Reconfiguration dynamique de la commande d'un système manufacturier : approche par la synthèse de la commande

Eun Joo Lee

ÉCOLE CENTRALE DE LILLE

ANNEE 2006 **N° d'ordre : 0029**

THESE

Pour obtenir le grade de

DOCTEUR DE L'ÉCOLE CENTRALE DE LILLE

DOCTORAT DELIVRE CONJOINTEMENT PAR L'ECOLE CENTRALE DE LILLE
et l'Université des Sciences et Technologies de Lille

Discipline : AUTOMATIQUE et INFORMATIQUE INDUSTRIELLE

Présentée et soutenue publiquement le 18 décembre 2006

par

Eun Joo LEE

Reconfiguration dynamique de la commande d'un système manufacturier : approche par la synthèse de la commande

Devant le jury composé de :

Mr. AbdEl-Kader SAHRAOUI	***Rapporteur***	Professeur à l'Université de Toulouse le Mirail
Mr. Nidhal REZG	***Rapporteur***	Professeur à l'Université de Metz
Mr. Etienne CRAYE	***Examinateur***	Professeur à l'École Centrale de Lille
Mr. Pascal Berruet	***Examinateur***	Maitre de Conférences à l'Université de Bretagne sud
Melle Nathalie Dangoumau	***Co-directeur***	Maître de Conférences à l'École Centrale de Lille
Mr. Armand Toguyéni	***Directeur***	Professeur à l'École Centrale de Lille

Directeurs de Thèse :

Armand Toguyéni Professeur à l'École Centrale de Lille

Nathalie Dangoumau à l'École Centrale de Lille

Thèse préparée au Laboratoire d'Automatique, de Génie Informatique et Signal
L.A.G.I.S, UMR 8146 - École Centrale de Lille

Table des matières

Liste des tableaux

Liste des figures

Glossaires

CGEM : Contraintes Généralisées d'Exclusion Mutuelle

EC : État Courant

EO : État Objectif

EPO : Elément de Partie Opérative

ER : État de Reprise

GAS : Graphe d'Accessibilité Synchrone

GASC : Graphe d'Accessibilité Synchrone Contraint

GCRC : Graphes de Coordination des Ressources Complexes

GCST : Graphe de Commande du Système de Transport

GEC : Graphes d'Evénements Cycliques

PC : Partie Commande

PEI : Problème des Etats Interdits

PO : Partie Opérative

PSITE : Problème des Séquences Interdites de Transitions d'État

PTEI : Problème de Transitions d'Etat Interdits

PTI : Problème de Transitions Interdites

RdP : Réseaux de Petri

RdPC : Réseaux de Petri Colorés

SED : Systèmes à Événements Discrets

SFPM : Systèmes Flexibles de Production Manufacturière

SITE : Séquences Interdites de Transition d'Etats

SMR : Système Manufacturier Reconfigurable

Remerciements

Je tiens d'abord à remercier Armand TOGUYENI et Nathalie DANGOUMAU, qui m'ont guidé dès les premiers jours et ont su me soutenir dans les moments difficiles.

Je remercie sincèrement mes rapporteurs : Messieurs AbdEl-Kader SAHRAOUI et Nidhal REZG, qui ont eu l'infinie patience de relire un manuscrit trop long, envoyé bien tardivement.

Je remercie également les membres du jury : Messieurs Etienne CRAYE et Pascal BERRUET d'avoir bien voulu participer à la soutenance.

J'ai également une pensée particulière au Professeur JongKun LEE qui m'a encouragé à réaliser ma thèse en France.

Je n'oublie pas tous mes collègues de l'Ecole Centrale de Lille, mes amis de l'Eglise évangélique « toute les nations » représentent pour moi une deuxième famille et mes amis en Corée.

Je remercie également mes parant, mes sœurs, mon beau-frère et mes neveux. Je ne pouvais pas réussir sans leurs encouragements et leur amour.

Enfin, je remercie Dieu (Jésus Christ).

Introduction générale

Les systèmes de production manufacturiers sont aujourd'hui soumis à de fortes contraintes induites par un environnement incertain, changeant et dominé par une forte concurrence internationale. Cet environnement fait qu'un système de production est de plus en plus orienté vers une grande diversification des produits fabriqués en petites et moyennes séries et non vers un seul type de produit.

L'impact de cette mutation dans l'industrie se traduit par la nécessité d'avoir des systèmes capables de s'adapter aux changements de production, de se révéler flexibles et robustes, afin de répondre aux exigences en diversité, en productivité, en qualité, en optimisation des coûts d'exploitation et en la diminution des risques des défaillances.

Le respect de ces contraintes qui deviennent de plus en plus exigeantes a entraîné une révolution dans le domaine de la production. Cela se manifeste par l'utilisation de plus en plus massive des systèmes d'information puissants et surtout par une automatisation croissante des ateliers et des processus.

L'automatisation des systèmes de production permet de gagner en productivité et d'accroître la compétitivité des entreprises travaillant dans le domaine de la production de biens manufacturés. Donc, elle est un enjeu économique important. Cette automatisation nécessite l'élaboration de méthodologies intégrant toutes les phases du cycle de vie d'un système, de la spécification à l'exploitation, et cela dans un contexte sûr de fonctionnement.

Cependant, étant donné les différents paramètres à prendre en compte dans un système de production, ces derniers deviennent très complexes. Cette complexité concerne à la fois la partie commande et la partie surveillance/supervision.

Le concept de Système Manufacturier Reconfigurable (SMR) a été proposé par l'université du Michigan en 1999, peut être une nouvelle solution pour répondre à la compétition international engendrée par l'émergence des nouveaux pays industrialisés (Chine,

Inde, Brésil, …). Pour gagner en compétitivité et répondre aux exigences d'un marché en mutations constantes, Yoran Korean pense qu'il est nécessaire de concevoir des systèmes de production facilement reconfigurable afin de les adapter rapidement à une demande changeante [KOR99]. Cela nécessite donc d'avoir des systèmes à la fois reconfigurable au niveau matériel mais également au niveau de la commande.

Nous nous intéressons donc dans ce travail à la conception de la commande d'une classe particulière de Systèmes de Production : les Systèmes à Événements Discrets (SED) et plus particulièrement les systèmes manufacturiers. Ces derniers sont des systèmes dynamiques dont l'espace d'états est discret. Leur évolution est régie par l'occurrence d'événements discrets. Ces événements physiques provoquent une modification de l'état du système. Leur nature discrète et flexible rendent leur conception difficile. Les démarches de recherche actuelle tendent ainsi à améliorer et à renforcer la rigueur de leur conception.

Dans le domaine de la Sûreté de Fonctionnement, les approches de type Surveillance/Supervision peuvent être dissociées en deux grandes classes : l'approche système correspondant en fait au pilotage en temps réel des ressources du système de production et la notion de commande supervisée visant à interdire l'accès à certains états jugés dangereux.

Les approches de synthèse de commande sont caractérisées par une prise en compte séparée des spécifications du cahier de charges et des contraintes du procédé. Formellement, ces approches permettent a priori de mieux garantir le respect du cahier de charges. En productique, elles sont utilisées pour la conception de la commande de coordination. Des équipes de recherche commencent à proposer des extensions de la synthèse de commande permettant de surveiller et superviser les SED. Toutefois, sur le plan de la théorie, les SED ont longtemps péché par un manque de conceptualisation et de formalisation comparable au domaine de l'automatique des systèmes continus.

Dans cet esprit, en 1987 Ramadge et Wonham ont proposé la théorie du « supervisory control » (contrôle par supervision) pour modéliser la commande des systèmes à événements discrets [RAM87a]. Cette théorie RW, qui représente l'une des avancées les plus significatives en matière de synthèse des SED permet d'établir pour ces derniers SED des concepts et notions analogues à ceux de l'automatique continu.

Elle est basée sur les automates à états finis et les langages formels [RAM87a] [WON00] [WON03]. Mais, elle a une limite importante : l'approche de Ramadge et Wonham

est basée sur l'utilisation du formalisme des automates à états finis. En effet lorsque les modèles partiels des ressources considérées partagent peu d'événements communs, la composition synchrone peut entraîner une explosion combinatoire notamment dans le cas d'un système composé de nombreuses ressources évoluant en parallèle. Pour modéliser les Systèmes à Événements Discrets (SED), une alternative aux automates à états finis réside dans les Réseaux de Petri (RdP). En effet, les RdP permettent de modéliser facilement différentes classes de système dont les comportements sont notamment caractérisés par des évolutions parallèles et des synchronisations.

Ce travail porte donc sur la synthèse de la commande de SED au fonctionnement caractérisé par un fort parallélisme avec des états de synchronisation. Ainsi, les Réseaux de Petri (RdP) apparaissent comme un outil naturel pour cette synthèse.

Dans ce cadre, beaucoup d'approches à base de RdP portent sur l'exploitation du graphe d'accessibilité. En effet de nombreux travaux concernent la synthèse de commande répondant à la problématique d'états interdits. Mais le graphe d'accessibilité est lui-même un automate. Il peut donc mener à un problème d'explosion combinatoire. Dans nos travaux, nous allons proposer une nouvelle méthode de synthèse de RdP selon le principe du produit synchrone d'automates. Notre objectif est de répondre à une nouvelle problématique que nous introduisons dans cette étude : le Problème des Séquences Interdites de Transitions d'Etat (PSITE). Notre approche est également basée sur la théorie des régions de Ghaffari. Notre approche nous amène à prendre en compte au sein de la même commande les spécifications de l'utilisateur (client) et les contraintes du procédé.

Par conséquent, ce rapport a la structure suivante. Dans le chapitre I, nous introduirons d'abord les concepts de Système Manufacturier Reconfigurable et de processus de reconfiguration. Ce deuxième concept nous permettra notamment de présenter les fonctions couramment mises en œuvre pour permettre la reconfiguration dynamique. Dans un second temps, nous intéresserons plus particulièrement à la conception de la commande de coordination des systèmes manufacturiers. Nous ferons notamment un état de l'art sur les approches d'ingénierie de la commande et sur les approches de synthèse de la commande. Dans le chapitre II, nous définirons d'abord le Problème des Séquences Interdites de Transitions d'Etat qui est un problème de contrôle respectant des spécifications de type supervision. Pour résoudre ce problème, nous proposerons une approche basée sur le formalisme des Réseaux de Petri et nous proposons une nouvelle méthode de synthèse basée

sur la construction du Graphe d'Accessibilité Synchrone Contraint. Nous montrerons dans ce chapitre que nous pouvons utiliser la théorie des régions proposée par A. Ghaffari [GHA02b] pour réaliser cette synthèse.

Dans le chapitre III nous monterons comment la méthode présentée au chapitre II peut être appliquée à la synthèse RdP d'une commande de coordination modulaire et reconfigurable pour les systèmes manufacturiers. Nous distinguerons deux contextes d'utilisation de cette méthode : d'une part pour la conception de la commande du système de transport d'un procédé manufacturier, d'autre part pour la synthèse de contrôleurs déterministes pour la fabrication de produits spécifiques.

Dans le quatrième chapitre, nous présenterons la conception de l'outil NetMDI ayant pour but de montrer la faisabilité des propositions des chapitres II et III.

Chapitre I : État de l'art des approches pour la reconfiguration des systèmes manufacturiers

1. Introduction

Selon l'AFNOR [AFN 91], la reconfiguration est définie comme étant un processus de réorganisation physique ou logique des postes de travail en terme d'implantation ou d'allocation à des produits et /ou des activités.

Ce processus de réorganisation implique les deux niveaux d'un système : le procédé et la commande. Pour le procédé il se traduit par la modification des modes d'exploitation de certaines ressources du système afin de l'adapter aux nouvelles conditions de fonctionnement et poursuivre, ainsi, sa mission voire ses objectifs de production. En ce qui concerne la commande, la reconfiguration permet de modifier certains paramètres du programme en cours.

La reconfiguration est vue ici comme une technologie d'ingénierie qui traite la problématique de réaction des systèmes face à des nouvelles conditions de fonctionnement résultant de l'évolution rapide du marché. Le concept de Système Manufacturier Reconfigurable (ou SMR) inventé par l'université de Michigan en 1999 est vu comme l'une des réponses clés pour que l'industrie manufacturière puisse continuer à exister en Occident dans les années à venir ([KOR99]).

Dans ce chapitre, nous allons présenter le SMR en se basant sur la recherche de Y. Korean ([KOR99] et [KOR05]).

2. Système Manufacturier Reconfigurable

Actuellement, les Systèmes Manufacturiers Reconfigurables, concept inventé par l'université de Michigan en 1999 est un domaine de recherche bien établi. Un système manufacturier qui peut être reconfiguré de manière précise, rapide et peu coûteuse quand le marché exige un changement offre un avantage économique important aux entreprises. Le but du SMR d'après Y. Korean est de concevoir des systèmes avec des machines et des contrôleurs capable de répondre rapidement au coût minimum aux exigences des nouveaux marchés qui se caractérisent par des besoins variés et nécessitant une grande réactivité. Les SMR ont aussi comme objectif de s'adapter aux changements des environnements tant internes qu'externes qu'on subit les entreprises.

Définition 1 : un Système Manufacturier Reconfigurable (SMR) est un système conçu pour permettre des changements rapides de sa structure (aussi bien sa partie matérielle que sa partie logicielle), dans le but d'ajuster rapidement sa capacité de production et ses fonctionnalités face aux changements brusques du marché et des changements intrinsèques du système ([KOR05]).

Le développement d'une théorie sur les SMR et leur exploitation ne cesse d'évoluer selon les besoins qui deviennent de plus en plus nombreux et variés dans les entreprises qui vivent dans un contexte industriel caractérisé par des changements fréquents et imprévisibles du marché. Comme exemple de tels changements du marché, nous citons l'introduction de fréquence d'évolution de nouveaux produits, les produits existants, les grandes fluctuations dans la demande de produit et l'amalgame de produit, le changement dans les réglementions du gouvernement (la sécurité et l'environnement) et le changement dans la technologie des processus. Pour vivre dans cette nouvelle situation manufacturière, les entreprises doivent réagir rapidement et avec le minimum de coût aux changements. Elles doivent être dotées d'un système de production robuste et flexible pour s'adapter facilement à la fluctuation de la demande du produit dicté par le nouvel environnement compétitif.

La plupart des entreprises manufacturières utilisent un portfolio de Systèmes Flexibles de Production Manufacturière (ou SFPM) et de chaînes de production dédiée.

Les chaînes de production dédiée, ou lignes de transfert, sont fondés sur des processus automatisés fabriquant des pièces en grande série. Chaque chaîne dédiée est typiquement créée pour produire une seule pièce à un taux élevé de production accompli par le fonctionnement de plusieurs machines-outils simultanément dans des stations de machines. Quand la demande du produit augmente, le coût par pièce diminue relativement. Les chaînes de production dédiée sont économiques tant que la demande dépasse l'offre et ils peuvent fonctionner à des cadences de production élevées. Cependant, actuellement la concurrence mondiale exige des systèmes produisant des petites séries pour répondre à des contraintes de diversité, de productivité et de qualité. Ces systèmes sont appelés des Systèmes Flexibles de Production Manufacturière. (SFPM).

Ces systèmes sont un compromis entre, d'une part les chaînes de production de type lignes de transfert automatisées à très haute productivité, mais conçu pour des productions de grande série, et d'autre part des installations manuelles à très faible automatisation permettant une production unitaire.

La réponse économique aux changements du marché selon le SMR demande que la nouvelle méthode de production combine non seulement la flexibilité des SFPM et le rendement élevé des chaînes de production dédiées mais aussi elle nécessite aussi la réactivité et l'efficace.

Pour répondre aux variations continues des demandes des utilisateurs, des systèmes peu coûteux sont conçus avec des ressources réglables et avec une grande flexibilité. Ces systèmes permettent l'utilisation de plusieurs outils simultanément.

