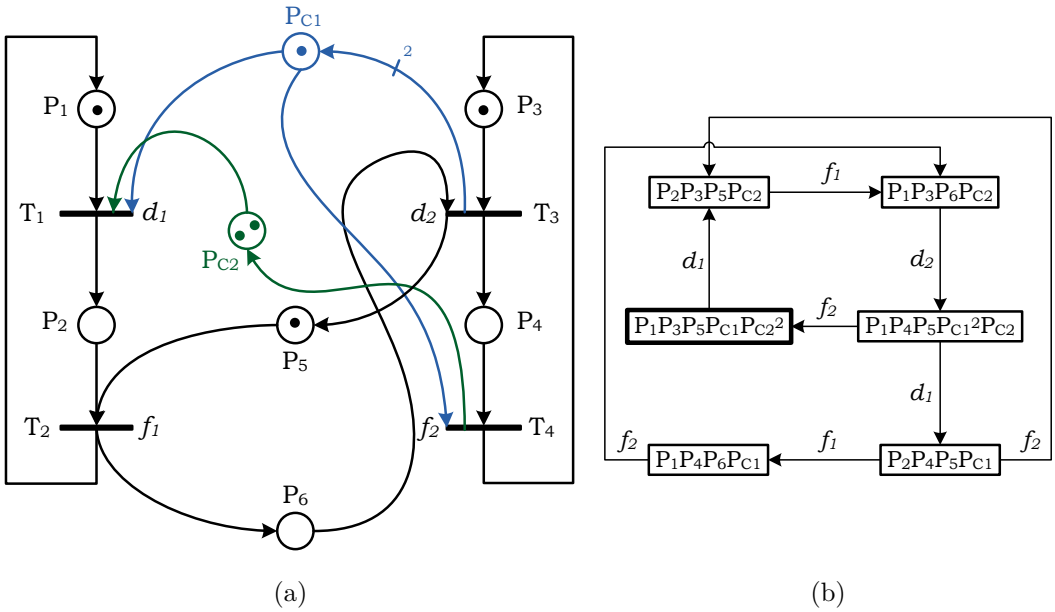
(a) Identification de l'ensemble d'états interdits frontière :
$\mathcal{M}_{\mathcal{B}} = \{P_2P_3P_6; P_2P_4P_6\}$

(b) Construction des contraintes :

$$\left.\begin{array}{rcl} M_{B_1} = P_2P_3P_6 & \Rightarrow & m_2 + m_3 + m_6 \leq 2 \\ M_{B_2} = P_2P_4P_6 & \Rightarrow & m_2 + m_4 + m_6 \leq 2 \end{array}\right| \Rightarrow L^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \; ; \; b_v = \begin{bmatrix} 2 \\ 2 \end{bmatrix}.$$

(c) Synthèse de contrôleur :

$$\mathcal{W}_{\mathscr{C}} = -L^T \cdot \mathcal{W}_{\mathscr{P}} = \begin{bmatrix} -1 & 0 & 2 & -1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad M_{\mathscr{C}_{ini}} = b_v - L^T \cdot M_{\mathscr{P}_{ini}} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$



(a) (b)

FIGURE 4.3 – Modèle contrôlé pour l'exemple 4.2.1 : a) modèle RdP du système contrôlé ; b) graphe de marquage.

Le contrôleur obtenu de cette manière est optimal, fait qui est reflété aussi par le graphe de marquage du système commandé (figure 4.3). L'optimalité du contrôleur repose sur la conservativité du modèle du système en boucle fermée. Celle-ci garantit qu'aucun marquage interdit ne couvre un marquage autorisé. Cependant, la question qu'il faut se poser est que se passerait-il quand l'hypothèse de conservativité est abandonnée ?

L'optimalité du contrôleur ainsi obtenu est-elle assurée également dans le cas où les modèles du procédé et / ou de la spécification sont non-conservatifs ? Malheureusement, la réponse à cette question est négative, comme nous allons le voir. Même s'il est possible de trouver des cas favorables de modèles non-conservatifs où il est possible de calculer un contrôleur maximal permissif par la méthode des invariants, dans le cas général l'opération de synthèse donne des résultats insatisfaisants.

4.2.2 Problèmes posés par l'hypothèse de non-conservativité

En général, étant donné un RdP sauf et conservatif $\mathcal{R}$ avec n places et un état interdit quelconque $M_{B_i} \in \mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$, $M_{B_i} = P_{i_1} P_{i_2} ... P_{i_t}$, $t \leq n$, la contrainte interdisant M_{B_i} est donnée par l'inégalité (3.3.1) :

$$c_i : \sum_{j=1}^{t} M(P_{i_j}) \leq b_i$$

où t est le nombre des places marquées dans l'état M_{B_i}, $M(P_{i_j}) = m_{i_j}$ est le marquage de la place P_{i_j}, et $b_i = t - 1$ est la borne de la contrainte.

Nous allons montrer que ce type de contraintes ne convient pas aux modèles non-conservatifs. Pour illustrer ce fait, nous allons utiliser une variante non-conservative de l'exemple 4.2.1. Nous choisissons donc une spécification qui impose que l'ensemble des deux machines peut usiner une et seulement une pièce. Cette variante de spécification est illustrée dans la figure 4.4a, et le modèle de fonctionnement désiré en boucle fermée correspondant est donné dans la figure 4.4b.

Après avoir construit le graphe pertinent de marquage pour ce système (figure 4.4c), nous pouvons identifier l'ensemble d'états interdits frontière :

$$\mathcal{M}_{\mathcal{B}} = \{P_2 P_3 P_6; P_2 P_4; P_2 P_3\}$$

Le graphe de marquage de la figure 4.4d correspond au contrôleur maximal permissif obtenu avec la théorie de R&W.

Il est maintenant possible de synthétiser le contrôleur (par la méthode des invariants) :

a) Dans une première étape, nous déterminons les contraintes associées à chaque état interdit et nous construisons la matrice de pondération et le vecteur des bornes des contraintes (L^T et respectivement b_v) :

$$\left.\begin{array}{lcl} M_{B_1} = P_2 P_3 P_6 & \Rightarrow & m_2 + m_3 + m_6 \leq 2 \\ M_{B_2} = P_2 P_4 & \Rightarrow & m_2 + m_4 \leq 1 \\ M_{B_3} = P_2 P_3 & \Rightarrow & m_2 + m_3 \leq 1 \end{array}\right| \Rightarrow L^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} ; \; b_v = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

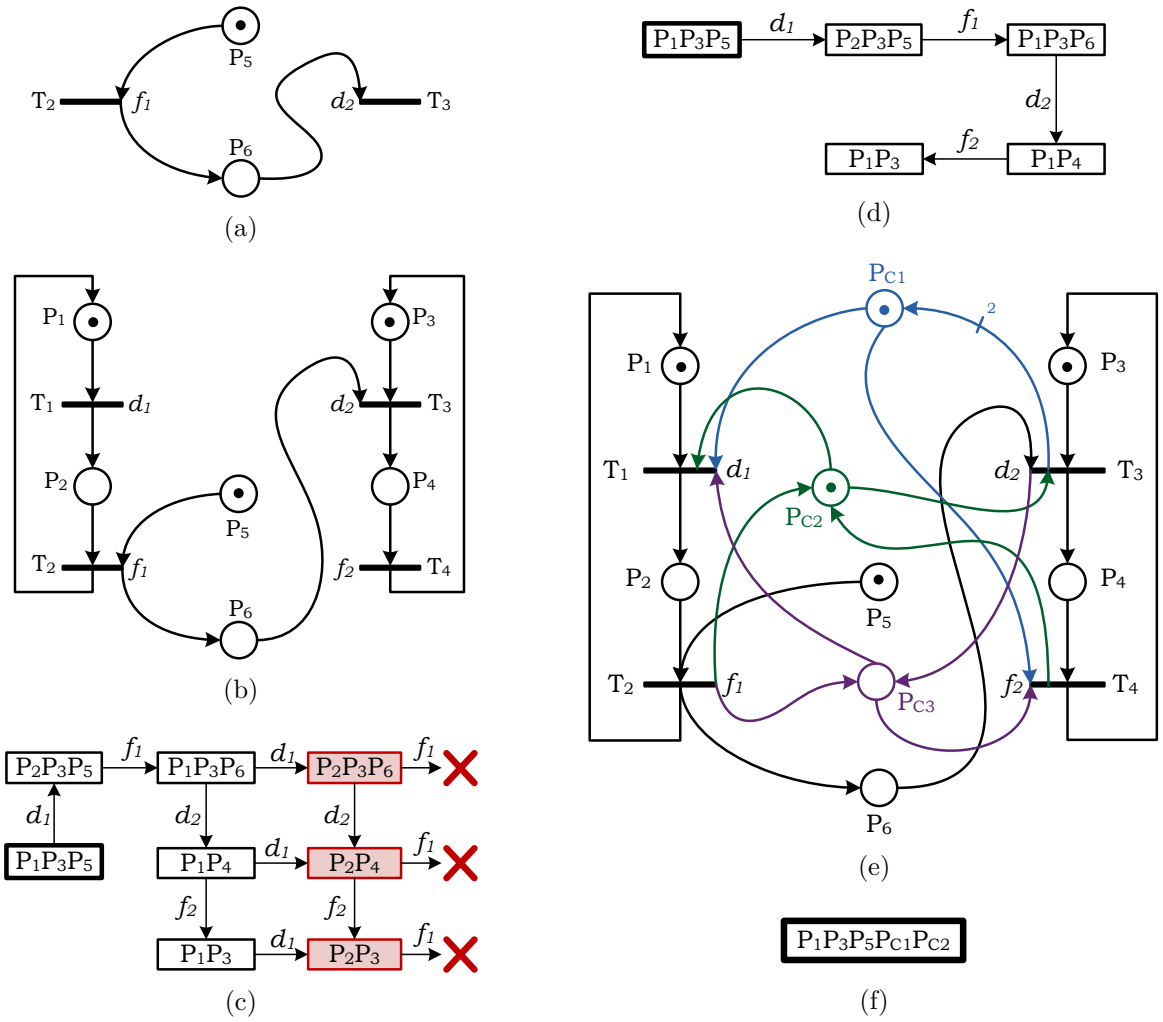


FIGURE 4.4 – Variante non-conservative du système manufacturier de l'exemple 4.2.1 : a) modèle RdP de la spécification non-conservative ; b) modèle RdP du fonctionnement en boucle fermée ; c) graphe pertinent de marquage en boucle fermée ; d) solution optimale de contrôle (R&W) ; e) modèle RdP du système contrôlé avec la méthode des invariants classique ; f) graphe de marquage du système contrôlé.

b) Ceci nous permet de calculer la matrice d'incidence et le marquage initial du contrôleur :

$$\mathcal{W}_{\mathscr{C}} = \begin{bmatrix} -1 & 0 & 2 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix} \quad ; \quad M_{\mathscr{C}_{ini}} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Le modèle RdP contrôlé est donné dans la figure 4.4e. C'est un RdP généralisé (remarque 2.4.2). Son graphe de marquage est donné dans la figure 4.4f. Il est réduit à un

seul état, ce n'est donc pas le contrôleur optimal. Cette restrictivité vient du fait que la contrainte associée à l'état interdit $M_{B_3} = P_2P_3$ bloque aussi l'accès vers l'état autorisé $P_2P_3P_5$.

4.3 Synthèse de contrôleur pour les RdP saufs non-conservatifs

Le problème du contrôle sous-optimal des modèles non-conservatifs ne réside pas dans la méthode de synthèse elle-même, mais dans le fait qu'un état interdit couvre un état autorisé (remarque 1.4.1). Les relations de couverture possibles entre les états d'un RdP non-conservatif rendent ce type de contraintes insuffisant pour caractériser de manière unique un état.

4.3.1 Redéfinition des contraintes

L'expression mathématique de la contrainte doit illustrer l'équivalence entre celle-ci et son état interdit associé. Retournant à l'exemple précédent, si nous considérons la contrainte associée à l'état interdit $M_{B_3} = P_2P_3 : m_2 + m_3 \leq 1$, la cause du problème devient évidente. Nous avons vu que cette contrainte interdit aussi l'état autorisé $P_2P_3P_5$, qui est couvert par M_{B_3}. Il s'ensuit que l'utilisation de la fonction support d'un état pour la détermination de sa contrainte équivalente n'est pas une pratique recommandée dans le cas des modèles non-saufs. Il faut donc utiliser une définition complète de l'état — celle de vecteur de marquage. Tout état interdit $M_{B_i} = P_{i_1}P_{i_2}...P_{i_t}$ est un vecteur de marquage :

$$M_{B_i} = P_{i_1}P_{i_2}...P_{i_t} = [\ \underbrace{0}_{P_1} \cdots 0\ \underbrace{1}_{P_{i_1}}\ 0 \cdots 0\ \underbrace{1}_{P_{i_2}}\ 0 \cdots 0\ \underbrace{1}_{P_{i_t}}\ 0 \cdots 0\ \underbrace{}_{P_{i_n}}\ ]^T ,$$

$P_1 \updownarrow$ noté $P_{i_{t+1}}$

où t est le nombre de places marquées de l'état M_{B_i} et n est le nombre total de places du réseau. Nous avons noté les marquages correspondants aux t places marquées avec un jeton, en ordre, de m_{i_1} à m_{i_t}, alors que les marquages correspondant aux $n-t$ places contenant zéro jetons ont été notées (aussi en ordre) de $m_{i_{t+1}}$ à m_{i_n}.

Partant du vecteur de marquage, la forme la plus générale de la contrainte doit alors contenir toutes les variables de marquage des places. La contrainte associée à un état M_{B_i} donné devient ainsi :

$$m_{i_1} + m_{i_2} + ... + m_{i_t} + \overline{m_{i_{t+1}}} + \overline{m_{i_n}} \leq n - 1 ,$$

où $\overline{m_j}$ représente le complément logique du marquage m_j. Comme tout vecteur de marquage est unique, une telle contrainte est évidemment équivalente à un et seulement un état. Pour notre exemple, la contrainte associée à l'état $M_{B_3} = P_2P_3$ devient : $\overline{m_1} + m_2 + m_3 + \overline{m_4} + \overline{m_5} + \overline{m_6} \leq 5$.

Soient $\mathcal{M}_\mathcal{A}$ et $\mathcal{M}_\mathcal{F}$ les ensembles d'états autorisés et respectivement interdits d'un RdP $\mathcal{R}$ donné.

Définition 4.3.1.

1. *À tout état interdit M_{B_i} est associée une contrainte c_i. Soit $\mathcal{C}_\mathcal{F}$ l'ensemble de ces contraintes.*
2. *$\mathcal{M}_{\mathcal{C}_\mathcal{F}} \subset \{0,1\}^n$ représente l'ensemble des vecteurs booléens de dimension n qui vérifient les contraintes $\mathcal{C}_\mathcal{F}$.*

❑

La propriété suivante peut être énoncée :

Propriété 4.3.1. *L'espace d'états défini par $\mathcal{M}_{\mathcal{C}_\mathcal{F}}$ est égal à l'ensemble d'états autorisés, $\mathcal{M}_\mathcal{A}$:*

$$\mathcal{M}_{\mathcal{C}_\mathcal{F}} = \mathcal{M}_\mathcal{A}.$$

❑

Preuve 4.3.1 :

L'implication directe, $\mathcal{M}_\mathcal{A} \subset \mathcal{M}_{\mathcal{C}_\mathcal{F}}$, est rendue évidente par la construction de $\mathcal{M}_{\mathcal{C}_\mathcal{F}}$.

Pour prouver que $\mathcal{M}_{\mathcal{C}_\mathcal{F}} \subset \mathcal{M}_\mathcal{A}$, supposons que $\mathcal{M}_{\mathcal{C}_\mathcal{F}} \not\subset \mathcal{M}_\mathcal{A}$. Il y a alors au moins un état $M_i \in \mathcal{M}_{\mathcal{C}_\mathcal{F}}$ tel que $\mathcal{M}_i \notin \mathcal{M}_\mathcal{A}$; c'est-à-dire que $M_i \in \mathcal{M}_\mathcal{F}$. Cependant, $M_i \in \mathcal{M}_\mathcal{F}$ implique que $\sum_{j=1}^{t} m_{i_j} + \sum_{j=t+1}^{n} \overline{m_{i_j}} > n-1$, ce qui contredit l'hypothèse $M_i \subset \mathcal{M}_{\mathcal{C}_\mathcal{F}}$.

❑

Remarque 4.3.1. La propriété 4.3.1 garantit l'optimalité du contrôleur.

❑

Revenant à l'exemple de la figure 4.4b, l'ensemble final des contraintes est le suivant :

$$\begin{aligned}
M_{B_1} = P_2P_3P_6 &\Rightarrow \overline{m_1} + m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 \leq 5 \\
M_{B_2} = P_2P_4 &\Rightarrow \overline{m_1} + m_2 + \overline{m_3} + m_4 + \overline{m_5} + \overline{m_6} \leq 5 \\
M_{B_3} = P_2P_3 &\Rightarrow \overline{m_1} + m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 \leq 5
\end{aligned}$$

À ce niveau, nous venons de déterminer un ensemble de contraintes définissant de manière bijective l'espace d'états autorisés du système. Cependant, ces contraintes contiennent des marquages complémentés et donc elles ne sont pas compatibles avec la méthode de synthèse du contrôleur basée sur les invariants. Il faut maintenant trouver un moyen d'éliminer les compléments. L'idée consiste d'abord à rendre le modèle conservatif. Grâce aux invariants on pourra éliminer les compléments et on verra qu'il sera toujours possible de revenir au RdP de départ. C'est l'objet de la suite de ce chapitre.

4.3.2 Passage d'un modèle non-conservatif à un modèle conservatif

Conformément à la théorie générale des réseaux de Petri, tout RdP borné non-conservatif peut être rendu conservatif par l'addition de places complémentaires à ses places non-conservatives. Tout RdP sauf est (par définition) borné par 1, et peut donc être rendu conservatif (figure 4.5).

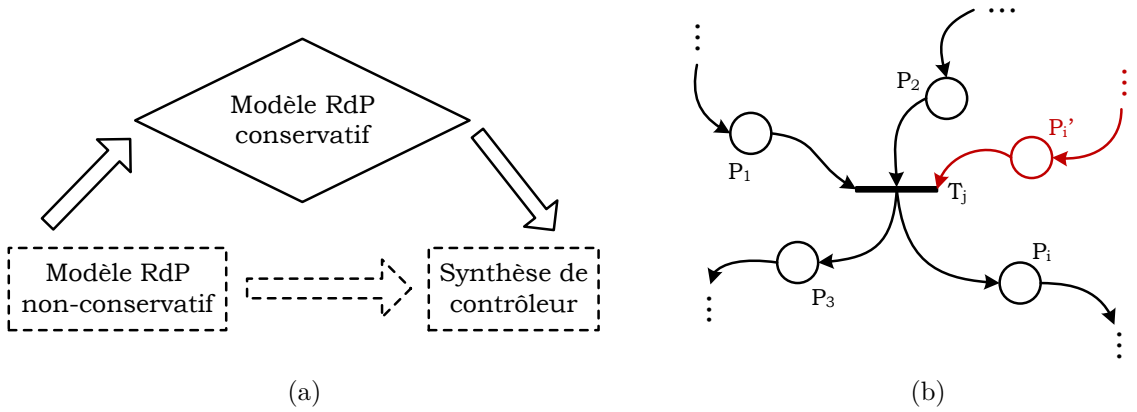


FIGURE 4.5 – Passage non-conservatif – conservatif : a) schéma général ; c) illustration sur un modèle RdP.

L'idée de la procédure est similaire à celle utilisée pour la transformation d'un RdP à capacités ou celle d'un RdP à arcs inhibiteurs borné en un RdP ordinaire. L'opération est redondante et n'a aucun effet sur la dynamique du système. Elle constitue une étape intermédiaire de la synthèse de contrôleur et sera inversée une fois la commande calculée. De plus, la transformation est systématique et peut être réalisée à partir de la matrice d'incidence du modèle.

Étant donné un RdP $\mathcal{R}$ avec une place non-conservative P_i (elle n'appartient à aucune composante conservative), la procédure de construction de la place complémentaire P_i' consiste en trois étapes :

1. une nouvelle place, $P_i^{'}$, est ajoutée au modèle de telle façon qu'elle constitue un reflet inversé en miroir de P_i (*i.e.* tout arc entrant dans- / sortant de- P_i aura un arc correspondant sortant de- / entrant dans- $P_i^{'}$) ;
2. le marquage de $P_i^{'}$ est tel que $M(P_i^{'}) = \overline{M(P_i)}$ ($M_0(P_i^{'}) = \overline{M_0(P_i)}$, et nous avons l'invariant $M(P_i^{'}) + M(P_i) = 1$).

L'étape (1.) consiste à ajouter une nouvelle ligne correspondant à $P_i^{'}$ à la matrice d'incidence du système. Cette ligne reflète la configuration d'arcs entrant dans- et, respectivement, partant de- la place $P_i^{'}$, et est construite comme l'opposé de la ligne correspondante à P_i.

La construction formelle de la matrice d'incidence augmentée consiste à identifier les lignes associées aux composantes non-conservatives, et ajouter à chacune de ces lignes son opposé :

$$\mathcal{W}_{\mathscr{P}}^{cons} = \begin{bmatrix} \dots & \dots & \dots & \dots \\ 0 & 1 & \dots & -1 \\ \dots & \dots & \dots & \dots \\ 0 & -1 & \dots & 1 \\ \dots & \dots & \dots & \dots \end{bmatrix} \begin{matrix} \\ P_i \\ \dots \\ P_i^{'} \\ \\ \end{matrix} \quad ; \quad M_{\mathscr{P}_{ini}}^{cons} = \begin{bmatrix} \dots \\ 1 \\ \dots \\ 0 \\ \dots \end{bmatrix} \begin{matrix} \\ P_i \\ \dots \\ P_i^{'} \\ \\ \end{matrix}$$

où $\mathcal{W}_{\mathscr{P}}^{cons}$ et $M_{\mathscr{P}_{ini}}^{cons}$ désignent la matrice d'incidence et, respectivement, le marquage initial du modèle conservatif obtenu de cette manière.

Propriété 4.3.2. *La place $P_i^{'}$ est une place implicite.*

❑

Preuve 4.3.2 :

Considérons le modèle général de la figure 4.5b. La place $P_i^{'}$ satisfait les deux conditions caractéristiques des places implicites :

- le marquage de $P_i^{'}$ peut être déduit à partir des marquages des autres places du réseau :
$$M(P_i^{'}) = \overline{M(P_i)}.$$
- le marquage de la place $P_i^{'}$ ne constitue jamais un empêchement pour le franchissement de ses transitions de sortie :
$$\text{Si } T_j \text{ franchissable } \xrightarrow{RdP\ sauf} M(P_1) = 1 \Rightarrow M(P_i) = 0 \Rightarrow M(P_i^{'}) = 1.$$

❑

Le graphe de marquages accessibles d'un modèle rendu conservatif est isomorphe à celui du modèle original (non-conservatif). Les deux modèles sont équivalents du point de vue du comportement dynamique.

Soit $\mathcal{R}_1$ un RdP non-conservatif, $\mathcal{R}_2$ son RdP conservatif équivalent obtenu par la technique présentée ci-dessus et soient $\mathcal{P}_1$ et respectivement $\mathcal{P}_2$ leurs ensembles des places, avec $\mathcal{P}_1 \subset \mathcal{P}_2$. Le corollaire suivant peut être fait concernant les deux modèles :

Corollaire 4.3.3. *Les RdP $\mathcal{R}_1$ et $\mathcal{R}_2$ sont équivalents, dans le sens où leurs graphes de marquage sont isomorphes :*

$$\exists\ f \text{ bijective telle que } \forall M \in \mathcal{G}_1, \exists\ M^{'} \in \mathcal{G}_2 \text{ avec } f(M) = M^{'}.$$

❑

Preuve 4.3.3 :

Nous allons présenter l'idée de base de la preuve de ce corollaire en partant du cas d'un RdP avec une seule place non-conservative (figure 4.6). La preuve est généralisable par extension.

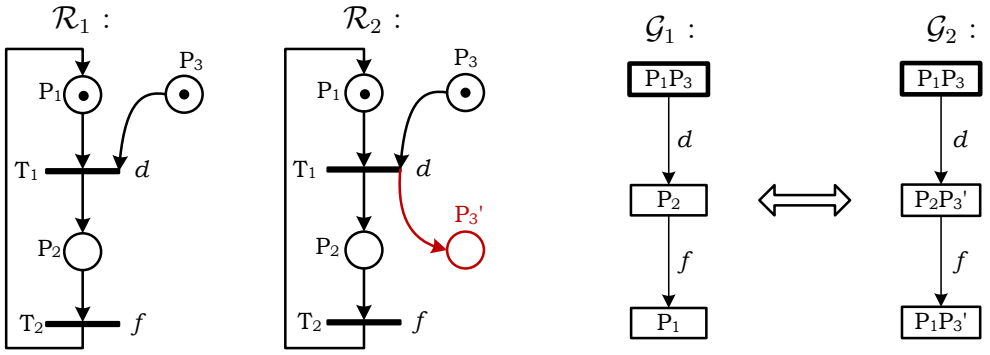


FIGURE 4.6 – Un RdP non-conservatif et son RdP conservatif équivalent : a) modèles RdP ; b) graphes de marquage.

Considérons un état $M \in \mathcal{G}_1$. Deux cas se présentent :

1. soit $P_3 \in Support(M) \Rightarrow M \in \mathcal{G}_2$, car $M(P_3^{'}) = 0$,
2. soit $P_3 \notin Support(M) \Rightarrow M^{'} = M \cup P_3^{'} \in \mathcal{G}_2$.

De manière symétrique, considérons un état $M^{'} \in \mathcal{G}_2$. Nous avons alors :

1. soit $P_3 \in Support(M^{'}) \Rightarrow M^{'} \in \mathcal{G}_1$,
2. soit $P_3^{'} \in Support(M^{'}) \Rightarrow M = M^{'} \backslash P_3^{'} \in \mathcal{G}_1$.

❑

4.3.3 Élimination des marquages complémentés

Il est maintenant possible d'éliminer les marquages complémentés de la structure de nos contraintes, afin de pouvoir les utiliser avec la méthode des invariants. Chaque

marquage complémenté peut être compensé grâce à l'invariant $m_i + \overline{m_i} = 1$. Pour les RdP conservatifs cette compensation est réalisable *via* des opérations élémentaires impliquant les invariants de marquage. Dans le cas des modèles non-conservatifs, les opérations peuvent être réalisées sur leurs équivalents conservatifs. En vertu du corolaire 4.3.3 et de la propriété 4.3.2, le contrôle calculé pour un modèle conservatif est valable aussi pour son homologue non-conservatif, et *vice-versa*.

Il s'agit en premier lieu de calculer l'ensemble des invariants minimaux du réseau : $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, ...\mathcal{I}_r\}$.

Soit (4.3.1) la forme générale de l'invariant de marquage :

$$\mathcal{I}_l : \sum_{j=1}^{h} m_{i_j}^l + \sum_{j=h+1}^{T_l} m_{i_j}^l = 1 \tag{4.3.1}$$

où la première somme correspond aux marquages complémentés dans la contrainte.

En deuxième lieu, c'est le partitionnement de la contrainte dans des sous-ensembles disjoints des marquages qui est réalisé. Ces sous-ensembles sont construits de la manière suivante :

1. on regroupe tous les marquages non-complémentés ;
2. on regroupe le maximum des marquages complémentés appartenant au même invariant ;
3. on regroupe tous les marquages complémentés qui ne sont inclus dans aucun invariant.

Les points (1.) et (3.) donnent chacun un sous-ensemble. Par contre, le point (2.) donnera dans le cas général plusieurs sous-ensembles.

La forme générale de la contrainte est comme décrit ci-dessous :

$$\begin{aligned} c_i : \overbrace{\overline{m_{i_1} + m_{i_2} + ... + m_{i_t}}}^{\text{Marquages non-complémentés}} + \overbrace{\sum_{j=t+1}^{J_1} \overline{m_{i_j}^1}}^{\text{Marquages complémentés inclus dans l'invariant } \mathcal{I}_1} + ... + \overbrace{\sum_{j=J_{l-1}+1}^{J_l} \overline{m_{i_j}^l}}^{\text{Marquages complémentés inclus dans l'invariant } \mathcal{I}_l} + ... \\ + \underbrace{\sum_{j=J_{r-1}+1}^{J_r} \overline{m_{i_j}^r}}_{\text{Marquages complémentés inclus dans l'invariant } \mathcal{I}_r} + \underbrace{\overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + ... + \overline{m_{i_\xi}}}_{\text{Marquages complémentés inclus dans aucun invariant}} \leq n - 1 \end{aligned} \tag{4.3.2}$$

Pour établir notre théorème, nous allons considérer deux sous-ensembles correspondants au point (2.)) (invariants $\mathcal{I}_1$ et $\mathcal{I}_2$) et trois marquages complémentés correspondants

au point (3.)). Ceci ne diminue en rien la généralité de notre démonstration :

$$c_i : \quad m_{i_1} + m_{i_2} + ... + m_{i_t} + \underbrace{\sum_{j=t+1}^{J_1} \overline{m_{i_j}^1}}_{I_1} + \underbrace{\sum_{j=J_1+1}^{J_2} \overline{m_{i_j}^2}}_{I_2} + \overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + \overline{m_{i_\gamma}} \leq n-1 \quad (4.3.3)$$

Théorème 4.3.4. *Pour toute contrainte de la forme* (4.3.3)*, il existe une contrainte équivalente qui ne contient aucun marquage complementé :*

$$\sum_{j=1}^{t} m_{i_j} + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 + m_{i_\alpha}^{'} + m_{i_\beta}^{'} + m_{i_\gamma}^{'} \leq n + t + 1 - J_2 \quad (4.3.4)$$

où $J_2 - t - 1$ est le nombre des marquages complémentés dans l'inégalité (4.3.3)*, par rapport aux invariants minimaux $\mathcal{I}_1$ et $\mathcal{I}_2$.*

❑

Preuve 4.3.4 :

Soit les invariants de marquage $\mathcal{I}_1$ et $\mathcal{I}_2$ ci-dessous :

$$\mathcal{I}_1 : \quad \sum_{j=t+1}^{J_1} m_{i_j}^1 + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 = 1$$

$$\mathcal{I}_2 : \quad \sum_{j=J_1+1}^{J_2} m_{i_j}^2 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 = 1$$

Nous pouvons utiliser les invariants pour apporter des transformations dans l'inégalité décrivant la contrainte. Considérons, pour commencer, l'invariant $\mathcal{I}_1$. Une simple opération d'addition entre l'invariant et la contrainte nous permettrait de compenser le premier sous-ensemble des marquages complémentés :

$$c_i : \quad \sum_{j=1}^{t} m_{i_j} + \underbrace{\sum_{j=t+1}^{J_1} \overline{m_{i_j}^1}}_{I_1} + \underbrace{\sum_{j=J_1+1}^{J_2} \overline{m_{i_j}^2}}_{I_2} + \overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + \overline{m_{i_\gamma}} \leq n-1$$

$$\mathcal{I}_1 : \quad \sum_{j=t+1}^{J_1} m_{i_j}^1 + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 = 1$$

$$\oplus : \quad \sum_{j=1}^{t} m_{i_j} + (J_1 - t) + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \underbrace{\sum_{j=J_1+1}^{J_2} \overline{m_{i_j}^2}}_{I_2} + \overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + \overline{m_{i_\gamma}} \leq n$$

De manière analogue, nous compensons des sous-ensembles des marquages complémentés jusqu'à l'épuisement de tous les invariants. Il reste ainsi à compenser seul le sous-ensemble des marquages complémentés n'appartenant à aucun invariant de marquage.

$$\sum_{j=1}^{t} m_{i_j} + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 + \overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + \overline{m_{i_\gamma}} \leq n + t + 1 - J_2$$

Les marquages complémentés restants peuvent être éliminés en rendant les places concernées conservatives. L'idée de base est d'inclure ces marquages dans un nouveau ensemble d'invariants. Comme les marquages complémentés restants correspondent aux places non-conservatives du réseau, l'addition de leur complémentaires rend le modèle conservatif.

$$\sum_{j=1}^{t} m_{i_j} + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 + \overline{m_{i_\alpha}} + \overline{m_{i_\beta}} + \overline{m_{i_\gamma}} \leq n + t + 1 - J_2$$

$$m_{i_\alpha} + m_{i'_\alpha} = 1$$

$$m_{i_\beta} + m_{i'_\beta} = 1$$

$$m_{i_\gamma} + m_{i'_\gamma} = 1$$

$$\oplus : \sum_{j=1}^{t} m_{i_j} + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 + 1 + m'_{i_\alpha} + 1 + m'_{i_\beta} + 1 + m'_{i_\gamma} \leq n + t + 4 - J_2$$

$$\Updownarrow$$

$$\sum_{j=1}^{t} m_{i_j} + \sum_{j=J_1+1}^{T_1} m_{i_j}^1 + \sum_{j=J_2+1}^{T_2} m_{i_j}^2 + m'_{i_\alpha} + m'_{i_\beta} + m'_{i_\gamma} \leq n + t + 1 - J_2$$

❑

4.3.4 Exemple

Dans cette section nous allons déterminer dans une première étape l'ensemble des contraintes ne contenant aucune place complémentée correspondant à l'exemple de la figure 4.4. Ces contraintes constituent la base pour le calcul des places de contrôle (en utilisant la méthode des invariant). Dans un deuxième temps nous allons calculer le contrôleur optimal pour cet exemple.

Construction du modèle conservatif :

Il faut, d'abord, écrire les contraintes en identifiant le sous-ensemble des places qui ne sont incluses dans aucun invariant de marquage. La construction du modèle conservatif est ensuite réalisée en appliquant le corollaire 4.3.3.