Comment peut-on construire un SMR ? Les systèmes reconfigurables doivent être conçu au départ pour être configurables et doivent être créés en utilisant de modules de hardware et software qui peuvent être intégrés rapidement et fidèlement sinon le processus de reconfiguration va être long et peu pratique. On présente six caractéristiques essentielles expliquées dans [KOR05] comme suit :

Modularité : la décomposition des fonctions et des besoins opérationnels en unités quantifiables qui peuvent être transférées en schémas de production alternés pour réaliser l'arrangement le plus optimal qui correspond à un ensemble de besoins.

Intégrabilité : la capacité d'intégrer des modules rapidement par un ensemble d'interfaces mécaniques, informationnelles et de commande qui offrent en même le temps l'intégration et la communication.

Adaptabilité : la capacité à 'adapter système de production et les machines pour assurer de nouveaux besoins avec une famille de produits similaires.

'Scalabilité' : la faculté à changer facilement la capacité de la production existante en réarrangeant le système de production existant et/ou changer la capacité de production des composants reconfigurables (e.g. machines) dans ce système.

Convertibilité : la capacité à transformer facilement les fonctionnalités des systèmes, machines et commandes existants pour les adapter à de nouveaux besoins.

Diagnosabilité : la capacité à lire automatiquement l'état courant du système et des commandes de manière à détecter et diagnostiquer les causes à l'origine des défaillances, et par conséquent, corriger rapidement les défauts opérationnels.

Quand ces caractéristiques sont introduites dans la conception du système, le haut degré de reconfigurabilité est accompli. La modularité, l'intégrabilité, la scalabilité et la diagnosabilité réduisent le temps de la reconfiguration et l'effort de la reconfiguration. L'adaptabilité et la convertibilité réduisent le prix.

En résumé, les SMR sont des systèmes réactifs dont la capacité de production est ajustable aux fluctuations de la demande du produit et dont les fonctionnalités sont adaptables aux nouveaux produits.

3. Modèle fonctionnel des processus de reconfiguration

Le processus de reconfiguration est un processus de réorganisation matérielle et/ou logicielle du système. L'objectif de cette réorganisation est de pouvoir assurer la production en réalisant un compromis entre les objectifs de production et l'état du système. Ce processus de reconfiguration peut être déclenché par deux catégories d'événements liés soit aux produits soit aux ressources de production.

Un changement de production peut être relatif à la nature de la production, la qualité ou la quantité des produits. En effet, dans l'industrie manufacturière, les systèmes de production flexibles ont été conçus afin de répondre à la production de petites ou moyennes

séries de produits. Cela entraîne qu'il peut être nécessaire sur un horizon de production donné, de lancer la fabrication de produits qui n'avaient pas été ordonnancés. Cela n'est possible que si les ressources engagées dans la production ne fonctionnent pas à charge maximale ou si l'on peut engager de nouvelles ressources en production. Un changement de production peut également être relatif à la qualité des produits. L'exigence d'une qualité supérieure à celle initialement prévue peut nécessiter l'engagement de ressources de transformations capables de l'obtenir. Il en est de même pour la quantité dont les exigences peuvent varier en cours de production.

Dans tous les cas, ces changements peuvent induire l'ajout ou la suppression de certaines ressources matérielles (par exemple des machines ou des robots dans un système manufacturier ou l'utilisation de pompes, cuves ou vannes différentes dans un système continue) par rapport à l'ensemble de celles engagées dans la production en cours.

Les changements d'états des ressources de production sont quant à eux caractérisés par deux grands types : les défaillances et les réparations. En cas de défaillance(s), le processus de reconfiguration doit d'abord chercher à substituer la (les) ressource(s) défaillante par une autre ressource du système de production. L'objectif dans ce contexte est d'utiliser des redondances actives ou passives pour remédier à la défaillance. Les deux types d'événements susceptibles de déclencher un processus de reconfiguration ne sont pas nécessairement découplés. En effet, une défaillance de ressource peut entraîner un changement de production en raison de l'impossibilité à retrouver les capacités de production nécessaires dans les délais requis.

La mise en œuvre du processus de reconfiguration dépend de deux paramètres : l'événement déclencheur et les contraintes temporelles s'exerçant sur le système lorsque survient cet événement. Deux situations complémentaires peuvent être considérées : le cas du lancement d'une nouvelle production avec le système à l'arrêt et le cas d'une défaillance survenant sur un système en cours de fonctionnement.

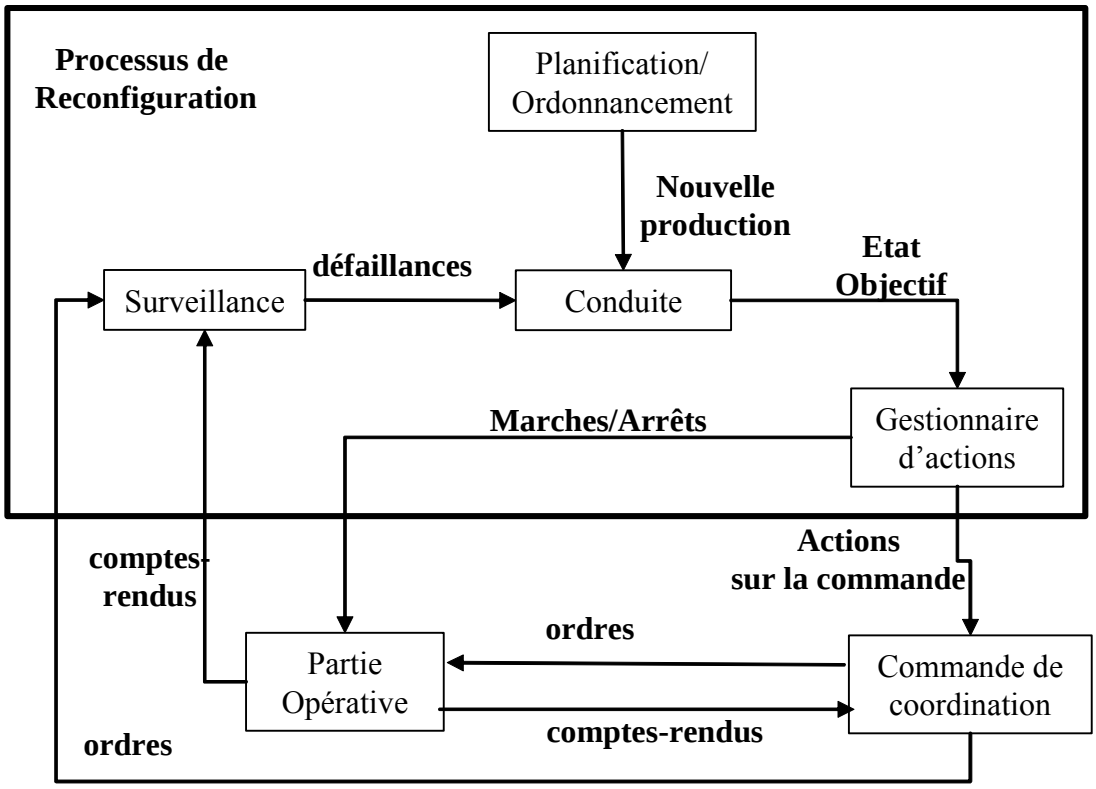


Figure 1. Représentation fonctionnelle du processus de reconfiguration

Une architecture de contrôle/commande de système manufacturier est construite autour d'un certain nombre de fonctions de contrôle (Figure 1) :

- Des fonctions informationnelles : la surveillance pour l'identification des défaillances et l'ordonnancement pour la spécification des pièces à produire ;
- Une fonction décisionnelle : la conduite,
- Une fonction opérationnelle : un gestionnaire de plans d'actions.

Définition 2 : Le processus de reconfiguration

Le processus de reconfiguration consiste à la boucle de réaction mise en œuvre à partir d'une fonction informationnelle qui déclenche la conduite. La conduite définit un état objectif dans lequel mettre le système en réaction à cet événement. Cet état objectif est atteint par la mise en œuvre d'un plan d'actions exécuté par la gestionnaire de plans d'actions.

La reconfiguration consiste à amener un Système de Production Automatisé de son Etat Courant EC vers un Etat Objectif EO. Par rapport aux objectifs de production et aux machines engagées en production ces deux états correspondent à des états permanents. Pour atteindre l'Etat Objectif, il est nécessaire d'évoluer étape par étape par le biais d'état transitoires. Pour définir formellement c'est qu'est la reconfiguration, nous commençons par définir la notion de configuration.

Définition 3 : Une configuration

Une configuration est un 2-uplé *(R, P)* avec R l'ensemble des ressources engagées en production et P l'ensemble des biens à produire sur l'horizon de production. Nous définissons par I=**({}, {}),** l'état initial du système dans lequel il n'y a aucune production en cours.

En cas de reconfiguration à la suite de l'occurrence d'une défaillance, l'Etat Objectif correspond à un Etat de Reprise (ER).

Définition 4 : Etat de Reprise

C'est un Etat Objectif qui permet de poursuivre la production en mode de fonctionnement nominal ou dégradé par rapport à l'Etat Courant, à la suite d'une défaillance. Le mode de fonctionnement est nominal si les ensembles de produits sont les mêmes dans les deux états : $EO(R_{EO},P)=EC(R_{EC},P))$; sinon l'état est dégradé. L'ensemble des ressources dans les deux états est forcément différent.

Remarque : la définition de l'état nominal que nous utilisons ici est une définition qualitative. En effet, en toute rigueur, il n'est pas suffisamment de maintenir la production qualitativement. Il faut également que les performances des nouvelles ressources engagées en production soient comparables à celles défaillantes.

4. Mise en œuvre de la décision de reprise

L'automatisation des systèmes de production est un enjeu économique vital. En effet, elle permet de gagner en productivité et d'accroître la compétitivité des entreprises travaillant

dans le domaine de la production de bien manufacturés. Cette automatisation nécessite l'élaboration de méthodologies intégrant toutes les phases de conception : de la spécification à l'exploitation de tels systèmes dans un contexte sûr de fonctionnement. Notre intérêt se porte sur la conception d'architectures de commande pour la conduite des Systèmes de Production à Événements Discrets tolérants aux fautes. L'objectif est de permettre la poursuite de la production en dépit des fautes pouvant entraver le bon fonctionnement du système. Dans ce cadre, en productique, il existe deux grandes approches : l'approche système et la synthèse de commande.

4.1 Méthodes d'ingénierie de la commande

Ces méthodes sont principalement développées en France. La plus part de ces méthodes ont pour origines les travaux effectués au LAAS à fin des années 80 [SAH87]. Elles sont caractérisées par la définition d'un cadre fonctionnel de la supervision et plus généralement du contrôle/commande : surveillance, gestion des modes, pilotage, reconfiguration … Ces fonctions sont souvent conçues à partir de modèle Réseaux de Petri (RdP) ou à base de connaissances (techniques d'intelligence artificielle). Dans la suite de cette partie, nous présentons un certain nombre d'approches pour la conclure avec la méthodologie CASPAIM développée au LAGIS/SED.

4.1.1. Approche du LAAS

L'objectif des travaux développés au LAAS est la conception d'une architecture de contrôle/commande modulaire, hiérarchique et distribuée pour les Systèmes Flexibles de Production Manufacturière (SFPM). Ces caractéristiques sont imposées par la complexité (en nature et en taille) des systèmes à modéliser. Ainsi dès 1987, A.E.K. Sahraoui proposait une structuration du contrôle/commande autour des fonctions de commande (de coordination) et de surveillance [SAH87], [SAH92]. Dans l'approche proposée, la surveillance était initialement structurée en :

- la détection : Elle est intégrée à la commande. Elle utilise la technique du chien de garde pour détecter toute violation de la commande par rapport à l'ordonnancement prévisionnel.

- le diagnostic : Il est basé sur des règles d'inférences. Il permet d'identifier les causes des défaillances.
- la décision/reprise [BON93] : La décision utilise également des techniques d'intelligence artificielle. Son rôle est de déterminer la manière de réagir aux défaillances. Lorsqu'elle est automatisée, cette réaction va s'appuyer sur des procédures de reprise prédéfinies, intégrées à la commande.
- l'interface opérateur,
- Les procédures d'urgence.

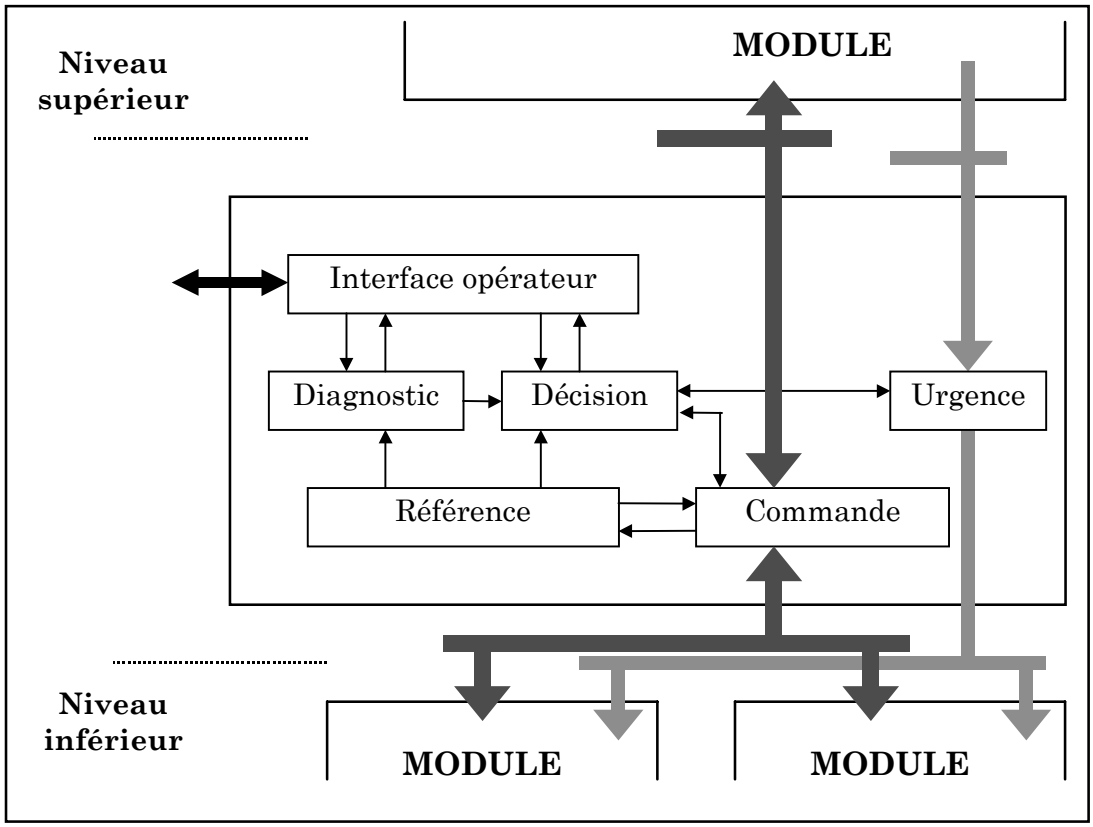


Figure 2. Structuration du contrôle/commande vue par le LAAS/OCSD [COM91]

A partir de 1991, cette architecture initiale va être complétée par l'introduction d'une fonction intermédiaire entre la surveillance et la commande : le module de référence. Il est bâti sur le modèle de référence qui représente le comportement normal du procédé, prenant en compte ses flexibilités potentielles uniquement restreintes par certaines contraintes de fonctionnement (contraintes de coopération, de synchronisation ou d'exclusion mutuelle). En fonctionnement normal avant chaque requête, la commande interroge le module de référence

afin d'en vérifier l'adéquation vis-à-vis des contraintes du procédé. Après émission d'un ordre, elle se met alors en attente d'un compte-rendu d'exécution. En cas de réception conforme (vis-à-vis d'une fenêtre temporelle) de ce compte-rendu, l'information est transmise au module de référence pour mettre à jour l'image du procédé qu'il maintient. La commande et le module de référence évoluent alors en parallèle. Dans le cas contraire, un symptôme est détecté et seul le modèle de référence est mis à jour. Ainsi, il constitue une véritable base de données qui va être exploitée par le diagnostic et la décision en cas de défaillance.