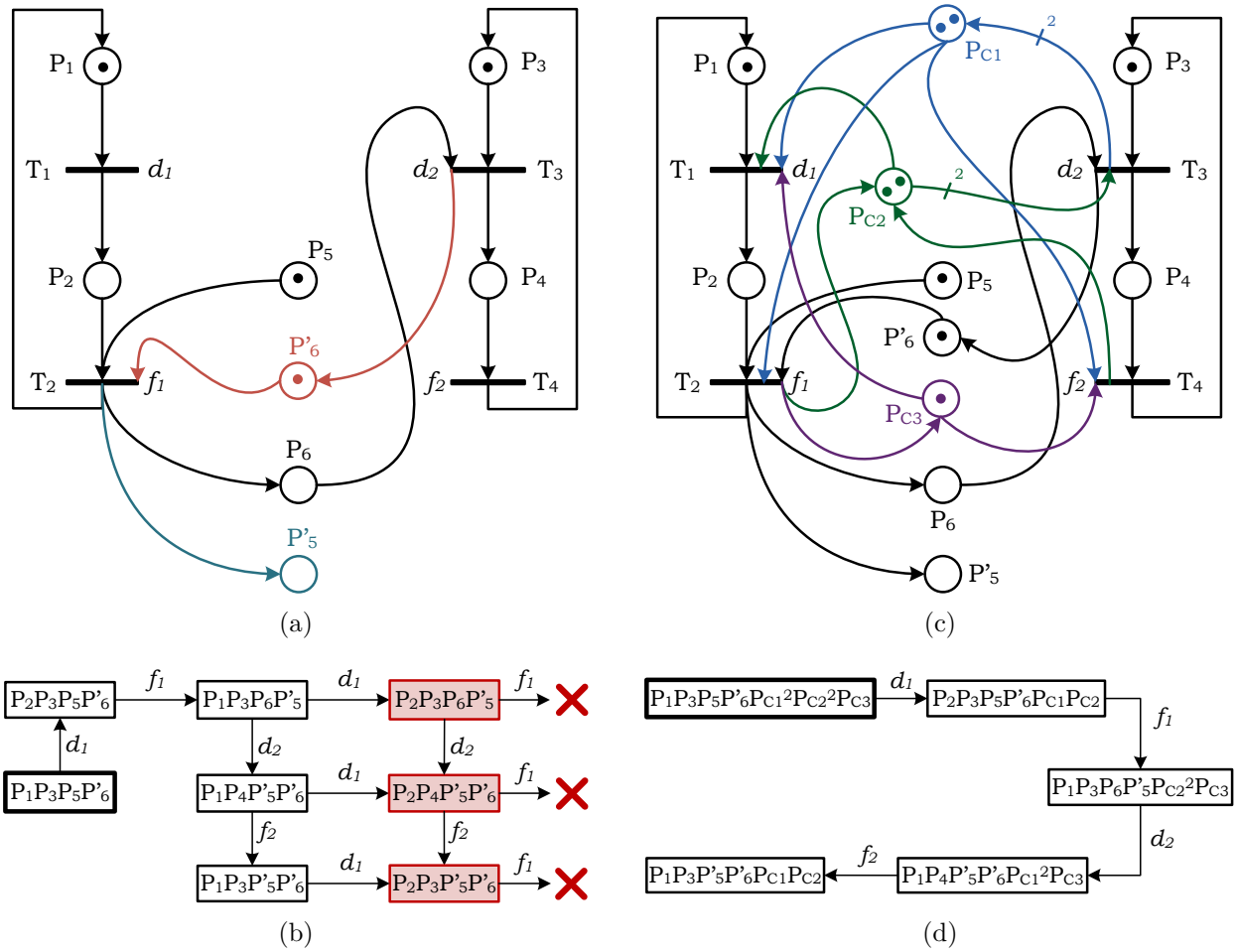


FIGURE 4.7 – Modèle conservatif et implémentation du contrôle pour le système de la figure 4.4 : a) modèle RdP conservatif ; b) graphe pertinent de marquage ; c) modèle RdP du système conservatif contrôlé ; d) graphe de marquage du système conservatif contrôlé.

Nous commençons par la liste des invariants minimaux de marquage :

$$m_1 + m_2 = 1$$

$$m_3 + m_4 = 1$$

Notre exemple contient deux places non-conservatives : P_5 et P_6. Il faut donc ajouter deux places au modèle : $P_5^{'}$ et $P_6^{'}$ (figure 4.7a). Nous obtenons ainsi deux invariants supplémentaires :

$$m_5 + m_5^{'} = 1$$

$$m_6 + m_6^{'} = 1$$

La transformation est reflétée par la matrice d'incidence du système :

$$\mathcal{W}_{\mathscr{P}}^{cons} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_5' \\ P_6' \end{matrix} \quad ; \quad M_{\mathscr{P}_{ini}}^{cons} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_5' \\ P_6' \end{matrix}$$

Le modèle décrit dans la figure 4.7a permet la détermination d'un ensemble de contraintes admissibles pour la synthèse du contrôleur. Le graphe de marquage (figure 4.7b) identifie l'ensemble suivant d'états interdits frontière :

$$\mathcal{M}_{\mathcal{B}} = \left\{ P_2P_3P_6P_5'; P_2P_4P_5'P_6'; P_2P_3P_5'P_6' \right\}.$$

Élimination des marquages complémentés :

Considérons maintenant la première contrainte, associée au marquage interdit $P_2P_3P_6P_5'$:

$$\overline{m_1} + m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 \leq 5.$$

Nous pouvons éliminer tous les marquages complémentés en appliquant la technique présentée dans le théorème 4.3.4 :

$$\begin{aligned} \overline{m_1} + m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 &\leq 5 \\ m_1 + m_2 &= 1 \\ \hline \oplus : \quad (m_1 + \overline{m_1}) + 2m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 &\leq 6 \\ \Updownarrow \; m_1 + \overline{m_1} &= 1 \\ 2m_2 + m_3 + \overline{m_4} + \overline{m_5} + m_6 &\leq 5 \\ m_3 + m_4 &= 1 \\ \hline \end{aligned}$$

$$\begin{aligned}
\oplus: \quad 2m_2 + 2m_3 + \overline{m_5} + m_6 &\leq 5 \\
m_5 + m_5^{'} &= 1 \\
\hline
\oplus: \quad 2m_2 + 2m_3 + m_6 + m_5^{'} &\leq 5 \\
m_6 &\leq 1 \\
m_5^{'} &\leq 1 \\
\hline
\oplus: \quad 2m_2 + 2m_3 + 2m_6 + 2m_5^{'} &\leq 7 \quad | : 2 \\
\Downarrow & \\
m_2 + m_3 + m_6 + m_5^{'} &\leq 3
\end{aligned}$$

En utilisant la même technique on trouve l'ensemble des contraintes pour tous les états interdits frontière de l'exemple :

$$\begin{aligned}
M_{B_1} = P_2P_3P_6P_5^{'} &\Rightarrow m_2 + m_3 + m_6 + m_5^{'} \leq 3 \\
M_{B_2} = P_2P_4P_5^{'}P_6^{'} &\Rightarrow m_2 + m_4 + m_5^{'} + m_6^{'} \leq 3 \\
M_{B_3} = P_2P_3P_5^{'}P_6^{'} &\Rightarrow m_2 + m_3 + m_5^{'} + m_6^{'} \leq 3
\end{aligned}$$

Synthèse de la commande :

Le contrôleur est calculé avec la méthode des invariants :

$$\mathcal{W}_{\mathscr{C}} = -L^T \cdot \mathcal{W}_{\mathscr{P}}^{cons} = \begin{bmatrix} -1 & -1 & 2 & -1 \\ -1 & 1 & -2 & 1 \\ -1 & 1 & 0 & -1 \end{bmatrix}$$

$$M_{\mathscr{C}_{ini}} = b_v - L^T \cdot M_{\mathscr{P}_{ini}}^{cons} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

Le modèle commandé et son graphe de marquage sont présentés, respectivement, dans les figures 4.7c et 4.7d. Le contrôleur est optimal.

Nous avons montré que le passage d'un modèle non-conservatif vers un modèle conservatif correspond à une transformation isomorphe nécessaire pour le calcul des contraintes équivalentes. C'est pour cela que c'est le modèle le plus simple (suppression des places ajoutées) qui sera retenu pour le modèle du contrôleur optimal final (figure 4.8).

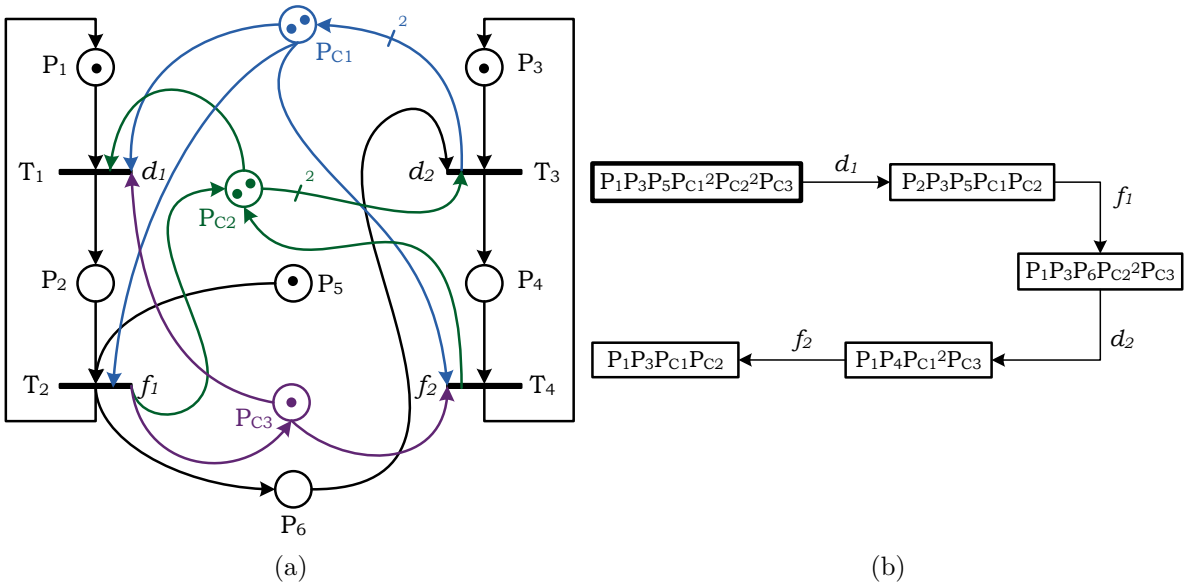


FIGURE 4.8 – Modèle contrôlé pour le système de la figure 4.4 : a) modèle RdP non-conservatif ; b) graphe de marquage.

4.4 Conclusions

Dans ce chapitre nous avons présenté une solution pour la détermination des contraintes équivalentes aux états interdits dans le contexte des modèles RdP saufs et non-conservatifs. Cette équivalence repose sur une première définition des contraintes contenant des marquages complémentés. Pour pouvoir appliquer la méthode des invariants il a été nécessaire d'éliminer les marquages complémentés. Ceci est réalisé en exploitant les composantes conservatives des RdP. Lorsqu'il existe des places non-conservatives, nous avons montré qu'il est possible de construire un RdP conservatif isomorphe. La procédure est réalisée en ajoutant des places implicites au modèle. Celles-ci sont supprimées dans le modèle du contrôleur final.

Nous avons mené à bien toute la synthèse optimale du contrôle dans le cas des RdP saufs ([Vasiliu & Alla, 2010], [Vasiliu & Alla, 2011*b*]). Celle-ci a été grandement facilitée par le fait que le marquage des places est binaire. Nous avons alors prouvé formellement l'équivalence du passage depuis un état interdit vers une contrainte linéaire. Il était alors naturel de voir si ces résultats peuvent être généralisés à un RdP ordinaire, où le marquage de chaque place est un entier quelconque. Ceci fera l'objet du prochain chapitre.

Chapitre 5

Détermination des contraintes en vue de la synthèse du contrôleur optimal pour les réseaux de Petri ordinaires et généralisés

Ce chapitre présente une technique de détermination des contraintes permettant la synthèse, avec la méthode des invariants, de contrôleur maximal permissif pour les réseaux de Petri généralisés. L'approche est basée sur l'idée d'équivalence entre les contraintes et leurs états interdits correspondants. La forme générale de la contrainte est déterminée à partir d'une analyse spatiale de l'espace d'états du système. Une solution est proposée aussi pour le problème de l'optimalité structurelle du contrôleur.

5.1 Introduction

Nous venons de présenter, dans le chapitre précédent, une méthode de détermination des contraintes permettant la synthèse (avec la méthode des invariants) d'un contrôleur maximal permissif pour tout RdP sauf. Malheureusement, cette approche est basée sur l'idée de complément binaire et a donc le grand inconvénient d'être spécifique aux RdP binaires. Dans ce chapitre nous allons proposer une démarche pour formuler des contraintes admissibles caractérisant le contrôleur optimal minimal pour les RdP généralisés.

Notre approche marie la théorie de R&W et la méthode des invariants afin de résoudre le problème des états interdits générés par les synchronisations incontrôlables entre le modèle du procédé et celui de la spécification. L'objectif est de développer une technique de détermination des contraintes admissibles pour la méthode des invariants qui fournira toujours un ensemble des contraintes caractérisant le contrôleur optimal, si un tel contrôleur existe. La méthode repose sur la séparabilité des ensembles d'états autorisés et interdits qui composent l'espace d'états accessibles d'un RdP.

Étant donné le modèle RdP d'un système en boucle fermée, nous utilisons dans un premier temps l'approche de R&W pour déterminer les deux ensembles nécessaires et suffisants pour le calcul du contrôleur optimal : le comportement autorisé, $\mathcal{M}_{\mathcal{A}}$, et l'ensemble des états interdits frontière, $\mathcal{M}_{\mathcal{B}}$. À partir de ce point, nous déterminons un ensemble des contraintes caractérisant de manière bijective le comportement autorisé. La forme générale de nos contraintes est formulée à partir d'une analyse spatiale de l'espace d'états du système. Une méthode d'optimisation est utilisée pour assurer la complexité minimale de chaque contrainte. Nous proposons aussi un algorithme de minimisation du nombre des contraintes, afin d'assurer l'optimalité structurelle de la solution de contrôle obtenue. Finalement, nous réalisons un étude de cas d'un système manufacturier complexe et implémentons notre démarche pour calculer le contrôleur optimal.

5.2 Contraintes et optimalité dans le contexte des RdP généralisés

Pour que le contrôleur d'un système en boucle fermée donné soit optimal, il doit satisfaire les deux conditions suivantes, données ici de manière intuitive : 1) il interdit tout comportement contraire à la spécification, et 2) il n'influence aucun autre comportement du système. Dans le cas des RdP, respecter ces exigences revient à assurer une équivalence entre les états interdits frontières et leurs contraintes associées. Si plusieurs états interdits sont exclues par la même contrainte, ils appartiennent à la même classe d'équivalence modulo la contrainte linéaire commune.

En bref, le problème n'est pas juste d'interdire des états (fait facilement réalisable), mais d'interdire *seulement* les états qui ne respectent pas les spécifications et en laissant un degré maximal de liberté au système. Cette section analyse l'importance de la propriété de permissivité maximale du contrôleur dans le contexte des RdP généralisés.

Un RdP généralisé, $\mathcal{R}$, est un RdP autonome dans lequel des poids (des nombres entiers strictement positifs) sont associés aux arcs (définition 1.4.4).

Supposant que $\mathcal{R}$ modélise le comportement d'un SED en boucle fermée ayant l'état initial M_0, son espace d'états accessibles $\mathcal{M}_{Acs}^{\langle \mathcal{R}, M_0 \rangle}$ peut être vu comme l'union de deux

ensembles d'états : l'ensemble $\mathcal{M}_{\mathcal{A}}$ d'états possibles dans le procédé à commander et autorisés par la spécification de contrôle (définition 3.2.3) et l'ensemble $\mathcal{M}_{\mathcal{I}}$ des états possibles dans le procédé mais interdits par la spécification (définition 3.2.1).

L'ensemble des états interdits contient un sous-ensemble de grand intérêt pour la synthèse du contrôleur — l'ensemble $\mathcal{M}_{\mathcal{B}}$ des états interdits atteignables à partir de l'ensemble des états autorisés *via* des transitions contrôlables. Cet ensemble d'états, appelé l'ensemble d'états interdits frontière (définition 3.2.4), est suffisant et nécessaire pour le calcul du contrôleur optimal, car pour accéder à tout autre état interdit il faut passer nécessairement par un état interdit frontière. Toute l'information nécessaire pour la synthèse du contrôleur est donc contenue dans l'ensemble pertinent d'états accessibles du système, $\mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle} = \mathcal{M}_{\mathcal{A}} \cup \mathcal{M}_{\mathcal{B}}$ (remarque 3.2.1).

Dorénavant toute référence à un état interdit est à considérer, par défaut, comme une référence à un état interdit frontière, sauf si autrement spécifié.

Étant donné un RdP généralisé $\mathcal{R}$ avec n places, et un état interdit $M_{B_i} \in \mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$, $M_{B_i} = P_{i_1}^{m_{i_1}} P_{i_2}^{m_{i_2}} ... P_{i_t}^{m_{i_t}}$, $t \leq n$, la contrainte interdisant M_{B_i} peut être déterminée de manière intuitive à partir de la relation (3.3.1) :

$$c_i : \sum_{j=1}^{t} M(P_{i_j}) \leq b_i$$

où t est le nombre des places marquées dans l'état M_{B_i}, $M(P_{i_j}) = m_{i_j}$ est le marquage de la place P_{i_j}, et $b_i = \mathscr{s}_i - 1$ est la borne de la contrainte, $\mathscr{s}_i$ représentant la somme des jetons du réseau dans l'état M_{B_i}, $\mathscr{s}_i = \sum_{j=1}^{t} M_{B_i}(P_{i_j}) = \sum_{j=1}^{n} M_{B_i}(P_j)$. Ceci est la manière la plus simple et intuitive d'assurer le fait que la somme des jetons dans un ensemble donné des places n'attend jamais la borne b_i. Cependant, il peut s'avérer qu'une telle contrainte impose aussi des limitations sur les comportements autorisés du système. Cette situation de hyper-restrictivité est très fréquente dans le cas des systèmes non-saufs complexes et peut avoir des implications graves sur la gamme d'activité d'un tel système.

Pour mieux illustrer ce problème, reprenons l'exemple 3.2.1. Nous avons donc un système manufacturier constitué par deux machines, $\mathscr{M}_1$ et $\mathscr{M}_2$, et trois robots. Les machines travaillent en parallèle pour produire deux variétés différentes du même type de pièces. Leurs capacités d'usinage sont d'une pièce à la fois pour la machine $\mathscr{M}_2$, et respectivement de deux pièces à la fois pour la machine $\mathscr{M}_1$. Les robots assurent le transport des pièces fabriquées par les deux machines vers un stock commun. Le cahier des charges du système stipule qu'il n'y a que trois robots dédiés au transport des pièces. Les débuts des opérations de fabrication (événements d_i) sont les seuls événements contrôlables du système. Les fins d'usinage (événements f_i) et le retour de chaque robot à l'état «disponible» (événement r) sont incontrôlables.

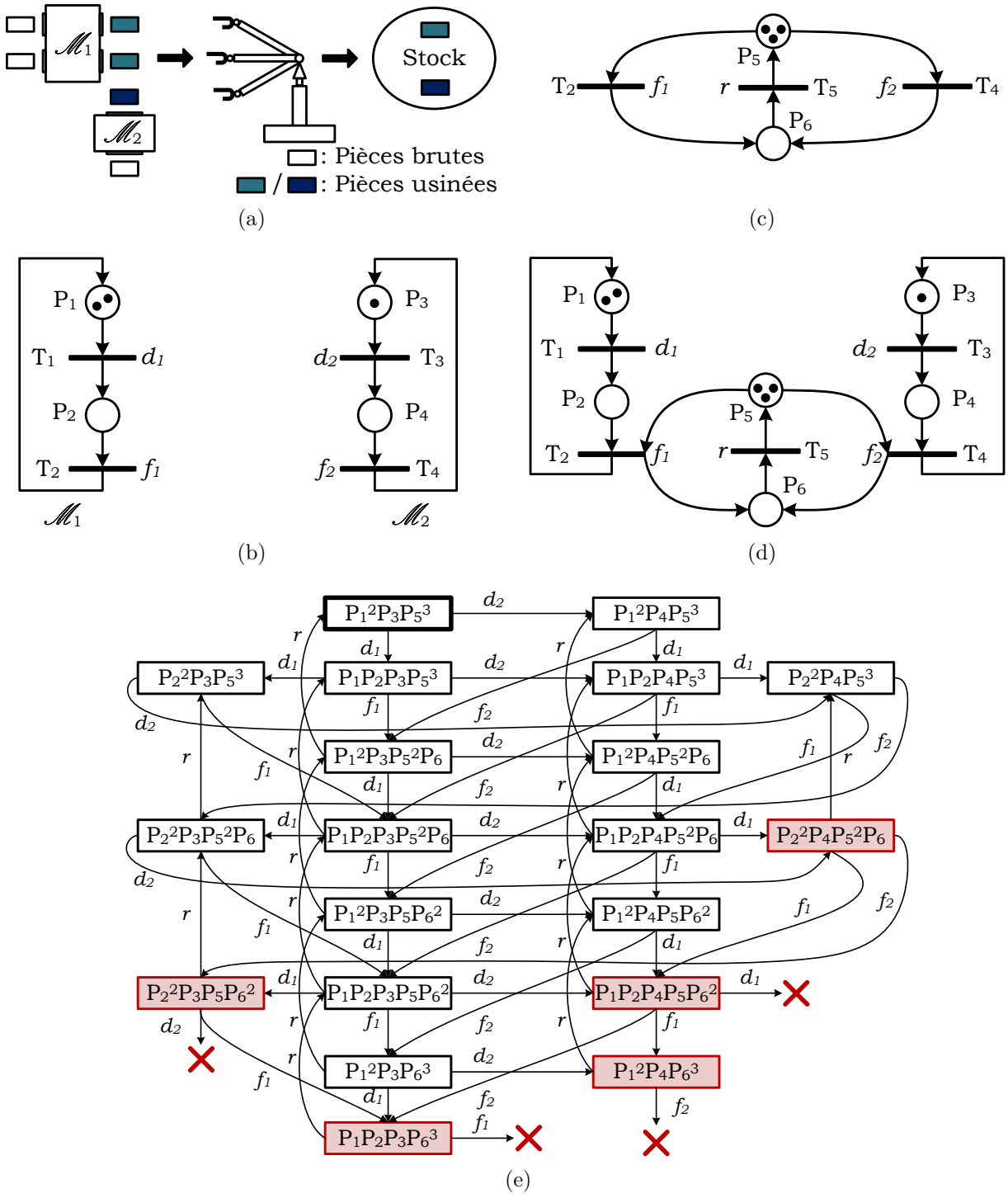


FIGURE 5.1 – Modèle du système manufacturier de l'exemple 3.2.1 : a) schéma du système manufacturier ; b) modèle RdP du procédé ; c) modèle RdP de la spécification ; d) modèle RdP du fonctionnement désiré en boucle fermée ; e) graphe pertinent de marquage.

La figure 5.1 présente le schéma du système en boucle fermée et les modèles RdP du procédé, de la spécification et du fonctionnement désiré en boucle fermée. La signification des places est la suivante :

- P_1/P_3 : la machine $\mathscr{M}_1/\mathscr{M}_2$ est à l'arrêt ;
- P_2/P_4 : la machine $\mathscr{M}_1/\mathscr{M}_2$ est en cours d'usinage d'une pièce ;
- P_5/P_6 : nombre des robots disponibles / non-disponibles à chaque moment.

Le graphe pertinent de marquage du système est donné dans la figure 5.1e. Les ensembles des états autorisés et interdits sont les suivantes :

- $\mathcal{M}_{\mathcal{B}} = \{P_1P_2P_3P_6^3;\ P_2^2P_3P_5P_6^2\ ;\ P_1P_2P_4P_5P_6^2\ ;\ P_1^2P_4P_6^3\ ;\ P_2^2P_4P_5^2P_6\ \}$,
- $\mathcal{M}_{\mathcal{A}} = \{P_1^2P_3P_5^3;\ P_1P_2P_3P_5^3\ ;\ P_1^2P_3P_5^2P_6\ ;\ P_1P_2P_3P_5^2P_6\ ;\ P_1^2P_3P_5P_6^2\ ;\ P_1P_2P_3P_5P_6^2\ ;\ P_1^2P_3P_6^3\ ;\ P_1^2P_4P_5^3\ ;\ P_1P_2P_4P_5^3\ ;\ P_1^2P_4P_5^2P_6\ ;\ P_1P_2P_4P_5^2P_6\ ;\ P_1^2P_4P_5P_6^2\ ;\ P_2^2P_3P_5^2P_6\ ;\ P_2^2P_3P_5^3\ ;\ P_2^2P_4P_5^3\ \}$.

Considérons maintenant l'état interdit $P_1P_2P_3P_6^3$. Cet état correspond au scénario où la machine $\mathscr{M}_1$ est en train de fabriquer une pièce, alors qu'aucun robot n'est disponible pour la transporter vers le stock. Il y a donc six jetons dans le système : trois jetons indiquant le nombre des robots engagés (P_6^3), et trois indiquant l'état des deux machines ($\mathscr{M}_1$ fabrique une pièce — un jeton dans chacune des places P_1 et P_2 — et $\mathscr{M}_2$ est en repos — un jeton dans P_3). Si nous calculons la contrainte associée à cet état en limitant la somme des jetons dans les places concernées, nous obtenons : $M(P_1)+M(P_2)+M(P_3)+M(P_6) \leq 5$. Cette contrainte interdit bien l'état $P_1P_2P_3P_6^3$, mais elle interdit aussi un état autorisé : $P_1^2P_3P_6^3 \in \mathcal{M}_{\mathcal{A}}$. Elle n'est donc pas admissible.

La transgression sur les comportements désirés du système ne doit jamais être possible. Il faut donc trouver une technique de détermination des contraintes admissibles qui assure un passage bijectif (une équivalence) entre l'ensemble des états interdits et celui des contraintes. On désire réaliser cet objectif en utilisant une contrainte de la forme suivante :

$$c_i : \sum_{j=1}^{n} a_{i_j} \cdot M(P_j) \leq b_i\ ,$$

où $A_i = \{a_{i_j}\}, j = \overline{1,n}$, est un ensemble de coefficients choisi de manière à assurer l'interdiction de l'état M_{B_i} tout en permettant l'atteignabilité de tout état autorisé. Notre objectif est de trouver l'ensemble des coefficients A_i le plus simple possible.

5.3 Détermination des contraintes admissibles

Étant donné les ensembles $\mathcal{M}_{\mathcal{B}}$ et $\mathcal{M}_{\mathcal{A}}$ des états interdits et, respectivement, autorisés d'un RdP $\mathcal{R}$, nous nous proposons de trouver un ensemble des contraintes qui restreint de manière exclusive l'évolution du système vers tout état interdit $\mathcal{M}_{B_i} \in \mathcal{M}_{\mathcal{B}}$. Pour alléger la présentation, nous allons d'abord expliquer notre démarche sur un seul état interdit ; la généralisation sera présentée dans la section suivante.

Soit M le vecteur de marquage générique, $M = [\ m_1 \quad m_2 \quad ... \quad m_n\ ]^T$ et soit A_i le vecteur ligne d'un ensemble de coefficients entiers, $A_i = [a_{i_j}]$, $j = \overline{1,n}$. La contrainte générique donnée ci-dessus peut être écrite sous la forme :

$$c_i : A_i \cdot M \leq b_i.$$

Ce choix de contrainte s'explique naturellement si nous analysons le problème d'états interdits d'un point de vue spatial. L'espace n-dimensionnel d'états d'un modèle RdP comprend, en essence, deux régions, correspondant aux ensembles disjointes des états autorisés et interdits du système. La contrainte doit constituer la frontière entre ces deux régions.

D'un point de vue mathématique, l'équation d'un hyperplan affine qui sépare deux régions d'un espace n-dimensionnel s'écrit sous la forme :

$$a_1 m_1 + a_2 m_2 + ... + a_n m_n = a_{n+1}.$$

C'est la frontière entre les deux ensembles.

Par rapport à cet hyperplan, les états autorisés occupent le demi-espace défini par l'inéquation :

$$a_1 m_1 + a_2 m_2 + ... + a_n m_n \leq a_{n+1},$$

alors que les états interdits résident dans l'autre demi-espace :

$$a_1 m_1 + a_2 m_2 + ... + a_n m_n > a_{n+1}.$$

Une représentation intuitive bidimensionnelle de cette idée est donnée dans la figure 5.2.

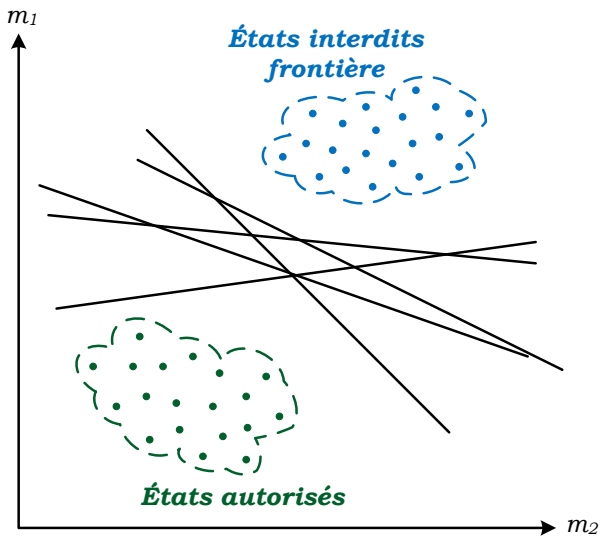


FIGURE 5.2 – Représentation de la frontière

Étant donné que l'espace d'états défini par une contrainte doit caractériser l'ensemble d'états validés par celle-ci, c'est-à-dire les états qu'elle autorise, il s'ensuit naturellement qu'une contrainte de la forme

$$c_i : a_{i_1} m_1 + a_{i_2} m_2 + ... + a_{i_n} m_n \leq a_{i_{n+1}}$$

peut être utilisée pour interdire un état M_{B_i} quelconque.

Il faut cependant se soucier de donner le maximum de liberté au système. Pour s'assurer qu'aucun état autorisé n'est inhibé par la contrainte interdisant M_{B_i} il faut garantir que

$$\sum_{j=1}^{n} a_{i_j} \cdot M_{A_\alpha}(P_j) \leq a_{i_{n+1}} \text{ soit vraie } \forall M_{A_\alpha} \in \mathcal{M}_\mathcal{A},$$

où $\mathcal{M}_\mathcal{A}$ représente l'ensemble des états autorisés.

L'ensemble $A_i = \{a_{i_j}\}$ des coefficients entiers doit vérifier le système suivant :

$$\begin{cases} \sum_{j=1}^{n} a_{i_j} \cdot M_{A_1}(P_j) - a_{i_{n+1}} \leq 0 \\ \sum_{j=1}^{n} a_{i_j} \cdot M_{A_2}(P_j) - a_{i_{n+1}} \leq 0 \\ \vdots \\ \sum_{j=1}^{n} a_{i_j} \cdot M_{A_{\text{Card}[\mathcal{M}_\mathcal{A}]}}(P_j) - a_{i_{n+1}} \leq 0 \\ \sum_{j=1}^{n} a_{i_j} \cdot M_{B_i}(P_j) - a_{i_{n+1}} > 0 \end{cases} \tag{5.3.1}$$

où Card$[\mathcal{M}_\mathcal{A}]$ représente le cardinal de l'ensemble $\mathcal{M}_\mathcal{A}$.

Proposition 5.3.1. *Étant donné un état interdit frontière M_{B_i}, sa contrainte équivalente est comme suit :*

$$c_i : \sum_{j=1}^{n} a_{i_j} \cdot M(P_j) \leq a_{i_{n+1}} \tag{5.3.2}$$

où $A_i = \{a_{i_j}\}, j = \overline{1, n+1}$ satisfait (5.3.1).

❑

En plus de notre objectif principal — l'optimalité de la commande — nous nous intéressons aussi à l'aspect de l'optimalité structurelle du contrôleur. Nous souhaitons alors ajouter le nombre minimal des places et des arcs au système. Le nombre des places de contrôle est lié au nombre des contraintes (sujet traité dans la section 5.4). La complexité du contrôleur est, cependant, déterminée par les complexités des contraintes. Tenant compte du fait que le système (5.3.1) est généralement surdéterminé, la contrainte de complexité minimale (la «borne optimale» de la figure 5.3) peut être calculée comme la solution du problème d'optimisation suivant :

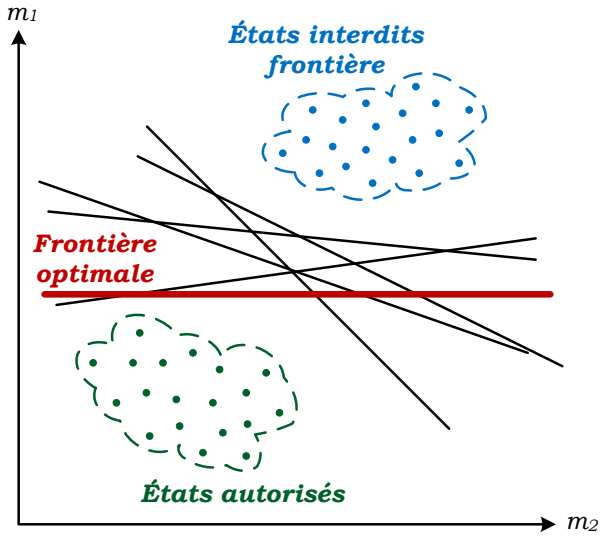


FIGURE 5.3 – Représentation de la frontière optimale

Définition 5.3.1. *L'ensemble optimal des coefficients entiers* $A_i = [a_{i_j}]$, $j = \overline{1, n+1}$, *définissant une contrainte satisfaisant la proposition 5.3.1, est donné par la minimisation du critère suivant :*

$$\min \sum_{j=1}^{n+1} a_{i_j}^2. \tag{5.3.3}$$

❑

Soit $\mathcal{M}_{C_i}$ l'espace d'états généré par une contrainte c_i définie selon la proposition 5.3.1 (l'ensemble des vecteurs $X \in \mathbb{Z}^n$ satisfaisant la proposition). Le théorème suivant peut être énoncé :

Théorème 5.3.2. *Si le problème d'optimisation* (5.3.3) *a une solution non-nulle*

$$A_{i_{min}} = \left[[A_{i_{min}}]_{j=\overline{1,n}} \;\middle|\; a_{i_{n+1_{min}}} \right],$$

alors l'espace d'états atteignables $\mathcal{M}_{C_i}$, *défini par la contrainte* $c_i : [A_{i_{min}}]_{j=\overline{1,n}} \cdot M \leq a_{i_{n+1_{min}}}$, *contient l'espace d'états autorisés du système,* $\mathcal{M}_{\mathcal{A}}$ *:*

$$\mathcal{M}_{C_i} \supset \mathcal{M}_{\mathcal{A}}. \tag{5.3.4}$$

❑

Preuve 5.3.2 :

Ce résultat est une conséquence directe de la proposition 5.3.1.

Le fait que $\mathcal{M}_{C_i} \supset \mathcal{M}_{\mathcal{A}}$ est évident par la construction de $\mathcal{M}_{C_i}$ — tout état autorisé $M_\alpha \in \mathcal{M}_{\mathcal{A}}$ est spécifiquement inclus dans $\mathcal{M}_{C_i}$, alors que seul l'état interdit associé, M_{B_i} est spécifiquement exclus de $\mathcal{M}_{C_i}$ (proposition 5.3.1).