Les travaux de A. Chaillet-Subias [CHA95a] et E. Zamai [ZAM97] permettent par ailleurs de préciser le système d'information à mettre en œuvre afin de permettre l'intégration des différentes fonctions dans un contexte hiérarchisé. Les travaux actuels s'intéressent également à la distribution des traitements.

4.1.2. Approche du CRAN/LACN

Comme le LAAS, les travaux du CRAN concernent surtout la formalisation du concept de surveillance dans le cadre du contrôle/commande des SFPM. Dans ce cadre, les concepts de base qui expliquent la structuration proposée sont la notion d'Elément de Partie Opérative (EPO) [ALA86] et la notion de filtre de commande [LHO91].

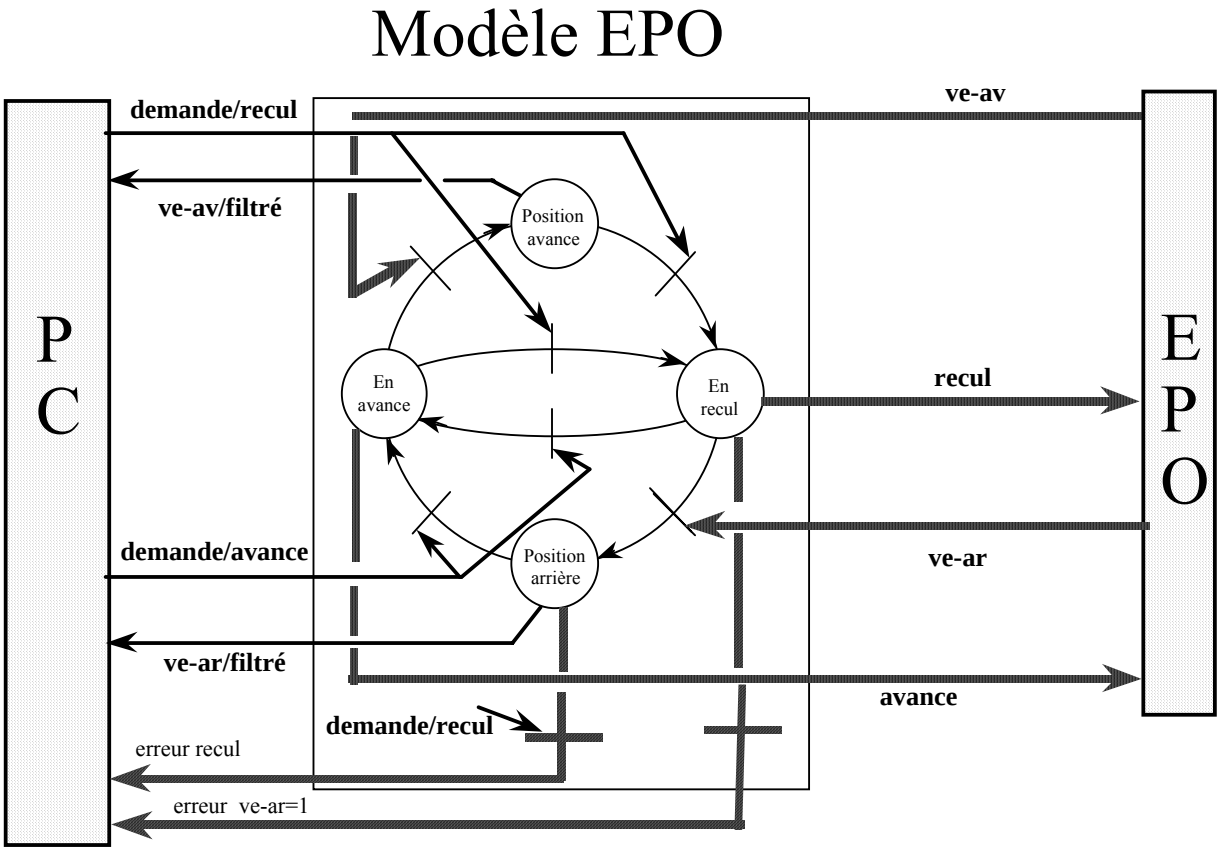


Figure 3 .Surveillance par prévision du comportement

Ainsi la Partie Opérative (PO) est décomposée en un ensemble d'EPO. Chaque EPO est caractérisé par un ensemble d'ordres et de comptes-rendus disjoints de ceux des autres EPO. L'idée est alors de se dire qu'il existe une "relation sémantique" entre les ordres et les comptes-rendus d'un EPO : c'est le comportement normal de l'EPO. Il est modélisé sous la forme d'un graphe d'états. Ces modèles comportementaux peuvent être alors positionnés entre la Partie Commande (PC) et la Partie Opérative (PO) pour faire de la surveillance par prévision de comportement (Figure 3).

La notion de filtre apporte beaucoup en fiabilité et sécurité. Dans [SFA89], les auteurs ont montré par une étude numérique comparative que les filtres permettent de réduire de près de 75% le temps de détection d'une défaillance. Ces résultats ont conduit au développement d'outils de CAO pour la manipulation de modèles de comportement en émulation ou en filtre (ADELAIDE, SPEX).

Plus récemment, les travaux de D. Gouyon [GOU04] ont proposé une approche mixte intégrant des techniques mais également de la synthèse de commande. Ces travaux sont basés

sur le concept d' «agile manufacturing ». L'idée est de décomposer la commande selon deux points de vue : celui des ressources de production et celui des produits. Les produits selon leurs besoins vont solliciter des opérations des ressources. Cette approche est extrêmement flexible et prometteuse pour la mise en œuvre de systèmes reconfigurables car ce sont les entités constituant le procédé (ressources et produits) qui portent-elles même une commande adaptable en fonction du contexte et de leur statut.

4.1.3. Approche du LAMIH

Les travaux du LAMIH concernent notamment la définition des modes d'exploitation d'un SAP (cf. la définition du MESAP dans [PAR92]) et la conception d'une architecture permettant sa reconfiguration. Intéressons-nous en particulier à ce deuxième point. L'objectif de la reconfiguration est la mise en œuvre de stratégies permettant de réagir sur le SAP après occurrence d'une défaillance de manière à conserver un fonctionnement optimal [MAB96].

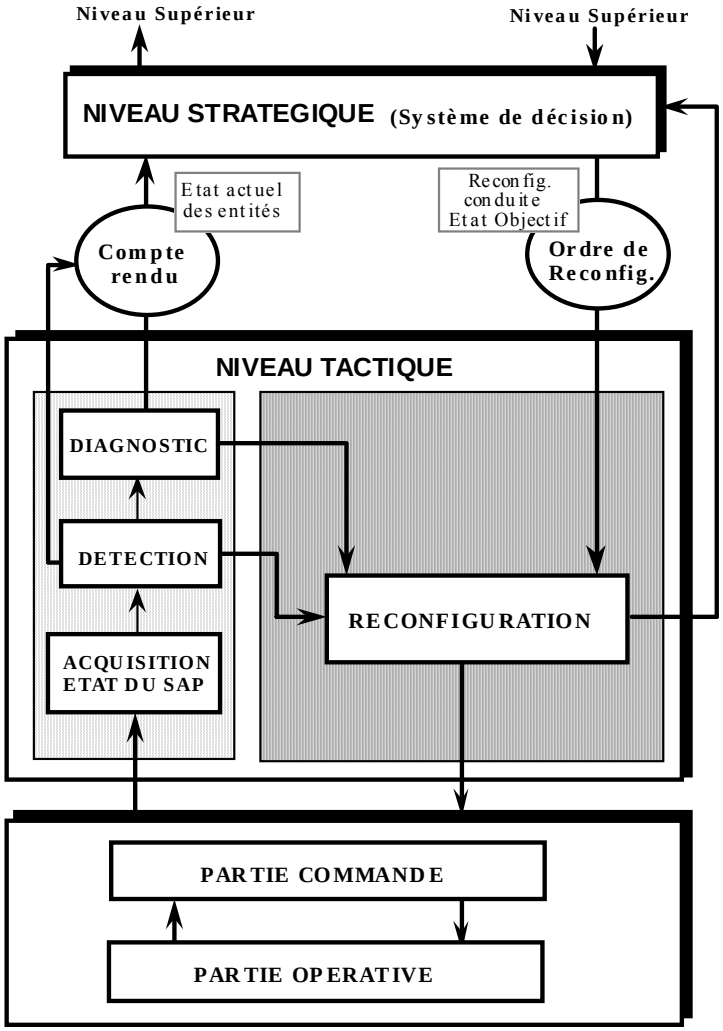


Figure 4. Structuration du contrôle/commande vue par le LAMIH

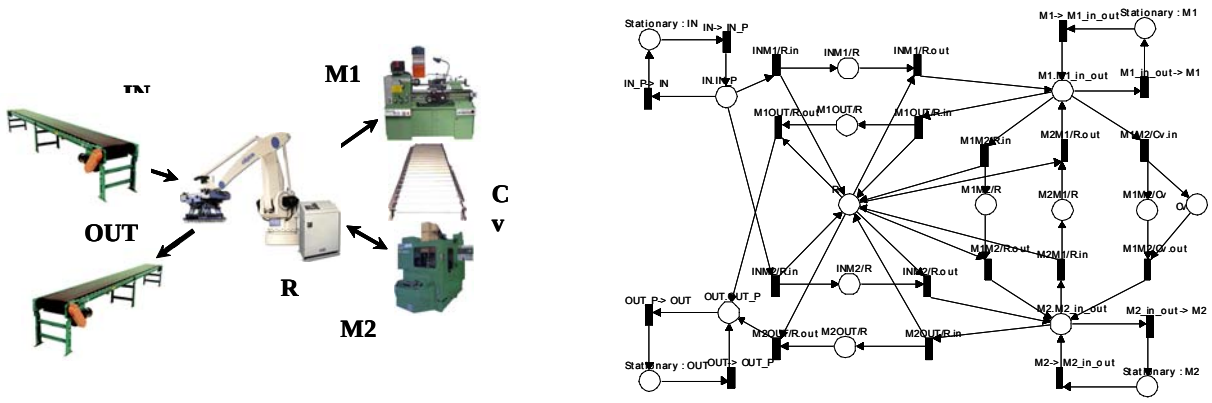
Pour cela, le processus est décomposé en deux niveaux (Figure 4) :

- un niveau stratégique qui va décider des réglages à opérer sur le flux des produits traversant les différentes ressources et des modifications concernant les modes d'exploitation du SAP ;
- un niveau tactique qui va surveiller le procédé commandé (fonctions acquisition, détection et diagnostic) et mettre en œuvre les décisions élaborées par le niveau stratégique (reconfiguration).

4.1.4. Approche du LESTER

Le LESTER propose une approche de type ingénierie de la commande pour la conception et l'évaluation d'architecture de systèmes reconfigurables. Leur approche

s'applique aussi bien à des systèmes d'électronique qu'à des systèmes manufacturiers (Figure 5A). Elle est d'abord basée sur une modélisation par composants du système que l'on veut étudier [BER05]. La modélisation par composant est réalisée à partir d'une modélisation UML des principales caractéristiques de chaque classe de systèmes. L'objectif de cette modélisation est double : permettre une analyse structurelle d'une architecture matérielle d'une part et évaluation ses capacités de tolérance aux fautes et de reconfiguration. Pour cela, le modèle de composant est transcrit en un modèle RdP (Figure 5B) plus approprié pour effectuer des analyses de propriétés. La transcription est réalisée en exploitant des techniques de transformation de modèles [LAM05a]. L'exploitation du formalisme UML pour la description des classes de composants permet également de



A B

Figure 5. Modèle RdP d'un système obtenu à partir d'une modélisation de ses composants

Les travaux actuels du LESTER ont pour objectif de permettre également la génération automatique du code de commande implantable sur des calculateurs industriels [LAM05b].

4.1.5. Approche du LAGIS

4.1.5.1 Architecture du contrôle/commande

La méthodologie CASPAIM (Conception Assistée de Système de Production Automatisé dans l'Industrie Manufacturière) est développée au LAGIS depuis pratiquement

deux dizaines d'années. Elle a pour objectif de proposer une architecture de contrôle/commande permettant le contrôle du système de production aussi bien en mode de fonctionnement nominal qu'en mode dégradé suite à l'occurrence d'une défaillance. Pour atteindre cet objectif, nous proposons une architecture composée de 2 fonctions hors-ligne et de cinq fonctions en lignes (Figure 6). Les fonctions hors-lignes sont la maintenance et la planification/ordonnancement. Le rôle de la fonction maintenance est d'informer la planification/ ordonnancement et la gestion des modes des ressources qui peuvent être engagée lors du prochain horizon de production. Les ressources en pannes et les ressources qui doivent faire l'objet d'une maintenance systématique sont rendues indisponibles. La fonction planification/ordonnancement utilise l'ensemble des ressources disponibles pour établir l'ordonnancement des pièces qui doivent être produites sur l'horizon. La méthodologie CASPAIM considère deux types de conduites des systèmes manufacturiers. Une conduite réactive ou une conduite cyclique. En cas de conduite réactive l'ordonnancement prévisionnel se traduit en pratique par la communication à la fonction pilotage de la séquence d'entrée dans le système des produits et des ratios de production par type de pièces devant être produites par chaque machine. En raison de l'occurrence de défaillances, ce scénario idéal ne peut pas être garanti. Aussi les fonctions en ligne doivent adapter la commande à poursuite de la production en mode de marche dégradé. La principale fonction en-ligne est la commande séquentielle. Cette fonction a pour objectif de définir les commandes permettant de transformer un produit brut en un produit fini. Elle est composée de contrôleurs qui doivent être paramétrées pour réaliser un compromis entre les objectifs de production donné par l'ordonnancement et l'état des ressources de production. C'est aux fonctions gestion des modes de marches et pilotage qu'il incombe de faire ce paramétrage. La gestion des modes effectue un paramétrage à moyen terme en paramétrant la commande par rapport au mode de marche courant. Le pilotage effectue un paramétrage en temps réel en interprétant les ratios de production par ressource en fonction de l'état du système.

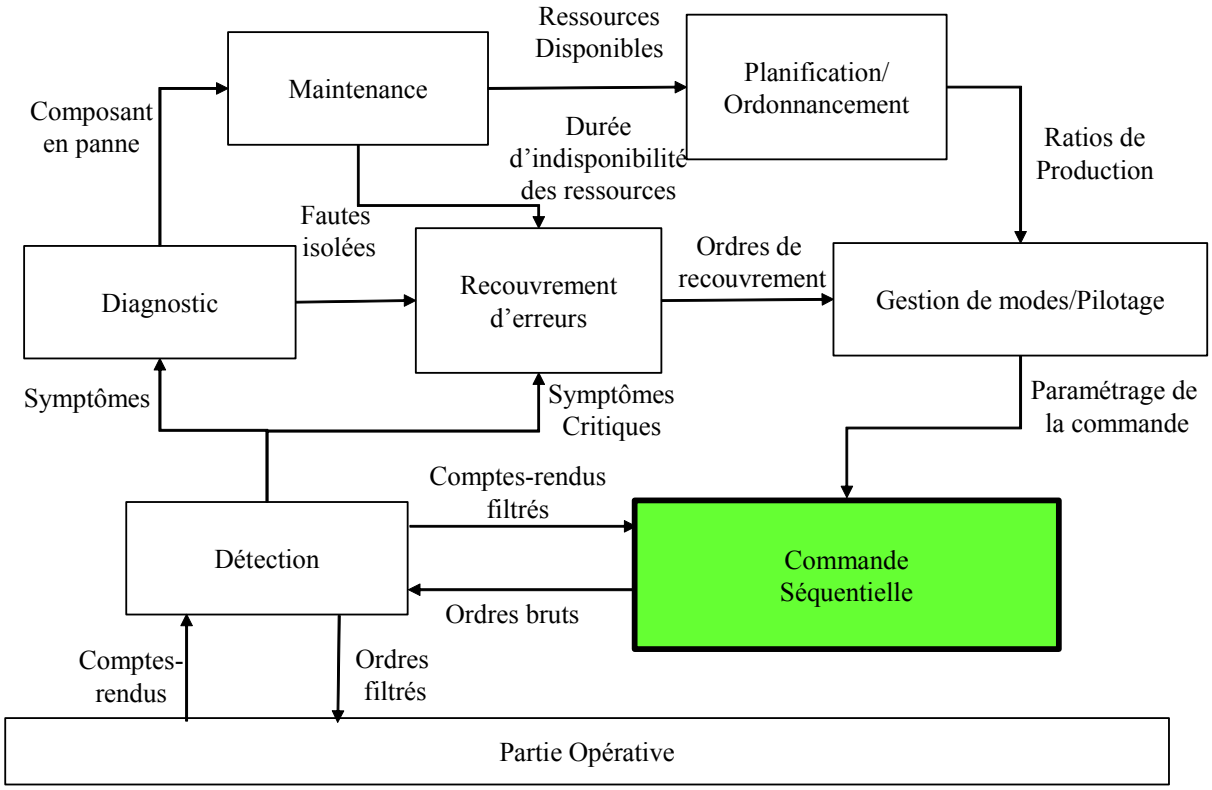


Figure 6. Architecture fonctionnelle du contrôle/commande selon la vision du LAGIS

La fonction de détection est positionnée en filtre entre la commande séquentielle et le procédé afin d'assurer que les ordres de commande et les comptes-rendus sont cohérents. En cas d'incohérence un symptôme est transmis la fonction diagnostic afin de permettre d'isoler la faute [TOG03]. Une fois la faute diagnostiquée, il faut transmettre l'information à la fonction recouvrement d'erreurs afin qu'elle décide comment traiter cette défaillance. Plusieurs traitements sont envisageables comme le gel, la reprise et la reconfiguration [SAH87]. Dans les chapitres suivant de ce mémoire, nous nous intéresserons en particulier à la reconfiguration du système notamment du point de vue traitement d'une défaillance.