❑

Retournant à notre exemple, nous pouvons maintenant calculer la contrainte admissible interdisant l'état $M_{B_1} = P_1P_2P_3P_6^3$:

(a) Nous identifions l'ensemble d'états autorisés :

$\mathcal{M}_{\mathcal{A}} = \{P_1^2P_3P_5^3\,;\ P_1P_2P_3P_5^3\,;\ P_1^2P_3P_5^2P_6\,;\ P_2^2P_4P_5^3\,;\ P_1P_2P_3P_5^2P_6\,;\ P_1^2P_3P_5P_6^2\,;\ P_1P_2P_3P_5P_6^2\,;\ P_1^2P_3P_6^3\,;\ P_1^2P_4P_5^3\,;\ P_1P_2P_4P_5^3\,;\ P_1^2P_4P_5^2P_6\,;\ P_1P_2P_4P_5^2P_6\,;\ P_1^2P_4P_5P_6^2\,;\ P_2^2P_3P_5^3\,;\ P_2^2P_3P_5^2P_6\}$

(b) Nous construisons la forme générale de la contrainte :

$$c_1 : \sum_{j=1}^{6} a_{1_j} M(P_j) = a_{1_1}m_1 + a_{1_2}m_2 + ... + a_{1_6}m_6 \leq a_{1_7}$$

(c) Nous calculons l'ensemble minimal des coefficients :

$$\begin{cases} M_{A_1} & : & 2a_{1_1} + a_{1_3} + 3a_{1_5} - a_{1_7} \leq 0 \\ M_{A_2} & : & a_{1_1} + a_{1_2} + a_{1_3} + 3a_{1_5} - a_{1_7} \leq 0 \\ & \vdots & \\ M_{A_{15}} & : & 2a_{1_2} + a_{1_3} + 2a_{1_5} + a_{1_6} - a_{1_7} \leq 0 \\ M_{B_1} & : & a_{1_1} + a_{1_2} + a_{1_3} + 3a_{1_6} - a_{1_7} > 0 \end{cases}$$

La solution minimale de cet système est le vecteur :

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}^T.$$

(d) Nous déterminons la contrainte admissible interdisant l'état $M_{B_1} = P_1P_2P_3P_6^3$:

$c_1 : m_2 - m_5 \leq 0.$

Remarque 5.3.1. Ce résultat extrêmement simple, contenant les marquages de seulement deux places, est surprenant et peut être justifié *à posteriori* intuitivement. Il garantit que si P_2 est marquée, la place P_5 est nécessairement marquée, ce qui signifie que lorsqu'une pièce est en fabrication dans la machine $\mathscr{M}_1$, il y a déjà un robot disponible pour décharger cette machine.

❑

5.4 Synthèse de contrôleur

Nous pouvons maintenant généraliser notre démarche. L'algorithme suivant calcule l'ensemble complet $\mathcal{C}_{\mathcal{B}}$ des contraintes admissibles pour un système en boucle fermée $\mathcal{R}$ caractérisé par les ensembles $\mathcal{M}_{\mathcal{A}}$ et $\mathcal{M}_{\mathcal{B}}$ des états autorisés et, respectivement, interdits frontière.

Algorithme 5.1 (**Algorithme général de construction des contraintes**).

Entrées : Les matrices $Ma \in \mathbb{N}^{r_A \times c_A}$ et $Mb \in \mathbb{N}^{r_B \times c_B}$, des états autorisés et, respectivement, interdits frontière, où $r_A = \text{Card}[\mathcal{M}_{\mathcal{A}}]$, $r_B = \text{Card}[\mathcal{M}_{\mathcal{B}}]$, et $c_A = c_B = n$.

Sorties : La matrice des coefficients optimaux, A, la matrice de pondération des contraintes, L^T, le vecteur des bornes, b_v, et une variable booléenne indiquant si un contrôleur maximal permissif peut être trouvé pour l'hypothèse donnée ($\mathcal{M}_{\mathcal{A}}$ and $\mathcal{M}_{\mathcal{B}}$), *ind*.

Pas 1 : Initialisation :

$$ind = true\ , \quad A = [0]\ , \quad L^T = [0]\ , \quad b_v = [0]\,.$$

Pas 2 : Pour chaque état interdit $i \in r_B$,

1. Si l'état M_{B_i} est interdit par l'une des contraintes déjà calculées :
   1.1. Commuter l'indicateur logique de convergence du problème d'optimisation pour l'état courant : $ind(i) = true$.
2. Autrement :
   2.1. Résoudre le problème d'optimisation quadratique :

$$\min_{A_i}\ A_i^T \cdot A_i\ \text{, tel que } X \cdot A_i \leq y$$

où :

- $A_i = [a_{i_j}], j = \overline{1, n+1}$,
- $A_i^T \cdot A_i = \sum_{j=1}^{n+1} a_{i_j}^2$,
- $X = \begin{bmatrix} Ma & -1 \\ -Mb(i,:) & 1 \end{bmatrix}$,
- $y = [0\ ...\ 0 \quad -1]$.

   2.2. Actualiser l'indicateur logique de convergence du problème d'optimisation pour l'état courant : $ind(i) \in \{true, false\}$.

2.3. Si $ind(i) == true$, actualiser la matrice des coefficients avec la solution obtenue pour l'état interdit i :

$$A^T = \left[\begin{array}{c|c} A & A_i \end{array} \right]^T.$$

3. Actualiser l'indicateur de sortie de l'algorithme : $ind = ind \wedge ind(i)$.

$ind = ind \wedge ind(i)$.

Pas 3 : Si l'algorithme est terminé avec succès ($ind == true$) :

1. Actualiser la matrice des pondération et le vecteur des bornes du contrôleur :

$L^T = A(:, 1 : n)$
$b_v = A(:, n + 1)$.

❑

Remarque 5.4.1. Dans le scénario le moins avantageux (où chaque contrainte interdit un et un seul état de $\mathcal{M}_{\mathcal{B}}$) nous devons résoudre (au maximum) r_B problèmes d'optimisation quadratique. Comme chaque optimisation est polynomiale dans la dimension de la variable, il s'ensuit que l'algorithme est dans la classe de complexité P (polynomiale).

❑

Soit $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}}$ l'espace d'états accessibles défini par $\mathcal{C}_{\mathcal{B}}$, $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} \subset \mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle}$.

Théorème 5.4.1. *L'espace d'états $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}}$, défini par l'ensemble $\mathcal{C}_{\mathcal{B}}$ des contraintes obtenues avec l'algorithme 5.1, est égal à l'espace d'états autorisés du système :*

$$\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} = \mathcal{M}_{\mathcal{A}}. \tag{5.4.1}$$

❑

Preuve 5.4.1 :

L'implication $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} \supseteq \mathcal{M}_{\mathcal{A}}$ est rendue évidente par la construction de $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}}$ (théorème 5.3.2).

Pour montrer que $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} \subseteq \mathcal{M}_{\mathcal{A}}$, supposons que $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} \nsubseteq \mathcal{M}_{\mathcal{A}}$. Alors $\exists\, M_i \in \mathcal{M}_{\mathcal{C}_{\mathcal{B}}}$ tel que $M_i \notin \mathcal{M}_{\mathcal{A}}$. Si $M_i \notin \mathcal{M}_{\mathcal{A}}$ sachant que $\mathcal{M}_{\mathcal{C}_{\mathcal{B}}} \subset \mathcal{M}_{Acs-l}^{\langle \mathcal{R}, M_0 \rangle} \Longleftrightarrow M_i \in \mathcal{M}_{\mathcal{B}} \Longleftrightarrow \sum_{j=1}^{n} a_{i_j} \cdot M_i(P_j) > b_i$, ce qui contredit l'hypothèse que $M_i \in \mathcal{M}_{\mathcal{C}_{\mathcal{B}}}$.

❑

Nous pouvons maintenant calculer l'ensemble des contraintes caractérisant le contrôleur optimal pour notre exemple :

(a) Identification des ensembles d'états autorisés et interdits :

- $M_A = \{P_1^2P_3P_5^3\,;\ P_1P_2P_3P_5^3\,;\ P_1^2P_3P_5^2P_6\,;\ P_2^2P_4P_5^3\,;\ P_1P_2P_3P_5^2P_6\,;\ P_1^2P_3P_5P_6^2\,;\ P_1P_2P_3P_5P_6^2\,;\ P_1^2P_3P_6^3\,;\ P_1^2P_4P_5^3\,;\ P_1P_2P_4P_5^3\,;\ P_1^2P_4P_5^2P_6\,;\ P_1P_2P_4P_5^2P_6\,;\ P_1^2P_4P_5P_6^2\,;\ P_2^2P_3P_5^3\,;\ P_2^2P_3P_5^2P_6\}$
- $M_B = \{P_1P_2P_3P_6^3\,;\ P_2^2P_3P_5P_6^2\,;\ P_1P_2P_4P_5P_6^2\,;\ P_1^2P_4P_6^3\,;\ P_2^2P_4P_5^2P_6\}$

(b) Construction des contraintes :

$$c_i : \sum_{j=1}^{6} a_{i_j} M(P_j) = a_{i_1}m_1 + a_{i_2}m_2 + ... + a_{i_6}m_6 \leq a_{i_7}$$

(c) Calcul de la matrice minimale des coefficients :

Nous obtenons les 5 contraintes suivantes.

$$\begin{cases} M_{B_1} : A_1 = [\ 0 \quad 1 \quad 0 \quad 0 \quad -1 \quad 0 \quad 0\ ] \\ M_{B_2} : A_2 = [\ 0 \quad 1 \quad 0 \quad 0 \quad -1 \quad 0 \quad 0\ ] \\ M_{B_3} : A_3 = [\ 0 \quad 1 \quad 0 \quad 1 \quad -1 \quad 0 \quad 0\ ] \\ M_{B_4} : A_4 = [\ 0 \quad 1 \quad 0 \quad 1 \quad -1 \quad 0 \quad 0\ ] \\ M_{B_5} : A_5 = [\ 0 \quad 1 \quad 0 \quad 1 \quad -1 \quad 0 \quad 0\ ] \end{cases}$$

Nous constatons que plusieurs d'entre eux sont identiques. Seules les deux contraintes indépendantes suivantes sont conservées :

$$\begin{cases} A = [\ 0 \quad 1 \quad 0 \quad 0 \quad -1 \quad 0 \quad 0\ ] \\ A' = [\ 0 \quad 1 \quad 0 \quad 1 \quad -1 \quad 0 \quad 0\ ] \end{cases}$$

$$\Downarrow$$

$$L^T = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & -1 & 0 \end{bmatrix} ; b_v = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(d) Synthèse du contrôleur :

$$W_C = -L^T \cdot W_P = \begin{bmatrix} -1 & 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & 0 & 1 \end{bmatrix}$$

$$M_{C_{ini}} = b_v - L^T \cdot M_{P_{ini}} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

Le résultat de la synthèse est donné par le contrôleur optimal de la figure 5.4.

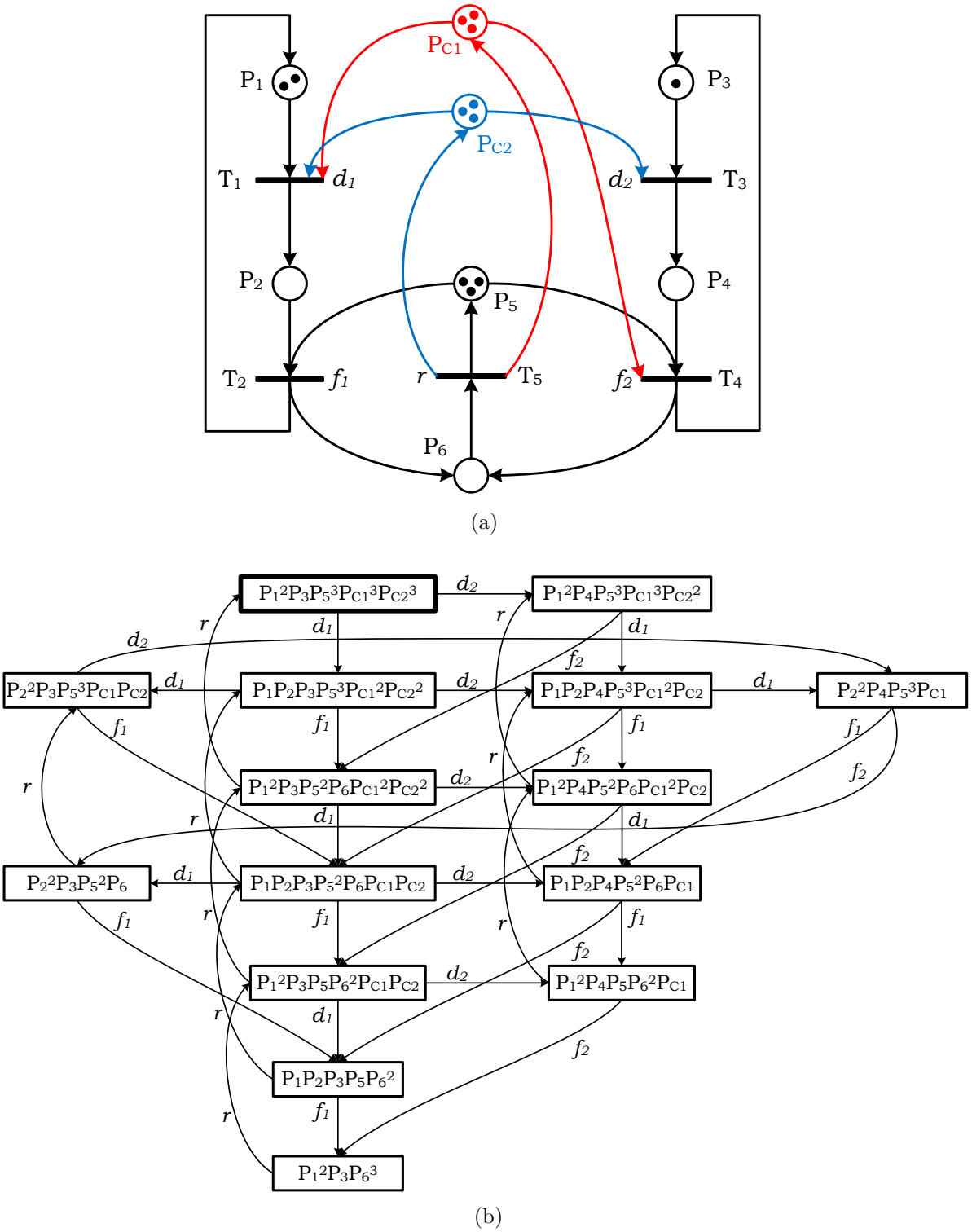


FIGURE 5.4 – Le système commandé : a) modèle RdP ; b) graphe de marquage.

Considérons le contrôleur obtenu par ajout des places de contrôle. Soit $\mathcal{M}_{\mathcal{A}_\mathcal{C}}$ l'ensemble de ses états accessibles. Comment peut-on garantir l'optimalité du contrôleur ? Il faut s'assurer qu'aucun état interdit supplémentaire ne peut être généré par la synchronisation entre les places de contrôle et le système en boucle fermée (l'ensemble procédé + spécification). Si cette synchronisation est réalisé *via* des événements incontrôlables, se peut-il que l'espace d'états accessibles du système commandé dépasse l'espace d'états du fonctionnement autorisé ($\mathcal{M}_\mathcal{A}$) ?

Le corollaire suivant garantit l'optimalité du contrôleur obtenu avec l'algorithme 5.1, même dans l'éventualité des synchronisations incontrôlables entre celui-ci et le modèle du système en boucle fermée à commander :

Corollaire 5.4.2. *L'ensemble $\mathcal{M}_{\mathcal{C}_\mathcal{B}}$ d'états accessibles défini par l'ensemble des contraintes ($\mathcal{C}_\mathcal{B}$) est égal à l'ensemble des états accessibles du contrôleur, $\mathcal{M}_{\mathcal{A}_\mathcal{C}}$:*

$$\mathcal{M}_{\mathcal{C}_\mathcal{B}} = \mathcal{M}_{\mathcal{A}_\mathcal{C}} \tag{5.4.2}$$

Le contrôleur obtenu avec la méthode des invariants à partir de l'ensemble des contraintes obtenues avec l'algorithme 5.1 est optimal.

❑

Preuve 5.4.2 :

• L'implication $\mathcal{M}_{\mathcal{C}_\mathcal{B}} \subseteq \mathcal{M}_{\mathcal{A}_\mathcal{C}}$ est déduite de la méthode de construction de $\mathcal{M}_{\mathcal{C}_\mathcal{B}}$ et $\mathcal{M}_{\mathcal{A}_\mathcal{C}}$. Si toutes les transitions sont contrôlables, l'ensemble $\mathcal{M}_{\mathcal{A}_\mathcal{C}}$ des états accessibles du système commandé serait identique au comportement autorisé : $\mathcal{M}_{\mathcal{A}_\mathcal{C}} = \mathcal{M}_\mathcal{A}$ (prouvé dans [Yamalidou et al., 1996]).