Dans cette étude, nous nous intéressons en particulier à la conception d'une commande séquentielle qui soit reconfigurable. Dans ce cadre, l'approche du LAGIS est proche dans son principe de celle proposée par D. Gouyon du CRAN. En effet, elle est basée sur un découplage entre le point de vue ressource et le point de vue produit. Compte tenu de ce découpage, notre approche distingue deux types de contrôleurs : les gammes opératoires qui servent au contrôle des séquences d'opérations appliquées pour l'obtention d'un produit et les Graphes de Coordination des Ressources Complexes (GCRC) qui mettent en œuvrent la

commande des ressources complexes. L'objectif de ce découplage est d'abord de réduire la complexité inhérente à une prise en compte globale de l'ensemble des problèmes à traiter. Notamment lors d'une approche globale, la prise en compte de la flexibilité offerte par les ressources complexes peut engendrer une combinatoire importante au niveau des différents modèles.

Définition 5 : Ressource complexe

Une ressource complexe est une ressource constituée d'au moins deux zones opératoires command ables.

Un exemple typique de ressources complexes est les convoyeurs souvent utilisés pour le transport des produits dans un système manufacturiers. Un convoyeur est composé de plusieurs zones opératoires sur lesquels des synchronisations peuvent être réalisées avec d'autres organes de transport comme les robots de chargement/déchargement. La grosse difficulté dans le cas d'un convoyeur est le contrôle des différents transferts de produits qui se font en parallèle et en empruntant des sections communes devant être gérées comme des sections critiques.

Un autre exemple de ressources complexes est le cas des machines outils à commandes numériques possédant plusieurs broches (tours multibroches) ou plusieurs tables de travail (cas des centres d'usinages à tables multiples).

Le découplage nous sert également pour concevoir une commande modulaire, condition nécessaire pour obtenir un système reconfigurable.

En pratique, pour réaliser ce découplage, nous nous appuyons sur deux caractéristiques fortes de notre approche : l'utilisation des Réseaux de Petri Colorés (RdPC) et l'exploitation de la technique du client/serveur.

4.1.5.2 Les Réseaux de Petri Colorés

Un Réseau de Petri Coloré est un 5-tuple RdPC=<P, T, Coul, Csec, W> où :

- P est un ensemble fini de places,
- T est un ensemble non vide mais fini de transitions
- Coul est un ensemble fini de couleurs,
- C_{sec} est un sous-ensemble de fonctions de couleurs qui associent à chaque place et chaque transition un sous-ensemble de Coul :

$$C_{\text{sec}} : P \cup T \rightarrow P(C_{oul})$$

- W est la fonction d'incidence tel que chaque élément W(p,t) est lui-même une fonction :

$$W(p,t) : C_{\text{sec}}(\text{t}) \times C_{\text{sec}}(\text{p}) \rightarrow N$$

avec N l'ensemble des entiers naturels

Un Réseau de Petri Coloré marqué est un doublet <N, M0> où :

- N est un RdPC
- M0 est un marquage initial tel que pour chaque place et à chaque couleur dans cette place est associé un nombre de marques :

$$M_0(p) : C_{\text{sec}}(p) \rightarrow N$$

Dans notre approche, la coloration sert dans les gammes opératoires à spécifier le type de l'opération à appliquer à un produit. Dans ce cas le jeton correspondant à un produit est représenté par un doublet de couleurs **<id, op>** où 'id' est l'identifiant du produit et 'op' l'opération qui doit être réalisée. Dans le cas, des contrôleurs du Graphe de Commande du Système de Transport (GCST), la coloration correspond à un doublet de couleurs **<id, dest>** où 'dest' représente la prochaine destination de la pièce. Cette destination correspond à l'un des postes opératoires du système de production.

4.1.5.3 Technique client/serveur

Nous utilisons la technique client/serveur pour communiquer entre les différents contrôleurs RdPC. Cette technique est notamment très utile pour dès la phase de la conception, prendre en compte la possibilité de distribuer le code sur différents calculateurs industriels. L'idée est par exemple d'implémenter chaque gamme opératoire par un modèle RdPC dans lequel chaque place représente l'état du produit par rapport aux opérations de transformations et par rapport à sa localisation dans le système de production. Une opération de transformation d'une machine est requise par le biais d'une paire de places Requête/Acknowledge (Figure 7). La place 'Req' permet de mettre en oeuvre une coordination asynchrone avec le contrôleur de la machine. Comme la machine possède plusieurs opérations, il est nécessaire de préciser par la couleur 'op' l'opération à réaliser. Si d'autre par c'est une machine complexe pouvant accueillir simultanément plusieurs produits, ont doit également préciser l'identifiant 'id' de la pièce traitée. Cela permet lors du retour de l'acknowledge de savoir quelle est la pièce qui a fini d'être transformée.

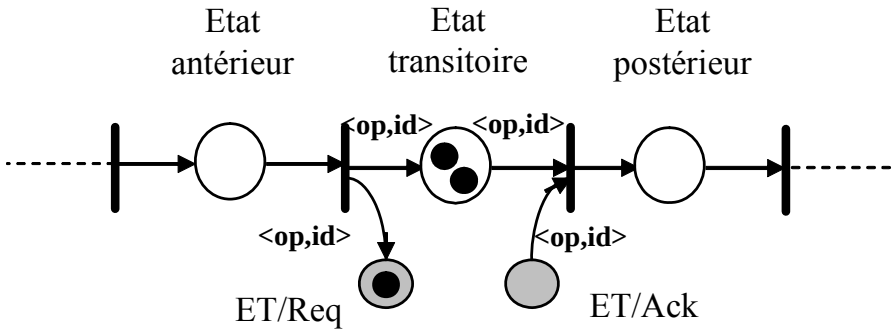


Figure 7. Coordination entre une gamme opératoire et le GCST à l'aide de la technique client/serveur

4.1.5.4 Processus de conception de la commande séquentielle

Selon le découplage présenté précédemment, notre approche de conception est décomposée en une phase de conception des gammes opératoires et en une phase de conception de la commande du système de transport (Figure 8).

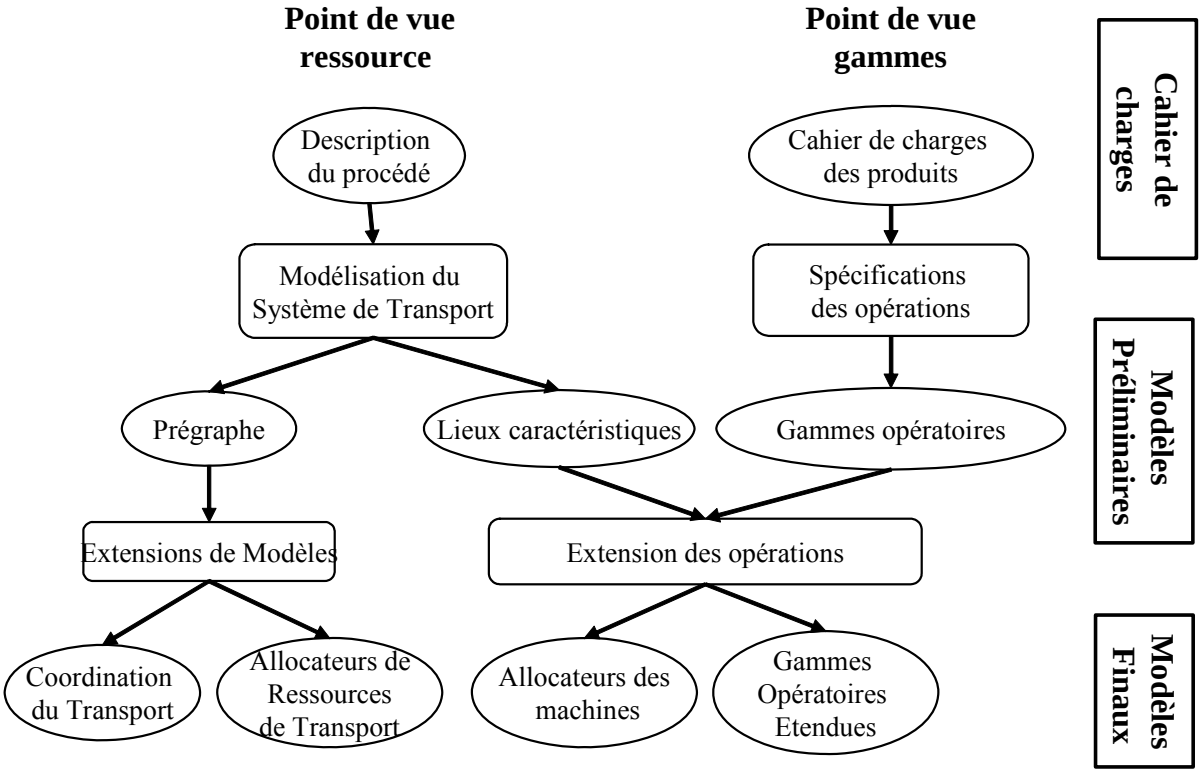


Figure 8. Synthèse du processus de conception de la commande séquentielle

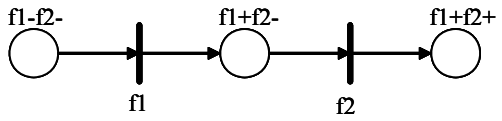
Notre approche consiste à partir du cahier de charges, pour construire les modèles de commande par raffinement successif selon une approche génie automatique. Ainsi dans un premier temps on construit des modèles préliminaires comme les gammes logiques (point de vue produit) et le prégraphe (Ressources point de vue). Ces modèles préliminaires sont ensuite raffinés en prenant en compte les contraintes induites par le procédé et l'architecture opérationnelle de commande (calculateurs industriels, réseaux locaux industriels, …).

4.1.5.5 Gammes opératoires

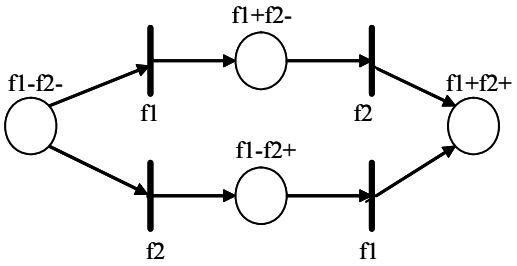
Elles décrivent les différentes opérations à appliquer à un produit brut pour obtenir un produit fini. Notre approche distingue trois types de gammes : les gammes logiques, les opératoires qualitatives et les gammes opératoires étendues. Dans la première étape du processus de conception on commence par établir les gammes logiques des produits.

Définition 6 : Gammes logiques

C'est un Réseau de Petri ordinaire qui modélise les différentes séquences des opérations de transformation permet de transformer un produit brut en produit fini. Chaque place du RdP modélise un état d'avancement du produit dans sa gamme d'usinage. Les transitions modélisent les opérations de transformation.



A. Cas d'un job shop



B. Cas d'un open shop

Figure 9. Gammes logiques

A ce stade, le concepteur ne doit pas prendre en considération la structure du procédé. Comme illustré par la Figure 9, il est toutefois important que le concepteur choisisse dès ce stade le type de système qu'il eut développé d'un point de vue flexibilité : un job shop ou un open shop. Le choix de l'open shop introduit des indéterminismes dans l'ordre des opérations nécessaires pour fabriquer un produit (Figure 9b).

Le second modèle utilisé est la gamme opératoire qualitative.

Définition 7 : Gamme opératoire qualitative

C'est un Réseaux de Petri ordinaire qui décrit les séquences opératoires à appliquer à un produit et sa localisation par rapport aux machines du système de production. Chaque place modélise à la fois l'état du produit de point de vue de sa gamme d'usinage mais également sa

position sur ou entre deux machines. Les transitions modélisent soit des opérations de transformation soit des méta-transferts entre machines.

Le terme méta-transfert signifie ici que les opérations de transferts peuvent être décomposées en des transferts plus élémentaires exécutés par des ressources de transfert du système de production. Ces transferts élémentaires ne sont pas représentés dans les gammes opératoires car ils sont en fait contrôlés par le Graphe de Coordination du Système de Transport.

Pour construire une gamme opératoire qualitative, on part de la donnée des gammes logiques et de l'ensemble des machines qui composent le système de production. Dans un premier temps, le processus de conception associe chaque opération de transformation d'une gamme logique avec des machines mettant en œuvre ces opérations. Quand plusieurs machines réalisent la même opération, on crée autant de place que de machines candidates. Ensuite, il faut créer une place pour modéliser la localisation initiale du produit brut sur le stock d'entrée et sa localisation finale sur le stock de sortie. Au final, les places sont connectés les unes aux autres depuis la localisation initiale, à l'aide de transition modélisant soit des méta-transferts soit des opérations de transformation en respectant la règle 1.

Règle 1 : Deux places consécutives d'une gamme opératoire qualitative doivent différer soit au niveau de la localisation, soit au niveau de l'état d'avancement du produit dans sa gamme d'usinage.

L'intérêt des gammes opératoires qualitatives est qu'elles permettent très tôt de faire de la vérification de propriétés et de la validation de la conformité du modèle par rapport au cahier de charges.

L'étape suivante consiste à construire les gammes opératoires étendues. En fait, il s'agit de rendre les gammes opératoires qualitatives adéquates à une distribution de code. Nous nous appuyons pour cela sur la technique client/serveur présentée précédemment. Ces gammes utilisent les concepts de lieu caractéristique et de relation d'accessibilité, proposés par S. Amar [AMA92]

Définition 8 : Lieu caractéristique [AMA92]

Un lieu est caractéristique est un lieu de travail (machine ou lieu d'assemblage) ou un lieu permettant le transfert d'un produit entre deux systèmes.

Définition 9 : Relation d'accessibilité [AMA92]

Deux lieux d'un système de production automatisé sont dits en relation d'accessibilité directe, s'il est possible de transférer directement le produit d'un lieu à l'autre en utilisant qu'une seule ressource de transport. Les relations d'accessibilité indirectes se déduisent par transitivité.

Notons qu'on distingue les relations d'accessibilité internes à une ressource complexe comme un convoyeur et les relations d'accessibilités externes. Les relations d'accessibilité permettent les transferts entre des lieux appartenant à des ressources différentes. Dans les gammes opératoires étendues, les méta-transferts correspondent à des relations d'accessibilités externes qui sont souvent indirectes.