Dans notre cas, il y a des transitions incontrôlables. Est-ce qu'il y a des états interdits additionnels générés par la synchronisation entre le système et le contrôleur. Supposons que c'est le cas. L'ensemble $\mathcal{M}_{\mathcal{A}_\mathcal{C}}$ serait alors composé par la réunion du comportement autorisé ($\mathcal{M}_\mathcal{A}$) et de l'ensemble d'états interdits additionnels ($\mathcal{M}_{F_A}$) : $\mathcal{M}_{\mathcal{A}_\mathcal{C}} = \mathcal{M}_\mathcal{A} \cup \mathcal{M}_{F_A}$.

Nous avons montré (dans le théorème 5.4.1) que l'espace d'états $\mathcal{M}_{\mathcal{C}_\mathcal{B}}$, défini par l'ensemble des contraintes $\mathcal{C}_\mathcal{B}$, est égal au comportement autorisé $\mathcal{M}_\mathcal{A}$. Cependant, $\mathcal{M}_{\mathcal{A}_\mathcal{C}} \supseteq \mathcal{M}_\mathcal{A}$. Il s'ensuit que $\mathcal{M}_{\mathcal{C}_\mathcal{B}} \subseteq \mathcal{M}_{\mathcal{A}_\mathcal{C}}$.

• Pour montrer que $\mathcal{M}_{\mathcal{C}_\mathcal{B}} \supseteq \mathcal{M}_{\mathcal{A}_\mathcal{C}}$, supposons que $\mathcal{M}_{\mathcal{C}_\mathcal{B}} \not\supseteq \mathcal{M}_{\mathcal{A}_\mathcal{C}}$.

Dans ce cas la, $\exists M_i \in \mathcal{M}_{\mathcal{A}_\mathcal{C}}$ et $M_i \notin \mathcal{M}_{\mathcal{C}_\mathcal{B}} \Longrightarrow \exists M_j \in \mathcal{M}_{\mathcal{C}_\mathcal{B}}$ et $\exists T \in \mathcal{T}_U$ tel que $M_j \xrightarrow{T} M_i$, où $\mathcal{T}_U$ est l'ensemble des transitions incontrôlables.

Cependant, $M_i \notin \mathcal{M}_{\mathcal{C}_\mathcal{B}} \iff M_i \notin \mathcal{M}_\mathcal{A}$ (théorème 5.4.1), ce qui signifie que $M_i \in \mathcal{M}_\mathcal{B}$.

Mais si $M_i \in \mathcal{M}_\mathcal{B}$ et $M_j \xrightarrow{T \in \mathcal{T}_U} M_i$, alors $M_j \in \mathcal{M}_\mathcal{B} \iff M_j \notin \mathcal{M}_\mathcal{A} \iff M_j \notin \mathcal{M}_{\mathcal{C}_\mathcal{B}}$, ce qui contredit l'hypothèse que $M_j \in \mathcal{M}_{\mathcal{C}_\mathcal{B}}$.

❑

Remarque 5.4.2. Dans le cas général, une contrainte bloque plusieurs états interdits. Il en résulte que le nombre des places de contrôle est, en réalité, beaucoup plus petit que Card$[\mathcal{M}_{\mathcal{B}}]$. Il est donc important de déterminer le nombre minimal de contraintes à conserver. Nous allons proposer une méthode de réduction dans la suite de ce chapitre.

❑

5.5 Optimalité structurelle

En plus du problème de l'optimalité du contrôleur, dans cette thèse nous nous intéressons aussi à un aspect de grande importance pratique : l'optimalité structurelle.

Le principe de l'optimalité structurelle est d'ajouter le minimum de structures de contrôle (places et arcs) au modèle. Ce problème a déjà été abordé dans les sections 5.3 et 5.4, par la minimisation du nombre d'arcs reliés au places de contrôle. Maintenant nous nous focalisons sur un deuxième point, qui cherche à minimiser le nombre de places de contrôle. Nous parlerons alors de contrôleur optimal *minimal* (il y a deux optimisations successives).

En pratique, la cardinalité de l'ensemble final des contraintes (obtenu avec l'algorithme 5.1) est inférieure au nombre des états interdits frontière (cas des contraintes identiques). Cependant, il est en général possible de minimiser d'avantage le nombre des contraintes. L'algorithme 5.1 traite les états interdits de manière indépendante. Néanmoins, une vision globale est indispensable si on veut réduire au maximum l'ensemble des contraintes.

Une fois son existence établie (le système (5.3.1) a une solution pour tout état interdit $\mathcal{M}_{B_i} \in \mathcal{M}_{\mathcal{B}}$), le contrôleur optimal minimal peut être déterminé en réalisant une analyse par sous-ensemble de l'ensemble d'états interdits frontière. Cette analyse est nécessaire parce qu'il est possible d'avoir des situations où plusieurs contraintes définissant des ensembles disjoints d'états sont couvertes par une contrainte plus générale (définissant un espace plus riche d'états).

Pour être efficace, cette analyse doit être descendante.

Il faut donc en premier vérifier si une contrainte unique solution de problème existe. Pour que ce soit le cas, il faut que

$$\sum_{j=1}^{n} a_i \cdot M_{A_\alpha}(P_j) \leq a_{n+1} \text{ soit vraie } \forall M_{A_\alpha} \in \mathcal{M}_{\mathcal{A}},$$

pendant que

$$\sum_{j=1}^{n} a_j \cdot M_{B_\beta}(P_j) > a_{n+1}, \forall M_{B_\beta} \in \mathcal{M}_{\mathcal{B}}.$$

Si c'est pas le cas, il faut commencer l'analyse par sous-ensemble ; c'est-à-dire qu'il faut vérifier dans une première étape tout sous-ensemble $\mathcal{S}_1$ de $\mathcal{K} = \text{Card}[\mathcal{M}_\mathcal{B}] - 1$ états, puis tout sous-ensemble de $\mathcal{S}_2$ de $\mathcal{K} = \text{Card}[\mathcal{M}_\mathcal{B}] - 2$ états, et ainsi de suite jusqu'à la détermination d'une solution de sous-ensemble. Il existe toujours une solution où $\mathcal{K} = 1$ (algorithme 5.1).

Supposons qu'une telle solution est trouvée pour le sous-ensemble $\mathcal{S}_i$ de $\mathcal{K} = \text{Card}[\mathcal{M}_\mathcal{B}] - i$ états. On répète le même processus pour les i états interdits frontière restants.

Soit $A^{\mathcal{S}_i} = [a_j^{\mathcal{S}_i}]$, $j = \overline{1, n+1}$, l'ensemble des coefficients d'un sous-ensemble $\mathcal{S}_i$ donné. Ces coefficients peuvent être déterminés de manière systématique, en remplaçant la dernière inégalité du système (5.3.1) avec $\mathcal{K} = \text{Card}[\mathcal{S}_i]$ inégalités, chacune correspondant à un état interdit du sous-ensemble $\mathcal{S}_i$:

$$\begin{cases} \sum_{j=1}^{n} a_j^{\mathcal{S}_i} \cdot M_{A_1}(P_j) - a_{n+1}^{\mathcal{S}_i} \leq 0 \\ \vdots \\ \sum_{j=1}^{n} a_j^{\mathcal{S}_i} \cdot M_{A_{\text{Card}[M_A]}}(P_j) - a_{n+1}^{\mathcal{S}_i} \leq 0 \\ \sum_{j=1}^{n} a_j^{\mathcal{S}_i} \cdot M_{B_1}^{\mathcal{S}_i}(P_j) - a_{n+1}^{\mathcal{S}_i} > 0 \\ \vdots \\ \sum_{j=1}^{n} a_j^{\mathcal{S}_i} \cdot M_{B_\mathcal{K}}^{\mathcal{S}_i}(P_j) - a_{n+1}^{\mathcal{S}_i} > 0 \end{cases} \tag{5.5.1}$$

où $M_{B_\beta}^{\mathcal{S}_i}$, $\beta = \overline{1, \mathcal{K}}$, est un état de $\mathcal{S}_i$.

Cette technique est décrite formellement dans l'algorithme suivant :

Algorithme 5.2 (**Algorithme de minimisation du nombre des contraintes**)**.** Étant donnés les ensembles $\mathcal{M}_\mathcal{A}$ et $\mathcal{M}_\mathcal{B}$ d'états autorisés et, respectivement, interdits frontières d'un RdP $\mathcal{R}$, cet algorithme détermine l'ensemble minimal des contraintes, $\mathcal{C}_\mathcal{F}$, nécessaire pour la synthèse d'un contrôleur maximal permissif utilisant la méthode des invariants.

1. Vérification de l'existence d'une solution optimale de contrôle (*i.e.* vérifier si le système (5.3.1) donne une solution $\forall M_{B_i} \in \mathcal{M}_\mathcal{B}$). Si oui, passer au pas 2. Autrement, aller au pas 9.
2. Initialisation du sous-ensemble à traiter : $\mathcal{S} = M_B$.
3. Si le sous-ensemble courant offre une solution de sous-ensemble, $A^\mathcal{S} = [a_j^\mathcal{S}]$, aller au pas 6.
4. Pour tout sous-ensemble $\mathcal{S}_i \subset \mathcal{S}$, avec $\text{Card}[\mathcal{S}_i] \equiv \text{Card}[\mathcal{S}] - 1$,
   4.1 Si une solution de sous-ensemble, $A^{\mathcal{S}_i} = [a_j^{\mathcal{S}_i}]$, existe pour $\mathcal{S}_i$, aller au pas 6.

5. Pour chaque sous-ensemble $\mathcal{S}_i \subset \mathcal{S}$, avec $\text{Card}[\mathcal{S}_i] \equiv \text{Card}[\mathcal{S}] - 1$,

   5.1 $\mathcal{S} = \mathcal{S}_i$ et aller au pas 4.

6. Ajout de la solution courante, $c^{\mathcal{S}}$, à l'ensemble final des contraintes, $\mathcal{C}_{\mathcal{F}}$.
7. Exclusion de $\mathcal{M}_{\mathcal{B}}$ du sous-ensemble déjà solutionné : $\mathcal{M}_{\mathcal{B}} = \mathcal{M}_{\mathcal{B}} - \mathcal{S}$.
8. Est-ce que tous les états interdits ont été traités ? Si oui, passer au pas 9. Autrement, aller au pas 2.
9. Stop.

❑

L'ensemble final des contraintes, $\mathcal{C}_{\mathcal{F}}$, obtenu avec l'algorithme 5.2 assure la minimalité de la solution de contrôle, en termes de nombre des places de contrôle et de la complexité des contraintes.

Reprenons l'exemple 3.2.1. L'implémentation de l'algorithme 5.2 révèle le fait qu'une seule contrainte est suffisante pour assurer un contrôle optimal (figures 5.5 et 5.6) :

$$c^{\mathcal{M}_{\mathcal{B}}} : m_2 + m_4 - m_5 \leq 0 \Rightarrow A^{\mathcal{M}_{\mathcal{B}}} = \begin{bmatrix} 0 & 1 & 0 & 1 & -1 & 0 & 0 \end{bmatrix}^T \Rightarrow$$
$$\Rightarrow \mathcal{W}_{\mathscr{C}} = \begin{bmatrix} -1 & 0 & -1 & 0 & 1 \end{bmatrix} \; ; \; M_{\mathscr{C}_{ini}} = [3]$$

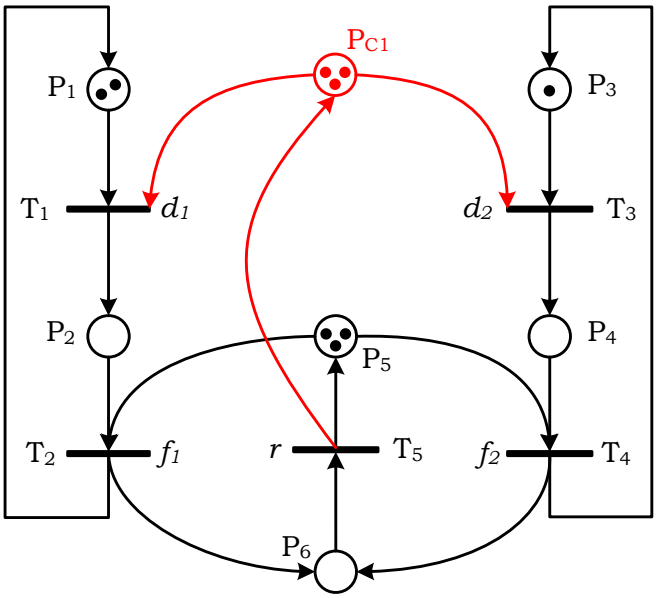


FIGURE 5.5 – Modèle RdP du système commandé avec le contrôleur optimal minimal

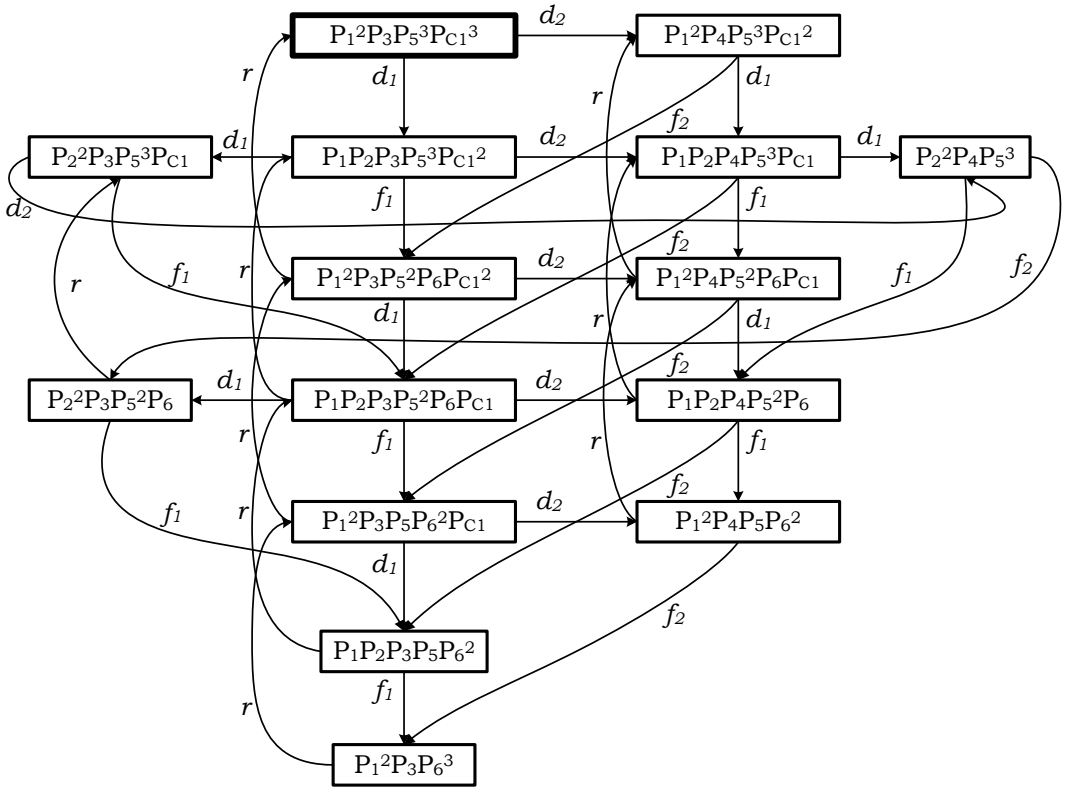


FIGURE 5.6 – Graphe de marquage du système commandé avec le contrôleur optimal minimal

5.6 Étude de cas

Dans cette section nous allons implémenter les techniques développées dans ce chapitre sur la ligne d'assemblage industriel de la figure 5.7. Le système comprend un convoyeur, une station d'assemblage, trois portes barrières (B_{1-3}) et cinq capteurs (C_{1-5}). Une station d'entrée / sortie relie le système d'assemblage avec les autres systèmes de la ligne et assure l'entrée / sortie des pièces dans la boucle d'assemblage.

La boucle d'assemblage est divisée en trois zones :

1. la zone d'entrée, entre la station entrée / sortie et la porte B_1,
2. la zone d'assemblage, entre les portes B_1 et B_2 et
3. la zone de sortie, entre les portes B_2 et B_3.

Une pièce entre dans le système par la station entrée / sortie, parcourt la zone d'entrée, et puis est admise dans la zone d'assemblage, où elle est introduite à l'intérieur de la station d'assemblage afin d'être usinée. Une fois l'usinage fini, les pièces assemblées sont retournées sur le convoyeur, parcourent la zone de sortie et quittent la boucle d'assemblage *via* la station entrée / sortie.