Définition 10 : Gamme opératoire étendue

C'est un Réseaux de Petri Coloré (RdPC) qui modélise les différentes séquences d'opérations de transformation et de méta-transferts entre lieux caractéristiques en relations d'accessibilité externe qui permettent de faire passer un produit de l'état brut à l'état fini. Les couleurs des jetons sont alternativement des doublets <id, dest> ou <ip, op> en fonction de la nature de la place du modèle RdPC.

Une gamme opératoire étendue utilise des paires d'arcs Requête/Acknowledge pour demander des l'allocation de machines aux allocateurs de ressources ou pour demander des services aux ressources complexes. Par contre, elle ignore les détails de la mise en œuvre de ces services ce qui permet de les reconfigurer en cas de besoin, sans qu'il n'y ait d'impact sur la gamme opératoire étendue.

A titre d'illustration, considérons l'exemple de système manufacturier de la Figure 10.

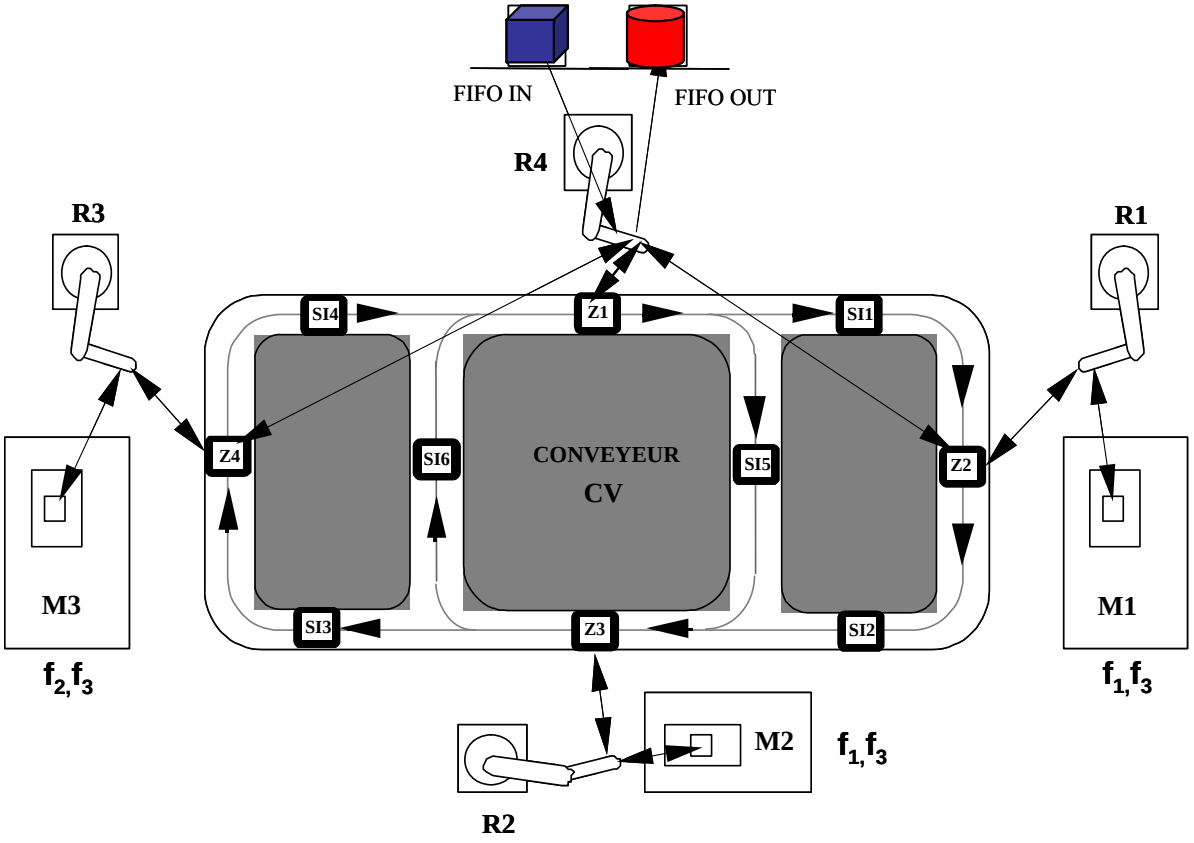


Figure 10. Exemple de système manufacturier [BER98]

Ce système est composé de 3 machines M1, M2 et M3 dont les opérations de transformation associées sont notées 'f_i' à côté de chaque machine. Nous avons donc une redondance pour certaine opération comme 'f3' qui est réalisée par les 3 machines. Le système de transport est composé d'un convoyeur central qui dessert des lieux de travail notés 'Zi' avec i [1..4] et des stocks intermédiaires notés 'SIj' avec j [1..6]. Les tampons d'entrée et de sortie du système, les zones opératoires des machines et les lieux Z1 sont des lieux caractéristiques. Les zones 'SIj' du convoyeur ne sont pas des lieux caractéristiques. Les flèches sur le schéma symbolisent les relations d'accessibilité. Le transfert de FIFO_IN vers Z1 est une accessibilité externe directe. Le transfert 'Z1' vers 'IS1' est une accessibilité direct mais interne à la ressource complexe qu'est le convoyeur. Le transfert 'Z1' ers 'M1' est une accessibilité externe indirecte mettant en œuvre deux ressources de transport.

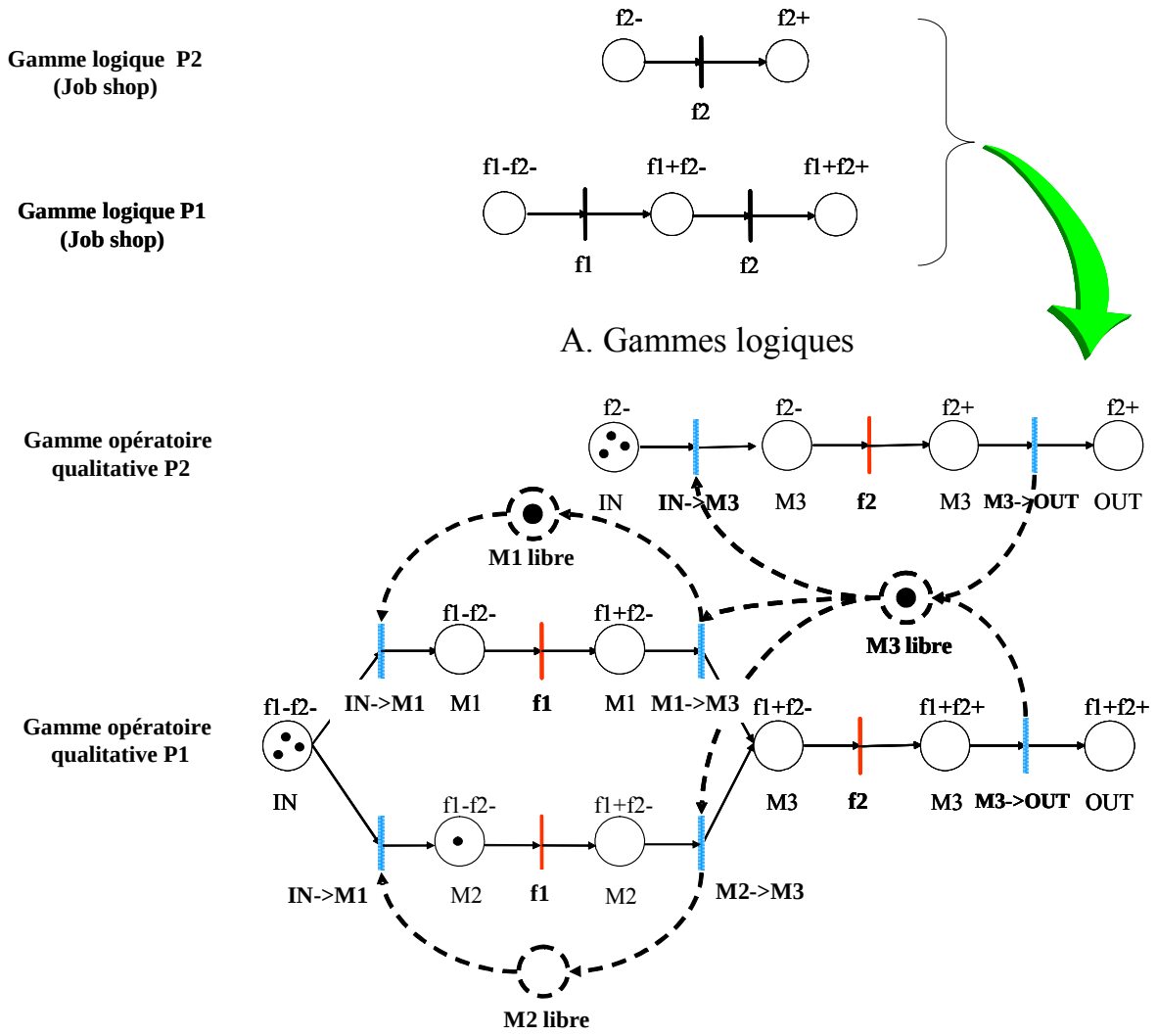


Figure 11. Gammes logiques et gammes opératoires associées au système manufacturier de la Figure 10.

La Figure 11 illustre les gammes relatives à deux types de pièces que l'on peut produire sur notre système de production illustratif. Elle montre comment la prise en compte progressive des différentes exigences du cahier de charges se traduit par une complexité accrue des modèles. Par exemple, on voit apparaître des indéterminismes dans la gamme opératoire P2 en raison de la redondance offerte par le procédé pour la réalisation de l'opération 'f1'. La complexité aurait été plus importante si on avait supposé que notre système fonctionne en Open Shop et non en Job Shop.

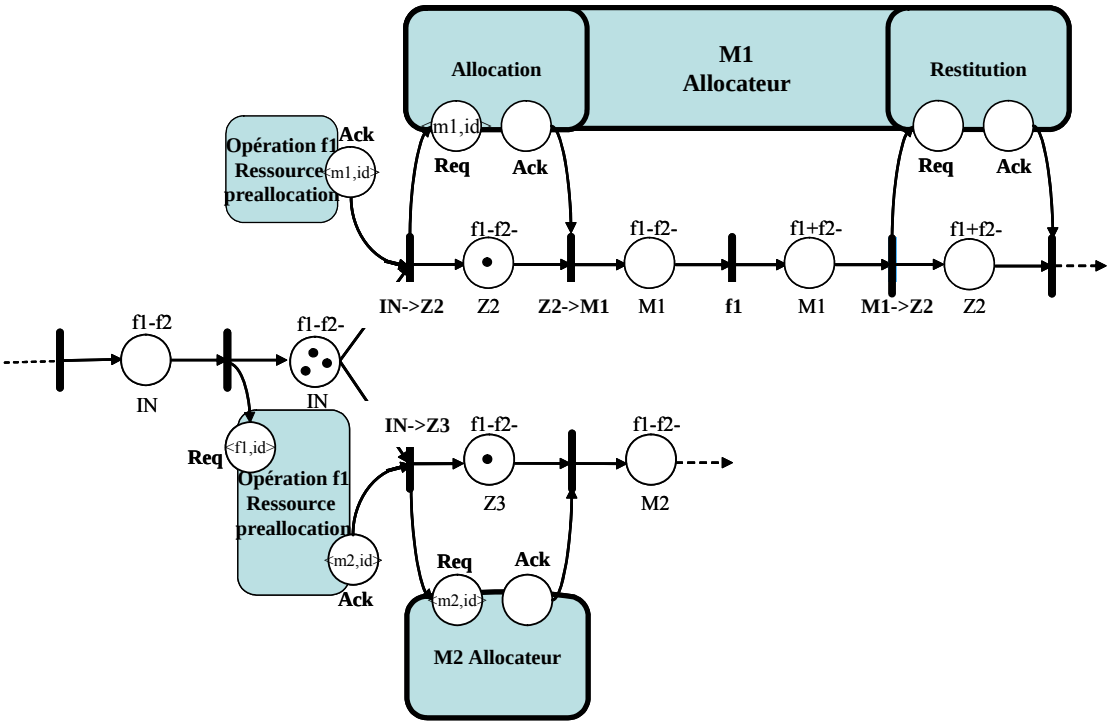


Figure 12. Extension progressive des gammes opératoires qualitatives

La Figure 12 illustre l'extension des gammes opératoires pour prendre en compte les nécessités d'allocation des ressources. Au niveau des gammes opératoires, on ne s'intéresse qu'à l'allocation des machines. Les demandes d'allocation des ressources de transport sont émises par le Graphe de Commande du système de Transport. Notons sur la figure précédente que pour obtenir les gammes opératoires définitives, il est encore nécessaire d'étendre les transitions du type 'IN->Z2' (lire du buffer 'IN' vers le lieu caractéristique 'Z2') en utilisant la prive client serveur présentée à la Figure 7.

4.1.5.6 Conclusion

Dans la première partie du chapitre III, nous montrerons comment le Graphe de Commande du Système de Transport (GCST) peut être construit à l'aide d'une approche de synthèse basée sur les RdPs. Dans la dernière partie du chapitre III, nous reviendrons sur une démarche systématique de construction des allocateurs de machines. C'est une des contributions forte de ce travail.

4.1.6. Conclusion sur les approches d'ingénierie du contrôle/commande

Cette partie nous a montré que les travaux proposés dans le cadre de l'ingénierie du contrôle/commande s'intéressent tous à définir les différentes fonctions du système de contrôle/commande et à concevoir une architecture de mise en œuvre qui prenne en compte la complexité des systèmes manufacturiers. Déjà, le fait de vouloir prendre en compte toutes les fonctions est en soit complexe car elles n'ont pas tous les mêmes exigences et nécessitent donc souvent des formalismes adaptés. Ainsi, même si d'une manière générale les Réseaux de Petri sont le formalisme le plus généralisé utilisé pour modéliser les comportements séquentiels, on est souvent obligé de faire appel à d'autres formalismes. Notamment, les tâches de diagnostic sont souvent développées en se basent sur des techniques d'intelligence artificielle [TOG01] ou des formalismes de type automates [LAF02]. On commence toutefois à voir apparaître des approches basées également sur les RdP [GHA05], [ASH04].

L'avantage de cette structuration fonctionnelle est donc qu'elle permet d'étudier chaque fonction séparément en utilisant les outils les mieux adaptés. Ainsi, selon les fonctions, différents formalismes sont exploités : le grafcet, les RdP, le graphe d'état, les systèmes à base de connaissance, [DAN00] … Cette richesse entraîne également l'une des faiblesses de cette approche. En effet, l'hétérogénéité des formalismes entraîne une difficulté majeure à valider ensemble les modèles développés pour chaque fonction. Souvent, seules des validations partielles et des simulations de l'ensemble pour vérifier un comportement global cohérent sont accessibles. Ce problème de la validation est rendu encore plus complexe lorsqu'on envisage une mise en œuvre distribuée.

Dans la suite de ce mémoire, nous nous focalisons sur l'utilisation des RdP associés aux méthodes de synthèse de ma commande.

4.2 Synthèse de commande

Les approches de synthèse de commande sont caractérisées par une prise en compte séparée des spécifications du cahier de charges et des contraintes du procédé. Plus formelles, elles permettent a priori de mieux garantir le respect du cahier de charges. En productique, elles sont utilisées pour la conception de la commande de coordination. Des équipes de

recherche commencent à proposer des extensions de la synthèse de commande permettant de surveiller et superviser les SED. Toutefois, l'intérêt pour le monde industriel de la synthèse de commande reste limité en raison de la complexité de l'approche pour traiter des procédés de grande taille.

Dans ce cadre, plusieurs travaux de recherche portant sur la synthèse de la commande [RAM87a] [WON00] [WON03] [ALP02] [BOU04] [GHA02c] [HOL90] seront présentés dans ce suit.

4.2.1. Contrôle par supervision

En 1987 Ramadge et Wonham ont proposé la théorie du « supervisory control » (contrôle par supervision) pour modéliser la commande des systèmes à événements discrets [RAM87a]. Cette théorie se fonde sur les automates à états finis et les langages formels [RAM87a] [WON00] [WON03]. Dans cette approche, deux modèles de base sont considérés : le modèle du procédé et le modèle de la spécification des utilisateurs. Considérons l'exemple illustratif de la figure 2 qui représente le modèle du procédé et le modèle de la spécification correspondant à un système producteur-consommateur.

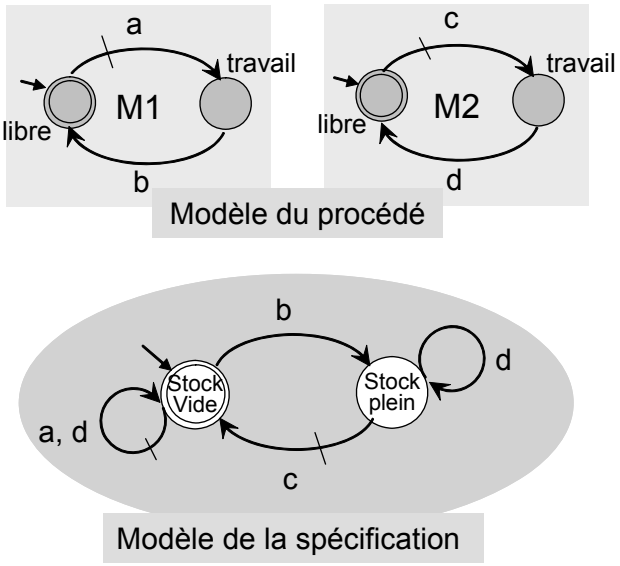


Figure 13. Modèle du procédé et modèle de la spécification

L'objectif de la théorie du contrôle par supervision est de construire le contrôleur le plus permissif qui respecte les spécifications de l'utilisateur. Pour accéder à ce but, les auteurs proposent de considérer deux types d'événements : les événements contrôlables et les événements incontrôlables. Par exemple, dans la Figure 13, 'a' et 'c' sont des événements contrôlables et 'b' et 'd' sont des événements incontrôlables.

La théorie de base propose d'abord d'élaborer le modèle global du procédé considéré en effectuant le produit synchrone des modèles partiels correspondant à chacune des ressources du procédé. Ensuite, le contrôleur est obtenu par l'intersection du modèle de la spécification avec le modèle global du procédé (Figure 14).

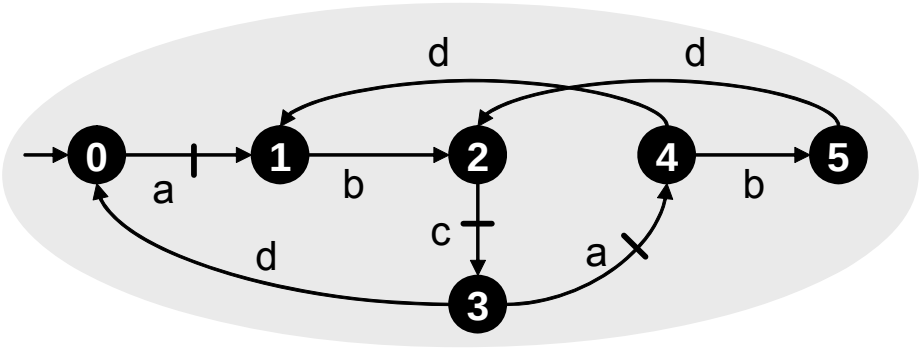


Figure 14. Modèle automate correspondant au contrôleur du producteur-consommateur

Une limite importante de l'approche de Ramadge et Wonham réside dans l'utilisation du formalisme des automates à états finis. En effet lorsque les modèles partiels des ressources considérées partagent peu d'événements communs, la composition synchrone peut entraîner une explosion combinatoire notamment dans le cas d'un système composé de nombreuses ressources évoluant en parallèle. Afin de minimiser le phénomène d'explosion combinatoire deux familles de solutions ont été étudiées : la réduction de la taille du modèle du procédé ou de celui de la spécification ([ALP02]) et le développement d'approche de modélisation modulaire ([WON88] et [QUE00]).

4.2.2. Modélisation utilisant les Réseaux de Petri et les Automates finis

Pour modéliser les Systèmes à Événements Discrets (SED), une alternative aux automates à états finis réside dans les Réseaux de Petri (RdP). En effet, les RdP permettent de modéliser facilement différentes classes d'un système dont les comportements sont notamment caractérisés par des évolutions parallèles et des synchronisations. Aussi, afin de minimiser le phénomène d'explosion combinatoire, certains auteurs proposent une approche mixte basée sur l'utilisation des RdP pour la modélisation du procédé et l'utilisation des automates pour formuler les spécifications. Le contrôleur est d'abord construit sous forme d'automate avant d'être traduit sous la forme de places de contrôle des transitions correspondantes aux événements contrôlables dans le RdP initial [ALP02]. Bien qu'intéressantes sur le plan théorique, ces approches nécessitent en pratique une réduction des modèles ainsi générés et un développement d'algorithmes performant permettant la projection du contrôleur au sein du modèle du procédé. Nous appliquons les travaux d'Alpan sur l'exemple du producteur-consommateur. La Figure 15 modélise sous forme de RdP le comportement de chacune des machines considérées de manière indépendante. La signification de chaque place et de chaque transition est précisée par le Tableau 1.

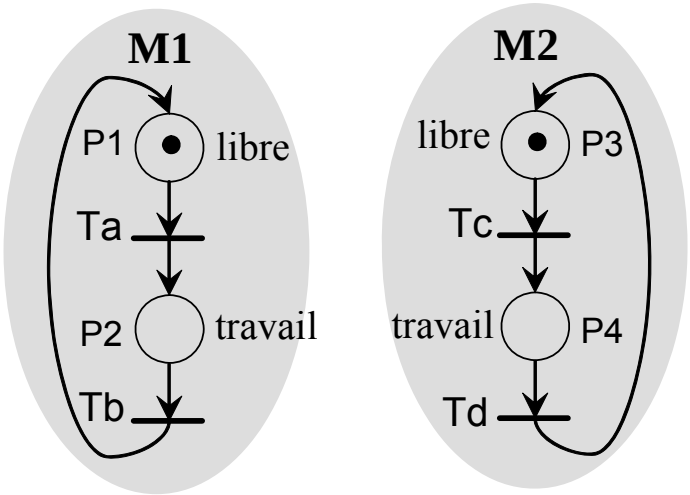


Figure 15. Le modèle du procédé du producteur-consommateur

Tableau 1. Les tâches de modèle du procédé RdP

Place	Condition	Transition	Événement	État
P1	Machine1 est disponible.	Ta	Machine1 commence à travailler	contrôlable
P2	Machine1 travaille.	Tb	Machine1 termine un travail.	incontrôlable
P3	Machine2 est disponible.	Tc	Machine2 commence à travailler	contrôlable
P4	Machine2 travaille.	Td	Machine2 termine un travail.	incontrôlable

Le modèle RdP du procédé que nous noterons ' N_P ' donne une représentation compacte du comportement en boucle ouverte du procédé. Afin d'obtenir un comportement conforme au cahier de charges de l'utilisateur, il est nécessaire de compléter ce modèle en analysant le comportement du système non contrôlé. Cette analyse permet ainsi de traduire le cahier de charges sous la forme de spécifications plus formelles pouvant être codées à l'aide du formalisme des automates à états finis. Pour retrouver le cadre de la théorie de base du « supervisory control », il faut alors traduire le comportement en boucle ouverte du procédé sous forme d'automates à états finis. L'approche naturelle consiste donc à établir le graphe d'accessibilité $R(N_P, M_0)$ correspondant au modèle RdP ' N_P '. Lorsque le modèle du procédé est représenté par un RdP de taille importante, cette approche « naïve » entraîne une explosion combinatoire liée au nombre d'états du graphe d'accessibilité. Afin de réduire cette combinatoire, [ALP02] propose de contourner le problème en ne construisant pas de manière explicite le graphe d'accessibilité. L'idée développée est l'exploitation de la connaissance liée aux P-invariants du modèle RdP.

Considérons de nouveau l'exemple du producteur-consommateur. Ses P-invariants sont obtenus par la résolution de l'équation $x^T C = 0$ avec C la matrice d'incidence de ' N_P '. Dans notre exemple, cette résolution nous conduit à déterminer les P-semiflots $x_1^T = [1 \quad 1 \quad 0 \quad 0]$ et $x_2^T = [0 \quad 0 \quad 1 \quad 1]$. Nous en déduisons donc les invariants de marquage données par les équations $\mu_1 + \mu_2 = 1$ et $\mu_3 + \mu_4 = 1$.

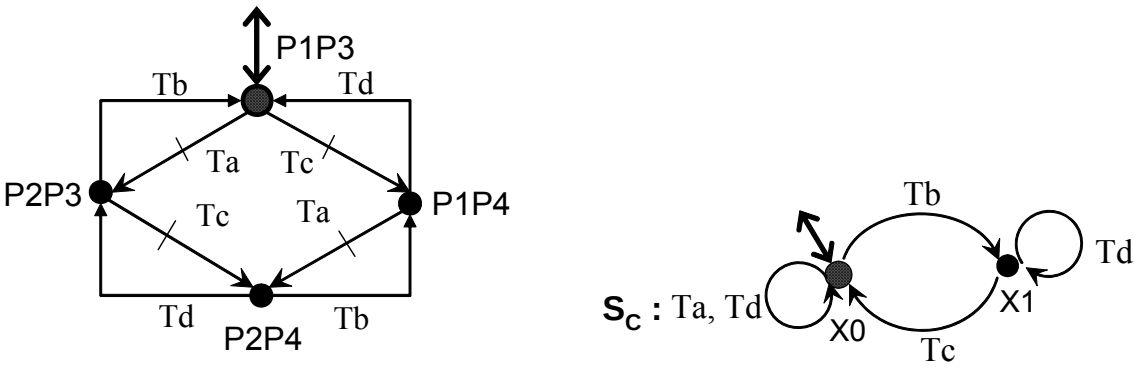
Une fois les P-invariants obtenus l'approche considérée consiste à effectuer un produit cartésien synchrone des places apparaissant dans ces invariants de marquage. Le produit réalisé est synchrone de manière à ne considérer que des combinaisons de places qui ne soient pas redondantes (on exclut les combinaisons relatives aux places figurant dans

plusieurs invariants). Ce modèle partiel est complété par la donnée des transitions permettant l'évolution d'une combinaison de places vers une autre (Tableau 2).

Tableau 2. Graphe d'accessibilité partiel obtenu par l'intermédiaire des P-invariants

	P1P3	P1P4	P2P3	P2P4
P1P3		Tc	Ta	
P1P4	Td			Ta
P2P3	Tb			Tc
P2P4		Tb	Td	

Le modèle résultant que nous appellerons 'G_m' pour graphe modifié est normalement un modèle réduit comparé à un graphe d'accessibilité standard comportant des états redondants (cf. [ALP02]). Ce n'est pas le cas dans l'exemple élémentaire du producteur-consommateur où 'G_m' est équivalent au graphe d'accessibilité (Figure 16.A). La Figure 16.B illustre la spécification automate qui consiste à n'autoriser la production d'une nouvelle pièce que si le stock est vide (état x0 actif). Dans la Figure 16, '→●' représente l'état initial et '●→' représente un état final (état marqué) du modèle automate. Donc, '↔●' représente l'état initial et final.



A. Graphe modifié (G_m) obtenu à l'aide des P-invariants

B. Modèle automate de la spécification

Figure 16. Graphe d'accessibilité modifié et modèle automate de la spécification

A partir du graphe modifié ' G_m ' et du modèle de spécification on peut appliquer la méthode classique du « supervisory control » telle que proposée par Ramadge et Wonham [RAM87a] [RAM87c]. A cet effet, après avoir complété les deux modèles par rapport à l'alphabet considéré, nous construisons le superviseur en effectuant l'intersection entre le modèle du procédé en boucle ouverte le modèle de spécification. Afin d'en faciliter la lecture, dans la figure 6 A. Nous renommons les états composés obtenus en utilisant les étiquettes ' S_i ', avec i = 0, …, 5. La fonction de contrôle relative au superviseur est donnée par le tableau correspondant de la figure 6A Dans ce tableau, l'étiquette 'γ' est associée aux événements contrôlables. Le modèle du superviseur et sa fonction de contrôle associée doivent être interprétés comme suit : quand le superviseur est dans l'état ' S_0 ' (respectivement dans l'état ' S_1 ') l'événement 'Ta' (respectivement 'Tb') est autorisé (γ=1) alors que l'événement 'Tc' n'est pas autorisé (γ =0). En effet, d'après la spécification de l'utilisateur la transition 'Tc' ne peut pas être franchie avant la transition 'Tb'. Notons que les événements incontrôlables ne peuvent jamais être inhibés.

Le principe de l'approche d'Alpan consiste ensuite à projeter le superviseur sous forme de places de contrôles dans le RdP modélisant le comportement du procédé en boucle ouverte. Afin de contrôler la taille du modèle projeté, Alpan propose des algorithmes de réduction du modèle en deux étapes :

- La première étape consiste à appliquer l'algorithme I d'Alpan (cf. Annexe B) selon le processus illustré par la figure 6 B. Soit ' S ' le modèle du superviseur. Formellement, il est présenté par un couple (S_a, Ψ) avec $S_a = (\Sigma, X, \xi, x_0, X_m)$ un automate fini et ' Ψ ' sa fonction de contrôle associée. Considérons les notations suivantes :

$\Psi_i^0 = \{x | \psi : x \times S_i \to 0\}$ désigne l'ensemble des événements qui sont invalidés à l'état ' S_i '.

$\Psi_i^1 = \{x | \psi : x \times S_i \to 1\}$ désigne l'ensemble des événements qui sont validés à l'état ' S_i '.

$\Psi_i^s = \{x | \ \delta(S_i, x) = S_i\}$ désigne l'ensemble des événements qui complètent la spécification de l'état ' S_i ' (selfloop).

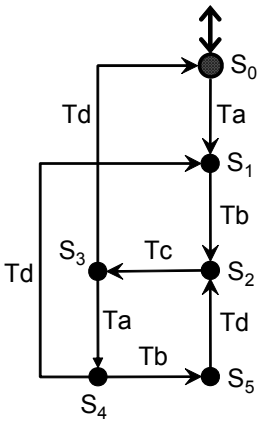
Notons $PN(N_p, S_a) = (P_p \cup X, T_p, I_p \cup I_s, O_p \cup O_s)$ RdP obtenu par la projection de l'automate fini ' S_a ' dans le modèle du procédé ' N_P '.

$I_s : X \times T_p \rightarrow \{0,1\}$ est l'application d'entrée des places de contrôle aux transitions du modèle du procédé en boucle ouverte et $O_s : T_p \times X \rightarrow \{0,1\}$ est l'application de sortie des transitions du modèle du procédé vers les places de contrôle implémentant le superviseur. $PN_c(N_p,(S_a,\psi))$ désigne le RdP obtenu après avoir ajouté la structure de contrôle ' Ψ ' à $PN(N_p,S_a)$.

L'idée de cet algorithme de réduction consiste à fusionner des places du superviseur qui ont un contrôle cohérent : la fonction de contrôle doit fournir la même sortie. La première étape de cet algorithme permet d'identifier les états incompatibles. Par exemple, les états ' S_0 ' et ' S_2 ' sont incompatibles en raison d'un contrôle incohérent vis-à-vis de l'événement 'Ta' (Figure 17.B).

- La deuxième étape consiste à fusionner les états compatibles en raison d'un contrôle cohérent et accessible depuis l'état initial par application de la fonction de transition 'δ'. Les états ' S_0 ' et ' S_3 ' sont compatibles donc, ils peuvent être fusionnées. La fonction de transition 'Td' permet d'atteindre les états ' S_1 ' et ' S_4 ' qui sont compatibles. Ces deux états peuvent, à leur tour, être fusionnés et ainsi de suite pour obtenir l'automate réduit de la figure 6 C.