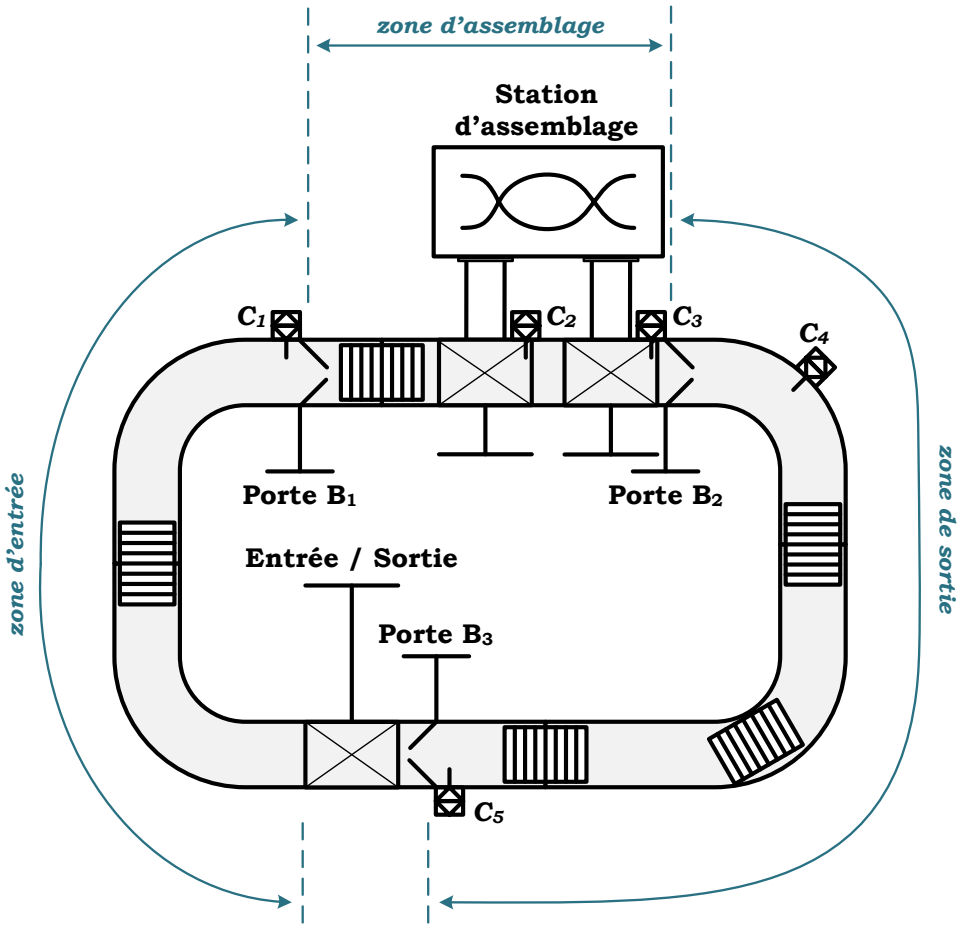


FIGURE 5.7 – Ligne d'assemblage

Le système doit satisfaire le cahier des charges suivant :

1. le nombre maximal des pièces permis à tout moment dans la zone d'assemblage (la longueur de la file d'attente d'assemblage) est de deux ;
2. le nombre maximal des pièces permis à tout moment dans la zone de sortie (la longueur de la file d'attente de sortie) est de trois.

Les modèles RdP du procédé et de la spécification sont illustrés dans la figure 5.8, alors que le modèle du comportement désiré en boucle fermée est donné dans la figure 5.9. Les événements associés à chaque transition d'état sont donnés dans le tableau 5.1, alors que les descriptions des places sont présentées dans le tableau 5.2. Après avoir construit le graphe de marquage nous avons constaté que celui-ci a 650 états, dont 94 états interdits frontière et 472 états autorisés.

TABLE 5.1: Événements associés aux transitions d'état

Événement	**Action**
c_1a	C_1 actif (pièce détectée à l'entrée de la porte B_1)
b_1o	Ouverture porte B_1
c_1i	C_1 inactif (la pièce est sortie de la porte B_1)
b_1f	Fermeture porte B_1
c_2a	C_2 actif (pièce détectée à l'entrée de la station d'assemblage)
da	Début assemblage
c_3a	C_3 actif (la pièce est sortie de la station d'assemblage / pièce détectée à l'entrée de la porte B_2)
b_2o	Ouverture porte B_2
c_4a	C_4 actif (la pièce est sortie de la porte B_2)
b_2f	Fermeture porte B_2
c_5a	C_5 actif (pièce détectée à l'entrée de la porte B_3)
b_3o	Ouverture porte B_3
c_5i	C_5 inactif (la pièce est sortie de la porte B_3)
b_3f	Fermeture porte B_3

TABLE 5.2: Description des places

Place	**Description**
P_1	Il n'y a aucune pièce à l'entrée de la zone d'assemblage
P_2	Pièce en attente d'entrer dans la zone d'assemblage
P_3	Pièce en train d'entrer dans la zone d'assemblage
P_4	Pièce entrée dans la zone d'assemblage
P_5	Aucune pièce n'est en attente d'assemblage
P_6	Pièce en attente d'assemblage
P_7	Pièce en cours d'assemblage
P_8	Pièce en attente de quitter la zone d'assemblage
P_9	Pièce en train de quitter la zone d'assemblage
Suite sur la page suivante	

TABLE 5.2: (suite)

Place	**Description**
P_{10}	Pièce sortie de la zone d'assemblage
P_{11}	Il n'y a pas de pièces en attente de quitter la boucle d'assemblage
P_{12}	Pièce en attente de quitter la boucle d'assemblage
P_{13}	Pièce en train de quitter la boucle d'assemblage
P_{14}	Pièce sortie de la boucle d'assemblage
P_{15}	Nombre courant de pièces dans la zone d'assemblage
P_{16}	Nombre de pièces en attente de quitter la boucle d'assemblage
P_{17}	Nombre de places disponibles dans la file d'attente d'assemblage
P_{18}	Nombre de places occupées dans la file d'attente d'assemblage
P_{19}	Nombre de places disponibles dans la file d'attente de sortie
P_{20}	Nombre de places occupées dans la file d'attente de sortie

L'implémentation de notre algorithme 5.1 a donnée le résultat très intéressant suivant :

$$L^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix};$$

$b_v = [0 \;\; 0]^T$.

Cela signifie que pour ne pas autoriser l'accès aux 94 états interdits frontières et laisser accessibles les 472 états autorisés, il est nécessaire et suffisant d'imposer au système les deux conditions très simples données ci-dessous :

$$\begin{cases} c_1 : \;\; m_3 - m_{17} \leq 0 \\ c_2 : \;\; m_9 - m_{19} \leq 0 \end{cases}$$

Ce résultat est facilement validé structurellement.

Tous les états interdits frontière sont des conséquences de la synchronisation procédé – spécification *via* les événements incontrôlables b_1c et b_2c.

Pour bien assurer le respect de la spécification il est donc nécessaire d'assurer, d'une part qu'aucune pièce ne peut entrer dans la zone d'assemblage pendant que la file d'attente d'assemblage est pleine (*i.e.* la place P_3 ne peut jamais être marquée quand la

place P_{17} est vide), et d'autre part qu'aucune pièce ne peut quitter la zone d'assemblage pendant que la file d'attente de sortie est pleine (*i.e.* la place P_9 ne peut jamais être marquée quand la place P_{19} est vide).

L'implémentation de l'algorithme 5.2 révèle que, pour ce système, il n'est pas possible de simplifier d'avantage le nombre des contraintes. Le contrôleur optimal minimal de la ligne d'assemblage de la figure 5.7 peut donc être calculé :

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

$$M_{C_{ini}} = [2 \;\; 3]^T .$$

Le modèle du système commandé est donné dans la figure 5.10.

5.7 Conclusions

Dans ce chapitre nous avons proposé une solution originale au problème de l'optimalité du contrôleur, dans le contexte des RdP généralisés. L'approche consiste à déterminer des contraintes liant la théorie généralisé du contrôle par supervision des SED (R&W) et la méthode des invariants. La méthode est simple, systématique et facilement implémentable. L'espace d'états généré par l'ensemble des contraintes déterminées de cette manière est isomorphe au comportement autorisé du système en boucle fermée et permet le calcul d'un contrôleur maximal permissif (avec la méthode des invariants).

Par ailleurs, nous avons assuré l'optimisation de la forme de chaque contrainte et la minimisation du nombre final des contraintes. Cela nous a permis de déterminer le contrôleur maximal permissif minimal et ainsi de garantir l'optimalité structurelle de la solution de contrôle.

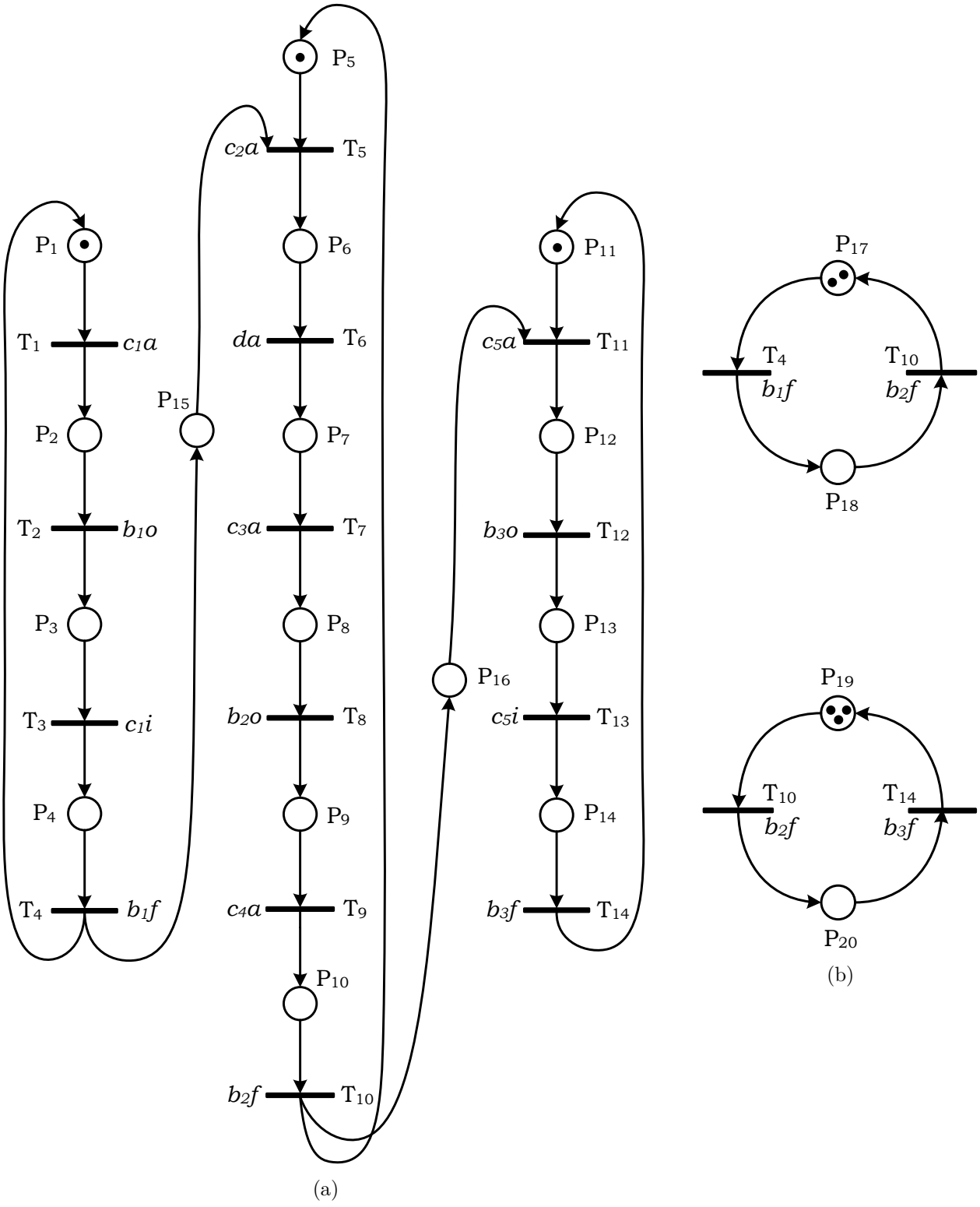


FIGURE 5.8 – Modèles RdP du procédé et de la spécification : a) le procédé ; b) la spécification.

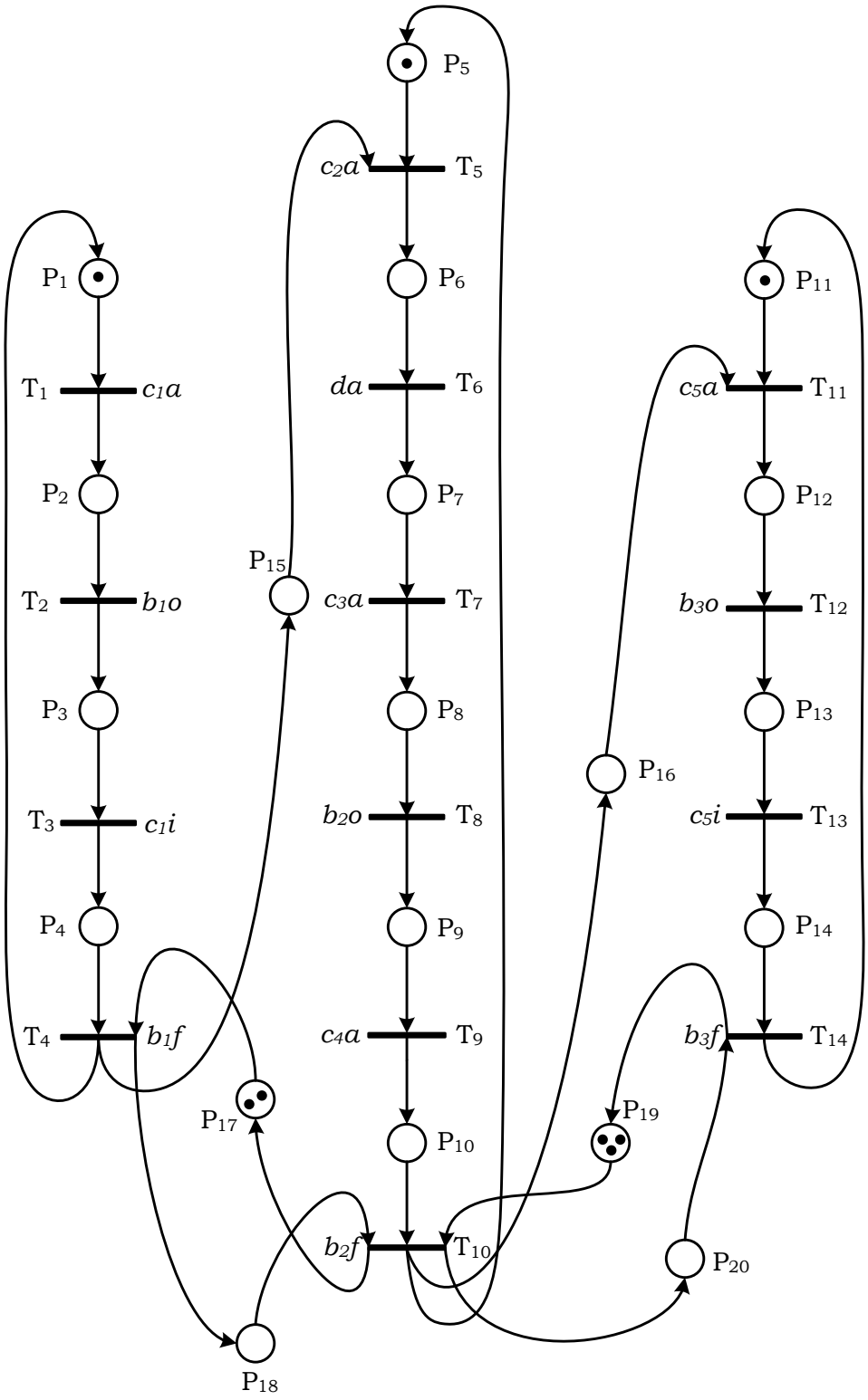


FIGURE 5.9 – Modèle RdP du fonctionnement désiré en boucle fermée

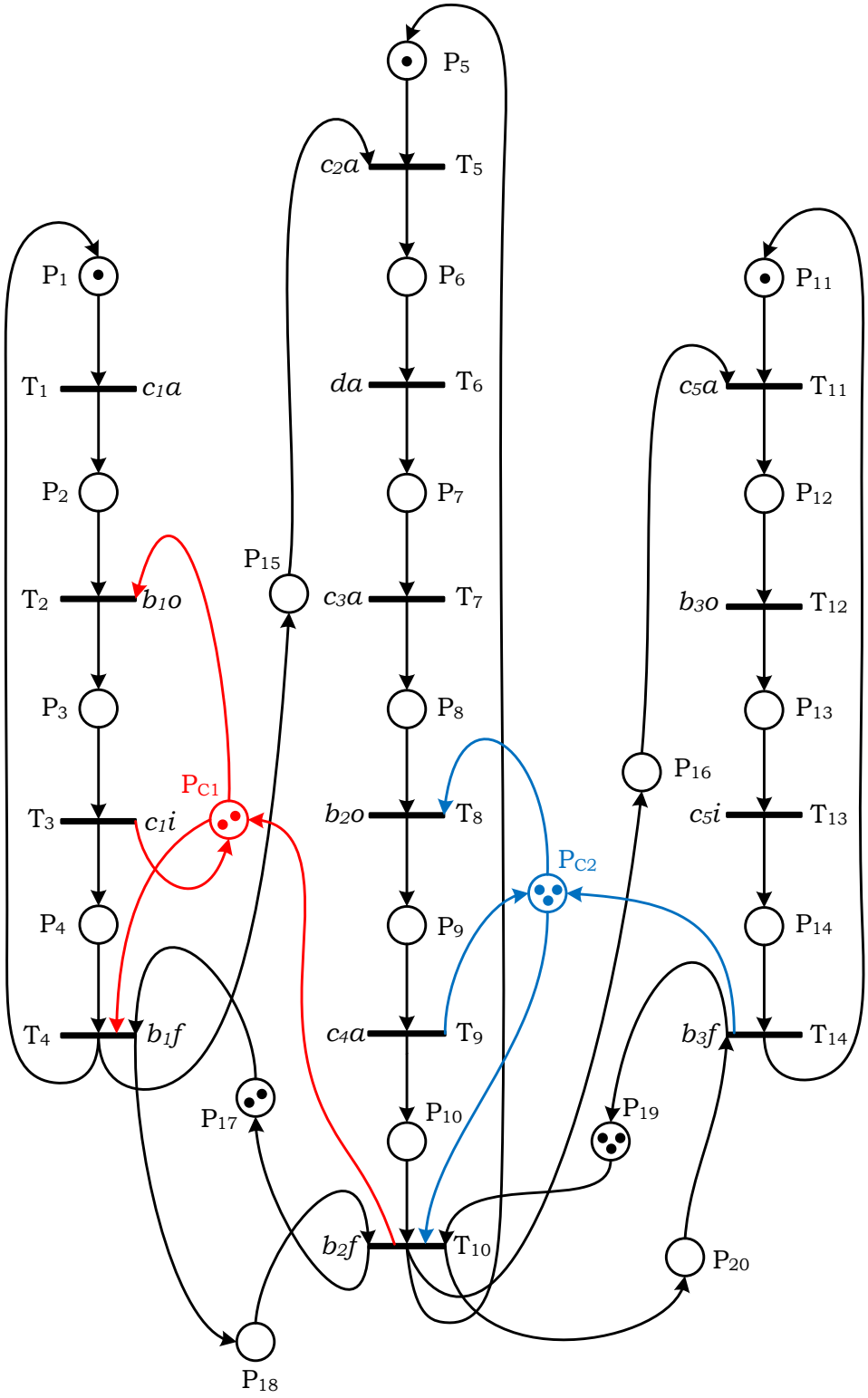


FIGURE 5.10 – Modèle RdP du système commandé

Conclusions et perspectives

Le but de la synthèse de la commande est de construire un contrôleur $\mathscr{S}$ capable de restreindre le fonctionnement d'un procédé $\mathscr{P}$ donné de telle manière qu'il respecte à tout moment le cahier des charges imposé, $\mathscr{S}_{spec}$. De plus, le contrôleur doit assurer le degré minimal nécessaire d'intrusion dans le fonctionnement du procédé. La contrôlabilité joue un rôle important dans la commande. Le contrôleur observe l'évolution du procédé et peut à tout moment interdire l'exécution des événements contrôlables de $\mathscr{P}$.