A.



Etats	Ta : γ	Tb	Tc : γ	Td
S_0	1	-	0	-
S_1	-	1	0	-
S_2	0	-	1	-
S_3	1	-	-	1
S_4	-	1	-	1
S_5	0	-	-	1

B. Première étape de l'algorithme I d'Alpan

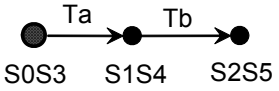
$\psi_0^1 = \{Ta\}, \psi_0^0 = \{Tc\}$
$\psi_1^1 = \{Tb\}, \psi_1^0 = \{Tc\}$
$\psi_2^1 = \{Tc\}, \psi_2^0 = \{Ta\}$
$\psi_3^1 = \{Ta, Td\}, \psi_3^0 = \varnothing$
$\psi_4^1 = \{Tb, Td\}, \psi_4^0 = \varnothing$
$\psi_5^1 = \{Td\}, \psi_5^0 = \{Ta\}$

$\psi_0^1 \cap \psi_2^0 = \{Ta\}$
$\psi_3^1 \cap \psi_2^0 = \{Ta\}$
$\psi_0^1 \cap \psi_5^0 = \{Ta\}$
$\psi_3^1 \cap \psi_5^0 = \{Ta\}$
$\psi_2^1 \cap \psi_1^0 = \{Tc\}$

Deuxième étape de l'algorithme I d'Alpan

1) $\psi_0^1 \cap \psi_3^1 = \{Ta\}$, $[\delta(S_0, Ta), \delta(S_3, Ta)] = (S_1, S_4)$, $\psi_1^1 \cap \psi_4^1 = \{Tb\}, \psi_{14} \neq \varnothing$

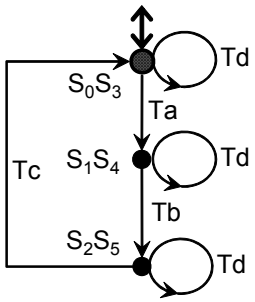
2) $\psi_1^1 \cap \psi_4^1 = \{Tb\}$, $[\delta(S_1, Tb), \delta(S_4, Tb)] = (S_2, S_5)$, $\psi_2^1 \cap \psi_5^1 = \varnothing, \psi_{25} = \varnothing$



3) $\psi_3^1 \cap \psi_4^1 = \{Td\}$, $[\delta(S_3, Td), \delta(S_4, Td)] = (S_0, S_1)$, $\psi_0^1 \cap \psi_1^1 = \varnothing, \psi_{01} = \varnothing$

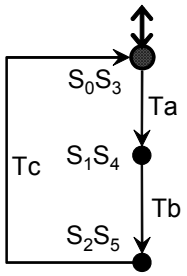
4) $\psi_4^1 \cap \psi_5^1 = \{Td\}$, $[\delta(S_4, Td), \delta(S_5, Td)] = (S_1, S_2)$

C.



Etats	Ta : γ	Tb	Tc : γ	Td
S0S3	1	-	0	1
S1S4	-	1	0	1
S2S5	0	-	1	1

D.



Etats	Ta : γ	Tb	Tc : γ
S0S3	1	-	0
S1S4	-	1	0
S2S5	0	-	1

Figure 17. Les différentes étapes de la construction d'un superviseur réduit.

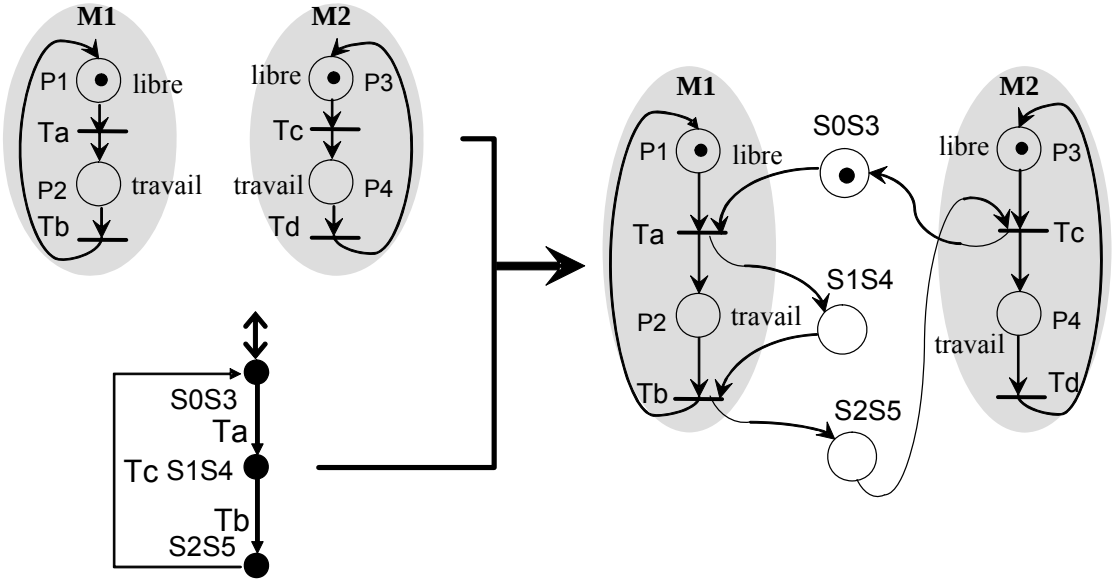


Figure 18. Projection du contrôleur réduit dans le modèle original du procédé pour obtenir le modèle RdP du comportement en boucle-fermé

L'approche d'Alpan est intéressante en raison de son caractère systématique et de sa rigueur. Nous pouvons notamment en retenir l'idée de l'intégration du superviseur dans le modèle initial sous forme de places de contrôles. Toutefois, malgré la proposition d'algorithmes de réductions, cette approche souffre dans le cas général des mêmes difficultés que l'approche de Ramadge et Wonham sur laquelle elle repose. Sa limite principale réside donc dans l'utilisation du formalisme des automates à états finis.

Dans la suite de cette analyse de l'existant, nous nous intéressons à des approches basées uniquement sur le formalisme des Réseaux de Petri.

4.2.3. Modélisation utilisant les Réseaux de Petri

D'autres approches comme celle d'A. Giua proposent la notion de Contraintes Généralisées d'Exclusion Mutuelle (ou CGEM) pour les SEDs modélisés par des réseaux de Place/Transition [GIU92] [GIU93] [QUE00] [YAM96]. Ce concept permet d'exprimer des spécifications de fonctionnement par le problème des états interdits (PEI –cf. §4.3.2). Pour des systèmes représentés en RdP, la notion de CGEM étend la notion classique de marquage interdit. Dans ce cadre, c'est une condition qui limite le nombre de jetons contenu dans un sous-ensemble de places des RdP considérés. Formellement un CGEM est défini comme suit :

Définition 11 [GIU92] : Soit $\langle N_p, M_0 \rangle$ un système de réseau avec un ensemble de places P. Un simple CGEM $(\vec{\omega}, k)$ définit un ensemble de marquages légaux

$$\mathrm{M}(\vec{\omega}, k) = \left\{ M \in IN^{|P|} \middle| (\overrightarrow{\omega})^T \cdot M \leq k \right\},$$

où $\vec{\omega} : P \to IN$ est un vecteur pondérable, et $k \in IN^+$. Le support de $\vec{\omega}$ est l'ensemble $Q_\omega = \{p \in P | \omega(p) > 0\}$.

Un ensemble de contraintes généralisées d'exclusion mutuelle $(W, \vec{k})$, avec $W = \left[\vec{w}_1 \cdots \vec{w}_m\right]$ et $\vec{k} = (k_1 \cdots k_m)^T$, définit un ensemble de marquages légaux

$$\begin{aligned} \mathrm{M}(W, \vec{k}) &= \cap_{i=1}^{m} \mathrm{M}(\vec{w}_i, k_i) \\ &= \left\{ M \in IN^{|P|} \middle| W^T \bullet M \leq \vec{k} \right\}. \end{aligned}$$

Définition 12 [GIU92] : Soit $\langle N_p, M_0 \rangle$ un système. Un CGEM $(\vec{\omega}, k)$ est réduit par rapport à l'ensemble de marquages $A \subseteq IN^{|P|}$ si $A \subseteq \mathrm{M}(\vec{w}, k)$.

Un CGEM $(\vec{\omega}, k)$ est redondant par rapport au système $\langle N_p, M_0 \rangle$ si $R(Np, M_0) \subseteq \mathrm{M}(\vec{w}, k)$.

Un ensemble de CGEM $(\vec{\omega}, k)$, où $W = \left[\vec{w}_1 \cdots \vec{w}_m\right]$ et $\vec{k} = (k_1 \cdots k_m)^T$, est redondant par rapport à $\langle N_p, M_0 \rangle$ si $(\vec{\omega}_i, k_i)$ est redondant pour tout $i = 1, \ldots, m$.

Définition 13 [GIU92] : Deux ensembles de CGEM $(W_1, \vec{k}_1)$ et $(W_2, \vec{k}_2)$ sont équivalent par rapport à $\langle N_p, M_0 \rangle$ si $R(N, M_0) \cap \mathrm{M}(W_1, \vec{k}_1) = R(N, M_0) \cap \mathrm{M}(W_2, \vec{k}_2)$.

Cependant, ces spécifications ne sont pas toujours applicables, notamment dans le cas où quelques transitions de réseaux sont incontrôlables [GIU92]. Un simple CGEM est peut-être exécuté par un « monitor », c'est-à-dire, une place dont le marquage initial représente les

unités disponibles d'une ressource et dont les transitions sortantes et entrantes représentent, respectivement, l'acquisition et la libération des unités de la ressource. La Définition 14 explique le concept de monitor.

Définition 14 [GIU92] : Soit un système $\langle N_p, M_0 \rangle$, avec $N = (P, T, \mathrm{Pr}e, Post)$, et un CGEM $(\vec{\omega}, k)$. Le contrôleur qui exécute cette contrainte est une nouvelle place *S* ajoutée à *N*. Le système résultant est dénoté $\langle N_p^S, M_0{}^S \rangle$, avec $N_p^S = (P \cup \{S\}, T, \mathrm{Pr}e^S, Post^S)$. Soit *C* la matrice d'incidence de *N*. Alors N_p^S aura pour matrice d'incidence :

$$C^S = \begin{bmatrix} C \\ -\vec{w}^T \bullet C \end{bmatrix}.$$

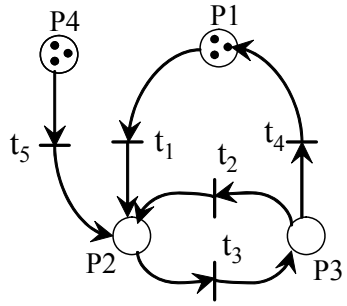
[YAM96] propose l'utilisation des invariants de places comme modèle représentatif des spécifications. Pour illustrer l'usage de la notion de CGEM, considérons l'exemple de la Figure 19.A. Nous pouvons déterminer sa matrice d'incidence '*C*' et son marquage initial $M_0 = [3 \quad 0 \quad 0 \quad 3]$ (Figure 19.B). Pour définir le comportement du système en boucle fermé, supposons donné la spécification suivante : contrôler le réseau afin que les places 'P2' et 'P3' ne contiennent jamais plus d'un jeton. En utilisant la notion de CGEM, elle est traduite par l'équation (2.1).

$$\mu_2 + \mu_3 \leq 1 \qquad (2.1)$$

L'équation (2.1) étant une inégalité elle peut être réécrite en l'équation (2.2) qui suppose l'existence d'une place de contrôle 'S'.

$$\mu_2 + \mu_3 + \mu_S = 1 \qquad (2.2)$$

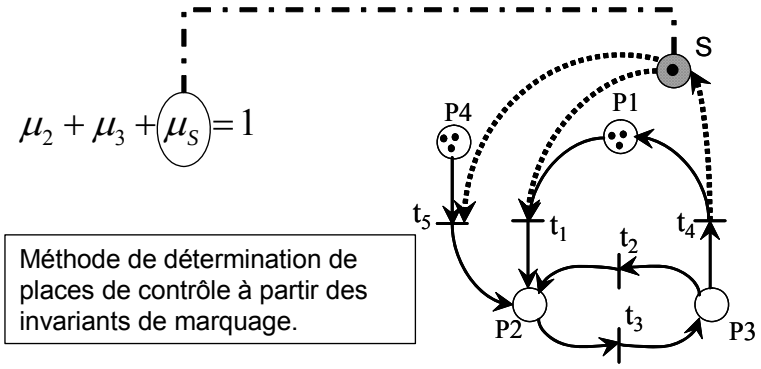
Cette réécriture permet donc de définir la place de contrôle 'S' qui est insérée dans le RdP initial (Figure 19.C). L'équation (2.2) correspond au P-semiflot $X^T = [0 \quad 1 \quad 1 \quad 0 \quad 1]$ qui permet de déterminer 'C_s' et donc les arcs reliant 'S'aux transitions du RdP initial.



A. Exemple de processus de Réseaux de Petri

$$C=\begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & 1 & -1 & 0 & 1 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad \mu_{P_0}=[\mu_1 \ \ \mu_2 \ \ \mu_3 \ \ \mu_4]^T=[3 \ \ 0 \ \ 0 \ \ 3]^T$$

B. Matrice d'incidence et marquage initial



C. Réseaux de Petri avec un contrôleur

Figure 19. Détermination de places de contrôleur basé sur la notion de CGEM des RdP initiaux

Le paragraphe suivant montre les limites de cette approche basée sur les CGEM.

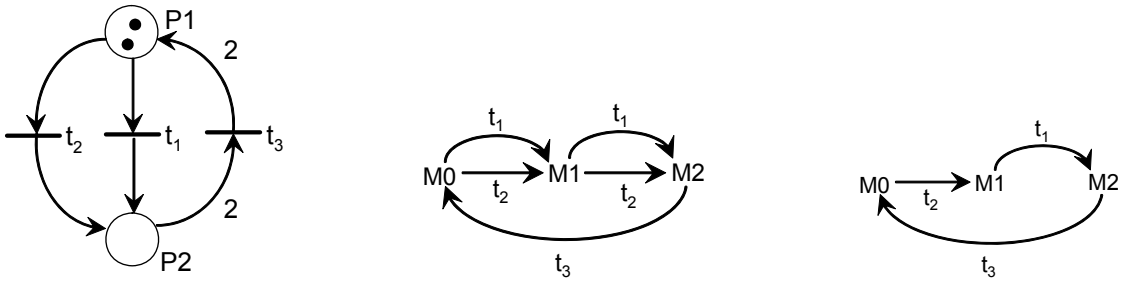
4.3 Autres types de spécifications utilisateurs

Dans cette partie, nous abordons le problème de construction formelle des spécifications utilisateurs dans le contexte d'approche de synthèse. En effet, même si la théorie du CGEM paraît être bien formalisée, le problème central reste d'assurer que le contrôleur exécute exactement la spécification des utilisateurs. Plus précisément, nous

voulons discuter des équivalences et des différences entre les spécifications fondées sur le comportement souhaité et les spécifications basées sur le comportement interdit. Dans la littérature, plusieurs travaux ([HOL90], [ACH04], [GHA02a] et [RAM87b]), montrent que ces deux types d'approches sont équivalentes. Selon nous, il est aussi nécessaire de considérer un comportement toléré. Quand on spécifie un comportement basé sur l'approche interdite, on n'obtient pas exactement le comportement souhaité mais le comportement toléré. Ce comportement peut être assimilé à un espace d'état plus grand que l'espace d'état correspondant exactement au comportement souhaité. Alors l'usage de concept de comportement interdit est plus permissif que celui du comportement souhaité. Mais notre objectif ici est de montrer que toutes les spécifications ne se ramènent pas de manière simple à des problèmes d'états interdits. On ne peut donc pas systématiquement appliquer la technique du CGEM.

4.3.1. Spécifications de type transitions d'état interdites [GHA02]

Pour illustrer le problème de transitions d'état interdites (ou PTEI) considérons l'exemple de la Figure 20.