Le résultat principal pour le contrôle des SED est la théorie générale de la commande par supervision, développée par Ramadge et Wonham. Malheureusement, la méthode est très sensible à l'explosion combinatoire du nombre d'états, et très souvent la taille du contrôleur dépasse celle du procédé en isolation. Pour maîtriser la taille des contrôleurs, la méthode des invariants est remarquable, car elle exploite les propriétés structurelles des réseaux de Petri. C'est la méthode fondamentale de synthèse pour le travail présenté dans ce mémoire.

Notre travail est situé en amont de la méthode des invariants, car celle-ci ne fournit pas de solution optimale dans le cas général. Plus spécifiquement, des difficultés importantes surgissent si l'incontrôlabilité des événements est considérée. Des états interdits supplémentaires peuvent être générés si la synchronisation entre le modèle du procédé et celui de la spécification est réalisée *via* des transitions incontrôlables. De plus, cette circonstance peut occasionner des inclusions (totales ou partielles) de support entre les états interdits et les états autorisés, et entraîner l'interdiction d'un grand nombre des comportements désirables du système (si le contrôle est calculé de manière traditionnelle). Pour faire face à cet obstacle, nous avons proposé un passage bijectif (une équivalence) entre l'ensemble des états interdits et celui des contraintes. Cela permet de déterminer des contraintes admissibles qui garantissent l'optimalité même dans le contexte des transitions incontrôlables. En reliant la théorie de R&W avec la méthode des invariants nous avons réussi, d'une part à réunir les avantages des deux méthodes (la solution formelle, générale et optimale de R&W et l'élégance, l'efficacité de calcul et l'aisance d'implémentation de la méthode des invariants) et d'autre part à éviter les inconvénients de l'explosion combinatoire du nombre d'états et du problème des synchronisations incontrôlables. Notre approche nécessite la construction du graphe de marquages accessibles,

qui peut être très grand pour les systèmes complexes. Heureusement, celui-ci est construit entièrement hors ligne et est calculé une seule fois, et en conséquence le temps de réponse en temps réel du système commandé n'est pas influencé.

Nous avons proposé deux techniques pour la détermination des contraintes permettant la synthèse des contrôleurs maximaux permissifs pour les RdP synchronisés saufs et généralisés. Nous nous sommes aussi intéressés à l'optimalité structurelle des contrôleurs — un objectif d'une haute importance pratique.

Dans un premier temps, nous avons considéré le problème pour le cas des RdP saufs. Par rapport au méthodes déjà existantes, nous avons aussi pris en compte le cas des réseaux non-conservatifs. Dans ce contexte, nous avons présenté une solution pour la détermination des contraintes équivalentes aux états interdits. Cette équivalence repose sur une première définition des contraintes contenant des marquages complémentés. Pour pouvoir appliquer la méthode des invariants il a été nécessaire d'éliminer les marquages complémentés. Ceci est réalisé en exploitant les composantes conservatives des RdP. Lorsqu'il existe des places non-conservatives, nous avons montré qu'il est possible de construire un RdP conservatif isomorphe. La procédure est réalisée en ajoutant des places implicites au modèle. Celles-ci sont supprimées dans le modèle du contrôleur final.

Nous avons dans un deuxième temps envisagé la généralisation de notre technique, pour pouvoir l'appliquer à une classe plus grande des systèmes — les RdP généralisés. Malheureusement, la généralisation directe n'est pas possible, l'approche étant basée sur une opération spécifique aux RdP binaires — le complément logique. Nous sommes donc retourné à la définition de base du problème et à l'idée d'un lien entre la théorie de R&W et la méthode des invariants. À partir de ces hypothèses, nous avons développé une technique de détermination des contraintes admissibles pour la méthode des invariants qui fournit toujours un ensemble des contraintes caractérisant le contrôleur optimal, si un tel contrôleur existe. La méthode repose sur la séparabilité des ensembles d'états autorisés et interdits qui composent l'espace d'états accessibles d'un RdP. Étant donné le modèle RdP d'un système en boucle fermée, nous utilisons dans un premier temps l'approche de R&W pour déterminer les deux ensembles nécessaires et suffisants pour le calcul du contrôleur optimal : le comportement autorisé et l'ensemble des états interdits frontières. À partir de ce point, nous déterminons un ensemble des contraintes caractérisant de manière bijective le comportement autorisé. La forme générale de nos contraintes est formulée à partir d'une analyse spatiale de l'espace d'états du système. Dans ce contexte, les contraintes sont des inégalités linéaires en nombres entiers dérivées de l'équation d'hyperplans affines qui séparent les régions interdite- et autorisée- de l'espace d'états. Une méthode d'optimisation est utilisée pour assurer la complexité minimale de chaque contrainte. Nous garantissons, de cette manière, la minimisation du nombre d'arcs de contrôle. Nous avons également développé un algorithme de minimisation du nombre des contraintes, afin d'assurer l'optimalité structurelle de la solution

de contrôle obtenue. Nous avons ainsi construit une méthode de synthèse du contrôleur optimal simple, systématique et facilement implémentable.

À présent, nous avons démontré que dans tous les cas où notre méthode a une solution, le résultat obtenu caractérise un contrôleur optimal. Cependant, dans le cas où la méthode ne donne pas de solution nous ne sommes pas certains qu'il n'existe pas de contrôleur optimal. Résoudre le problème de la séparabilité des espaces d'états, c'est-à-dire, prouver formellement que si une solution de contrôle existe, alors elle peut être déterminée par notre méthode, est notre principale perspective pour l'avenir.

Un autre perspective que nous envisageons est de minimiser la complexité de calcul de l'approche ; c'est-à-dire ne pas avoir à utiliser tout l'ensemble des marquages accessibles. Cela revient à réduire considérablement le nombre d'états autorisés impliqués dans le calcul des contraintes. À cette fin, nous souhaitons modifier notre démarche de manière à pouvoir utiliser juste l'ensemble des états autorisés critiques dans le calcul des contraintes.

Un troisième centre d'intérêt est de trouver une manière de calculer des contraintes admissibles pour la méthode des invariants tout en évitant de générer le graphe de marquage. Nous estimons qu'une telle approche, intrinsèquement structurelle, permettra le développement d'un algorithme très efficace de calcul de la commande et donnera lieu à de fortes applications pratiques.

Bibliographie

Badouel, E., Darondeau, P. & Bernardinello, L. [1994], Polynomial algorithms for the synthesis of bounded nets, *in* 'Proccedings Caap 95, Lecture Notes in Computer Science 915', Springer, pp. 364–378.

Brandin, B. & Wonham, W. [1994], 'Supervisory control of timed discrete event systems', *IEEE Trans. Autom. Control* **39**(2), 329–342.

Cassandras, C. & Lafortune, S. [2008], *Introduction to Discrete Event Systems, 2nd Edition*, Springer - Verlag.

Corona, D., Giua, A. & Seatzu, C. [2004], Marking estimation of Petri nets with silent transitions, *in* 'Proc. IEEE 43rd Conference on Decision and Control (CDC 2010)', Vol. 1, Atlanta (GA), pp. 966 – 971.

David, R. & Alla, H. [1992], *Petri nets and grafcet : tools for modeling discrete event systems*, Prentice Hall.

David, R. & Alla, H. [2010], *Discrete, Continuous, and Hybrid Petri Nets*, Springer - Verlag.

Dideban, A. [2007], Synthèse de contrôleurs discrets par simplification de contraintes et de conditions, PhD thesis, Université Joseph Fourier.

Dideban, A. & Alla, H. [2006], Solving the problem of forbidden states by feedback control logical synthesis, *in* 'Proc. IEEE 32nd Annual Conference on Industrial Electronics (IECON 2006)', Paris, pp. 348–353.

Dideban, A. & Alla, H. [2007], Determination of minimal sets of control places for safe Petri nets, *in* 'Proc. IEEE American Control Conference (ACC 2007)', New York (NY), pp. 4975–4980.

Dideban, A. & Alla, H. [2008], 'Reduction of constraints for controller synthesis based on safe Petri nets', *Automatica* **44**(7), 1697–1706.

Ezpeleta, J., Colom, J. & Martinez, J. [1995], 'A Petri net based deadlock prevention policy for flexible manufacturing systems', *IEEE Trans. Robot. Autom.* **11**(2), 173–184.

Gaubert, S. [1995], 'Performance evaluation of (max,+) automata', *IEEE Trans. Autom. Control* **40**(12), 2014–2025.

Ghaffari, A., Rezg, N. & Xie, X. [2003*a*], 'Design of a live and maximally permissive Petri net controller using the theory of regions', *IEEE Trans. Robot. Autom.* **19**(1), 137–141.

Ghaffari, A., Rezg, N. & Xie, X. [2003*b*], 'Feedback control logic for forbidden-state problems of marked graphs : application to a real manufacturing system', *IEEE Trans. Autom. Control* **48**(1), 18–29.

Giua, A., Corona, D. & Seatzu, C. [2005], 'State estimation of λ-free labeled Petri nets with contact-free nondeterministic transitions', *Discrete Event Dynamic Systems* **15**, 85–108.

Giua, A. & DiCesare, F. [1994*a*], 'Blocking and controllability of Petri nets in supervisory control', *IEEE Trans. Autom. Control* **39**(4), 818–823.

Giua, A. & DiCesare, F. [1994*b*], 'Petri net structural analysis for supervisory control', *IEEE Trans. Robot. Autom.* **10**(2), 185–195.

Giua, A., DiCesare, F. & Silva, M. [1992], Generalized mutual exclusion constraints on nets with uncontrollable transitions, *in* 'Proc. IEEE International Conference on Systems, Man, and Cybernetics', Vol. 2, Chicago (IL), pp. 974–979.

Giua, A. & Seatzu, C. [2002], 'Observability of place/transition nets', *IEEE Trans. Autom. Control* **47**(9), 1424–1437.

Giua, A. & Seatzu, C. [2007], A systems theory view of Petri nets, *in* C. Bonivento, L. Marconi, C. Rossi & A. Isidori, eds, 'Advances in Control Theory and Applications', Vol. 353 of *Lecture Notes in Control and Information Sciences*, Springer Berlin / Heidelberg, pp. 99–127.

Giua, A., Seatzu, C. & Basile, F. [2004], 'Observer-based state-feedback control of timed Petri nets with deadlock recovery', *IEEE Trans. Autom. Control* **49**(1), 17–29.

Giua, A., Seatzu, C. & Júlvez, J. [2004], 'Marking estimation of Petri nets with pairs of nondeterministic transitions', *Asian J. of Control* **6**(2), 270–280.

Giua, A. & Xie, X. [2005], 'Control of safe ordinary Petri nets using unfolding', *Discrete Event Dynamic Systems* **15**, 349–373.

Holloway, L., Guan, X. & Zhang, L. [1996], 'A generalization of state avoidance policies for controlled Petri nets', *IEEE Trans. Autom. Control* **41**(6), 804–816.

Holloway, L. & Krogh, B. [1990], 'Synthesis of feedback control logic for a class of controlled Petri nets', *IEEE Trans. Autom. Control* **35**(5), 514–523.

Hopcroft, J., Motwani, R. & Ullman, J. [2007], *Introduction to Automata Theory, Languages, and Computation, 3rd Edition*, Prentice Hall.

Iordache, M. & Antsaklis, P. [2006], *Supervisory Control of Concurrent Systems - A Petri Net Structural Approach*, Birkhäuser, Boston.

Iordache, M., Moody, J. & Antsaklis, P. [2001], A method for the synthesis of liveness enforcing supervisors in Petri nets, *in* 'Proc. IEEE American Control Conference (ACC 2001)', Vol. 6, Arlington (VA), pp. 4943–4948.

Kumar, R. [1991], Supervisory Synthesis Techniques for Discrete Event Dynamical Systems : Transition Model Based Approach, PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin.

Kumar, R. & Holloway, L. [1996], 'Supervisory control of deterministic Petri nets with regular specification languages', *IEEE Trans. Autom. Control* **41**(2), 245–249.

Li, Y. & Wonham, W. [1994], 'Control of vector discrete-event systems. II. controller synthesis', *IEEE Trans. Autom. Control* **39**(3), 512–531.

Liao, H., Lafortune, S., Reveliotis, S., Wang, Y. & Mahlke, S. [2010], Synthesis of maximally-permissive liveness-enforcing control policies for gadara Petri nets, *in* 'Proc. IEEE 49th Conference on Decision and Control (CDC 2010)', Atlanta (GA), pp. 2797 – 2804.

Moody, J. & Antsaklis, P. [1998], *Supervisory Control of Discrete Event Systems Using Petri Nets*, Kluwer Academic Publishers, Boston.

Moody, J. & Antsaklis, P. [2000], 'Petri net supervisors for DES with uncontrollable and unobservable transitions', *IEEE Trans. Autom. Control* **45**(3), 462–476.

Nazeem, A., Reveliotis, S., Yin, W. & Lafortune, S. [2011], 'Designing compact and maximally permissive deadlock avoidance policies for complex resource allocation systems through classification theory : The linear case', *IEEE Trans. Autom. Control* **56**(8), 1818 – 1833.

Ramadge, P. & Wonham, W. [1987*a*], 'On the supremal controllable sublanguage of a given language', *SIAM J. Control Optim.* **25**(3), 637–659.

Ramadge, P. & Wonham, W. [1987*b*], 'Supervisory control of a class of discrete event processes', *SIAM J. Control Optim.* **25**(1), 206–230.

Ramadge, P. & Wonham, W. [1989], The control of discrete event systems, *in* 'Proc. IEEE : Special Issue on Dynamics of Discrete Event Systems', Vol. 77 - 1, pp. 81–98.

Sava, A. [2002], Sur la synthèse de la commande des systèmes à événements discrets temporisés, PhD thesis, Départment d'Automatique de GIPSA-Lab, Institut National Polytechnique de Grenoble.

Sreenivas, R. & Krogh, B. [1992], 'On Petri net models of infinite state supervisors', *IEEE Trans. Autom. Control* **37**(2), 274–277.

Uzam, M. & Wonham, W. [2006], 'A hybrid approach to supervisory control of discrete event systems coupling RW supervisors to Petri nets', *Int. J. of Adv. Manuf. Technol.* **28**(7 - 8), 747 – 760.

Vasiliu, A. & Alla, H. [2010], 'Structural optimal control for safe Petri nets', *Int. J. of Control* **83**(9), 1810 – 1822.

Vasiliu, A. & Alla, H. [2011*a*], Border forbidden states and constraints for optimal controller synthesis using generalized Petri nets. soumission Automatica.

Vasiliu, A. & Alla, H. [2011*b*], Control optimality for ordinary Petri nets, *in* '18th IFAC World Congress', Vol. 18 - 1, Milano, pp. 3599 – 3604.

Vasiliu, A., Dideban, A. & Alla, H. [2009], 'Control synthesis for manufacturing systems using non-safe Petri nets', *J. of Control Eng. and Applied Informatics* **11**(2), 43 – 50.

Wonham, W. [2011], Supervisory control of discrete event systems, Technical report, University of Toronto, Dept. of Electrical and Computer Engineering. `http://www.control.utoronto.ca/~wonham/`.

Wu, L. & Zhou, M. [2004], 'Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems', *IEEE Trans. Syst., Man, Cybern. A* **34**(1), 38–51.

Yamalidou, K., Moody, J., Lemmon, M. & Antsaklis, P. [1996], 'Feedback control of Petri nets based on place invariants', *Automatica* **32**(1), 15–28.