A. Modèle du procédé B. Graphe d'accessibilité C. Comportement souhaité

Figure 20. Problème de transitions d'état interdites [GHA02a]

Dans la Figure 20.A, les transitions 't_1' et 't_2' sont toutes deux franchissables à partir de le même place 'P1' et mènent à la place 'P2'. Le comportement en boucle ouverte de ce système est donné par le graphe d'accessibilité de la Figure 20.B. Supposons que le comportement souhaité par l'utilisateur consiste à effectuer les séquences de franchissement (t2.t1.t3)* tel qu'illustré par la Figure 20.C. Ce problème et le PEI sont différents. On peut

voir dans cet exemple que l'ensemble du comportement interdit est $M_0 \xrightarrow{t1} M_1$ et $M_1 \xrightarrow{t2} M_2$. ' M_1 ' et ' M_2 ' restent des marquages admissibles et les transitions ' t_1 ' et ' t_2 ' peuvent toujours être franchies dans le comportement souhaité mais seulement dans des conditions particulières. De manière formelle un PTEI est défini comme suit :

Définition 15 [GHA02a] : Un problème de transitions d'état interdits (ou PTEI) est un problème de supervisory control basée sur les RdP. Le comportement souhaité de modèle du procédé (N_p, M_0) doit satisfaire les conditions suivantes :

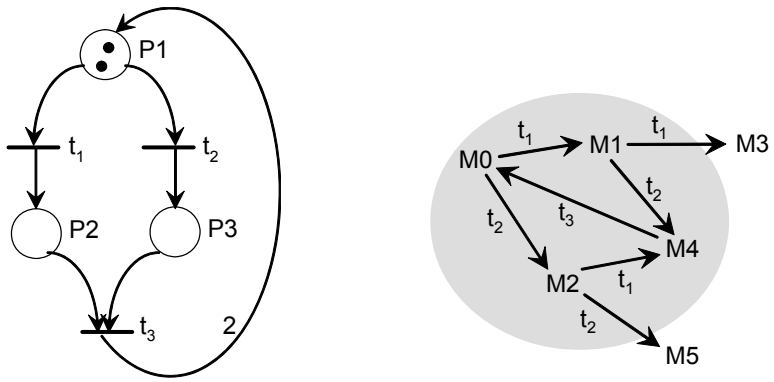
(a) $M_0 \in CS$ (CS: Comportement Souhaité)

(b) $\forall M \in CS$, $(M_0 \xrightarrow{\sigma} M) \in CS$

(σ : Séquence de transitions franchies)

4.3.2. Le problème d'état interdit

Comme nous l'avons déjà souligné, cette notion est la plus couramment utilisée dans les méthodes de synthèse basées sur le formalisme des RdP. Elle peut être expliquée par le fait que quand le comportement d'un SED est modélisé sous forme de RdP, son espace d'états est naturellement représenté par un graphe d'accessibilité. Dans ce contexte, l'exécution du concept d'états interdits consiste à supprimer les états correspondants de l'espace interdit pour obtenir le graphe d'accessibilité du comportement toléré. Présentons l'exemple de la Figure 21 pour expliquer le concept du problème d'état interdit (ou PEI).



A. Modèle du procédé B. Graphe d'accessibilité

Figure 21. Problème d'état interdit

La Figure 21.A représente le modèle RdP du comportement d'un procédé en boucle ouverte. Le graphe d'accessibilité correspondant à ce modèle (Figure 10 B) montre bien que ce système peut atteindre deux états de blocage. L'objectif du contrôle serait d'éviter ces blocages en interdisant l'accès à ces états.

Définition 16 [GHA02a] : Un problème d'état interdit (ou PEI) est un problème de « supervisory control » basé sur les RdP. Le comportement souhaité de modèle du procédé (N_p, M_0) doit satisfaire les conditions suivantes :

(a) $M_0 \in CS$ (CS: Comportement Souhaité)

(b) $\forall M \in CS$, $(M_0 \xrightarrow{\sigma} M) \in CS$

(σ : Séquence de transitions franchies)

(c) $\forall (M \xrightarrow{t} M') \in R(N_p, M_0),\ M, M' \in CS \Rightarrow (M_0 \xrightarrow{t} M') \in CS$

($R(N_p, M_0)$: Graphe d'accessibilité)

De nombreux travaux proposent une méthode de résolution de PEI concernent les RdP ([HOL90], [GIU92], [ACH04] et [GHA02a]). [HOL90] et une méthode de conception de contrôleurs basés sur les RdP pour répondre au PEI. Cette méthode est basée sur un type de RdP particulier présentant des propriétés structurelles utiles pour le problème de synthèse de contrôle. Il est basé sur les prédicats linéaires définis sur les modèles de type graphe d'événement sauf et vivant, les graphes d'événements cycliques (GEC).

Giua s'intéresse principalement aux spécifications données par des inégalités linéaires sur les marquages, CGEM ([GIU92]). Malheureusement, toute spécification d'états interdits ne peut pas se ramener à un ensemble de CGEM. En effet, l'ensemble d'états admissibles correspondant à une contrainte de type CGEM est forcément convexe. Ce qui n'est pas nécessairement le cas pour un problème où les états interdits sont définies par une liste quelconque. Cependant, pour la classe des réseaux saufs et conservatifs, il existe toujours un ensemble de CGEM équivalent à une liste quelconque d'états interdits [GHA02b].

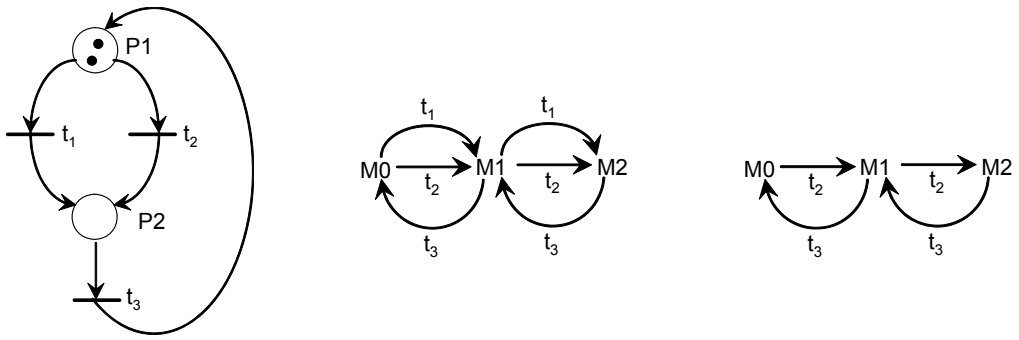
D'autres auteurs comme [ACH04] permettent de résoudre le PEI caractérisé par un ensemble de CGEM en exploitant les propriétés structurelles des graphes d'événements. Cela

a permis de synthétiser une loi de commande pour le PEI avec la contrainte d'observabilité des événements.

Ghaffari propose une approche de synthèse de contrôleurs RdP en utilisant la théorie des régions ([GHA02a], [GHA02b] et [GHA02c]). La théorie des régions a initialement été dédiée à la synthèse de RdP à partir d'automates. Elle a dû être adaptée au problème de synthèse de contrôle RdP. Dans le chapitre III, nous adopterons cette approche pour résoudre le problème de séquence interdite de transitions d'états.

4.3.3. Le problème de transitions interdites

Nous présentons à présent le problème de transitions interdites (ou PTI). C'est un problème défini dans le cadre du contrôle de SEDs basés sur le formalisme des RdPs. Quand une transition est concernée par le problème de PTI, cela signifie que toutes ses occurrences doivent être supprimées du graphe d'accessibilité original du modèle du procédé.



A. Modèle du procédé B. Graphe d'accessibilité C. Comportement souhaité

Figure 22. Problème de transitions interdites

Pour illustrer cette problématique, considérons l'exemple de la Figure 22. La Figure 22.A représente le modèle RdP du procédé en boucle ouverte et la Figure 22.B son graphe d'accessibilité. Nous supposons que l'utilisateur désire que le système exécute à chaque cycle la tâche correspondant au franchissement de la transition 't_2'. Pour obtenir le comportement souhaité, il suffit alors de supprimer la transition 't_1' dans le graphe d'accessibilité de départ, ce qui donne finalement le comportement souhaité, illustré par la Figure 22.C. Nous

remarquons sur cet exemple, que ce type de spécification ne nécessite pas forcément la suppression d'un marquage quelconque. Nous définissons le PTI comme suit.

Définition 17 : Un problème de transitions d'état interdits (ou PTEI) est un problème de « supervisory control » basée sur les RdP. Le comportement souhaité du modèle du procédé (N_p, M_0) doit satisfaire les conditions suivantes :

(a) $M_0 \in CS$ (CS: Comportement Souhaité)

(b) $\forall M \in CS$, $(M_0 \xrightarrow{\sigma} M) \in CS$

(σ : Séquence de transitions franchies)

(c) $\forall (M \xrightarrow{t} M') \in R(N_p, M_0),\ t \in T_F \Rightarrow (M \xrightarrow{t} M') \notin CS$

avec T_F l'ensemble des transitions non franchissables, et $R(N_p, M_0)$ le Graphe d'accessibilité.

5. Conclusion

Dans ce chapitre, nous avons introduit les concepts de système reconfigurable et de processus de reconfiguration. Nous avons cherché à montrer que la capacité à être reconfigurable exige une architecture de contrôle/commande intégrant de nombreuses fonctions afin de réagir à différents types d'événements pouvant engendrer une reconfiguration. Pour être reconfigurable, une commande doit être modulaire est paramétrable. Pour atteindre cet objectif, il existe deux grandes familles de méthodes : les méthodes basées sur l'ingénierie de la commande et les méthodes de synthèse de commande. Nous avons dégagé les avantages de chacune de ces méthodes. Originellement notre approche était exclusivement basée sur l'ingénierie de la commande. Mais nous pensons aujourd'hui qu'il faut développer des méthodes mixtes tirant parti des avantages des deux familles. Aussi dans la suite de cette étude, nous allons chercher à proposer une approche de synthèse de la commande basée sur le formalisme RdP par homogénéité avec nos approches antérieures basées sur ce formalisme.

Chapitre II: Approche Réseaux de Petri pour la synthèse de contrôleurs

1. Introduction

Dans le chapitre précédent, nous avons abordé le problème du codage formel des spécifications utilisateurs dans le contexte d'approches de synthèse. Dans ce cadre, nous avons discuté des équivalences et des différences entre les spécifications fondées sur le comportement souhaité et les spécifications basées sur le comportement interdit. Dans ce chapitre, nous allons d'abord proposer un nouveau type de spécifications utilisateurs modélisables à l'aide des Réseaux de Petri (RdP). Nous dénommons le problème associé : Problème de Séquences Interdites de Transitions d'Etat (ou PSITE).

Dans un second temps, nous allons proposer une méthode pour réaliser la synthèse de contrôleurs uniquement à partir de modèles RdP. Notre objectif est double : d'une part traiter les PSITE et d'autre part proposer une approche moins combinatoire que les approches basées sur les automates à états finis ou la construction du graphe d'accessibilité. La résolution proposée s'appuiera sur la théorie des régions revisitées par A. Ghaffari [GHA02b].

2. Problème des séquences interdites de transitions d'état

Dans cette section, nous présentons le problème des séquences interdites de transitions d'état qui est un problème de contrôle respectant des spécifications de type supervision et qui est basé sur le formalisme des Réseaux de Petri.

Définition 18 Un problème de séquences interdites de transitions d'état (ou PSITE) est un problème de supervision basé sur les RdP. Le comportement souhaité du modèle du procédé (N_p, M_0) doit satisfaire les conditions suivantes :

(a) $M_0 \in CS$ (CS: Comportement Souhaité)

(b) $\forall M \in CS$, $\exists \sigma \mid (M_0 \xrightarrow{\sigma} M) \in CS$

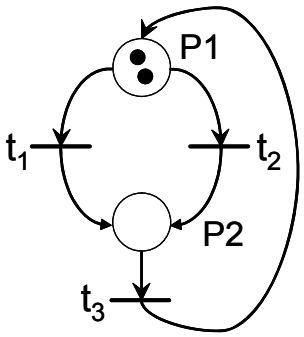
(σ : Séquence de transitions franchies)

(c) $\exists (M_0 \xrightarrow{\gamma} M') \in R(N_p, M_0)$ et $(M_0 \xrightarrow{\gamma} M') \notin CS$

(γ : Séquence de transitions qui est interdite et

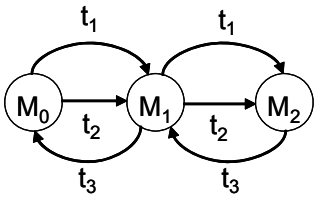
$R(N_p, M_0)$: Graphe d'accessibilité du modèle RdP considéré)

Pour comprendre cette notion, nous allons l'expliquer à l'aide de l'exemple de la Figure 23. Le comportement du procédé peut être formulé par un ensemble de séquences de transitions légales. La Figure 23.A illustre le modèle RdP du procédé en boucle ouverte et la Figure 23.B donne l'ensemble des séquences et le graphe d'accessibilité en boucle ouverte. Fondé sur la théorie du langage, cet ensemble peut être résumé par une expression régulière.



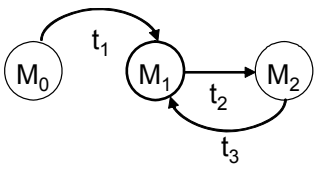
A. Modèle du procédé

$$\left((t_1 + t_2)\left(t_3 + \left((t_1 + t_2)t_3\right)^* t_3\right)\right)^*$$



B. Séquences et graphe d'accessibilité en boucle ouverte

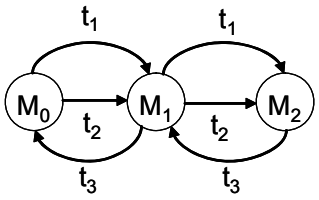
$$t_1 t_2 t_3$$



C. Séquence interdite et graphe d'accessibilité interdit

$$\left((t_1 + t_2)\left(t_3 + \left((t_1 + t_2)t_3\right)^* t_3\right)\right)^*$$

$$- \; t_1 t_2 t_3$$



D. Séquences souhaitées et le graphe d'accessibilité associé

Figure 23. Le problème des séquences interdites de transitions d'état

La Figure 23.B montre que plusieurs séquences permettent de conduire le système du marquage initial 'M_0' au marquage 'M_1'. Dans ce cas, supposons que l'utilisateur désire interdire l'ensemble des séquences préfixées par 't1*t2*t3' (Figure 23.C). Ce problème est fondamentalement différent du problème d'état interdit car pour réaliser une séquence comme 't1*t1*t3*t2', il est nécessaire de conserver tous les états du graphe d'accessibilité du comportement en boucle ouverte. Ce problème ne peut pas non plus être assimilé au problème de transitions d'état interdites (ou PTEI) qui est un processus inhibant certaines transitions. Par exemple, si l'on essaie d'inhiber $M_0 \xrightarrow{t_1} M_1$ pour éviter la séquence 't1*t2', il serait impossible de réaliser les séquences de type (t1.t3)* qui ne sont pas interdites. En fait ce qui est important dans ce problème, ce n'est pas l'existence ou pas d'un état, ou le fait qu'une transition d'état soit franchissable ou pas. Ce qui est important, c'est le comportement qui permet d'arriver dans un état. Si on considère la séquence interdite 't1*t2*t3', on peut donc réaliser 't1*t2' ou 't1*t1*t3'. Ce qu'on interdit, c'est si on est arrivé dans l'état M2 par la séquence 't1*t2', on ne veut surtout pas autoriser une évolution de M2 vers M3 par le franchissement de 't3'.

D'un point de vue pratique, ce problème peut être lié au contrôle d'évolutions dangereuses d'un système. L'arrivée dans M2 après l'évolution 't1*t2' peut traduire un comportement potentiellement dangereux par exemple en raison d'une défaillance, ce qui exigerait d'interdire les évolutions qui entraîneraient à utiliser les parties du système défaillantes.

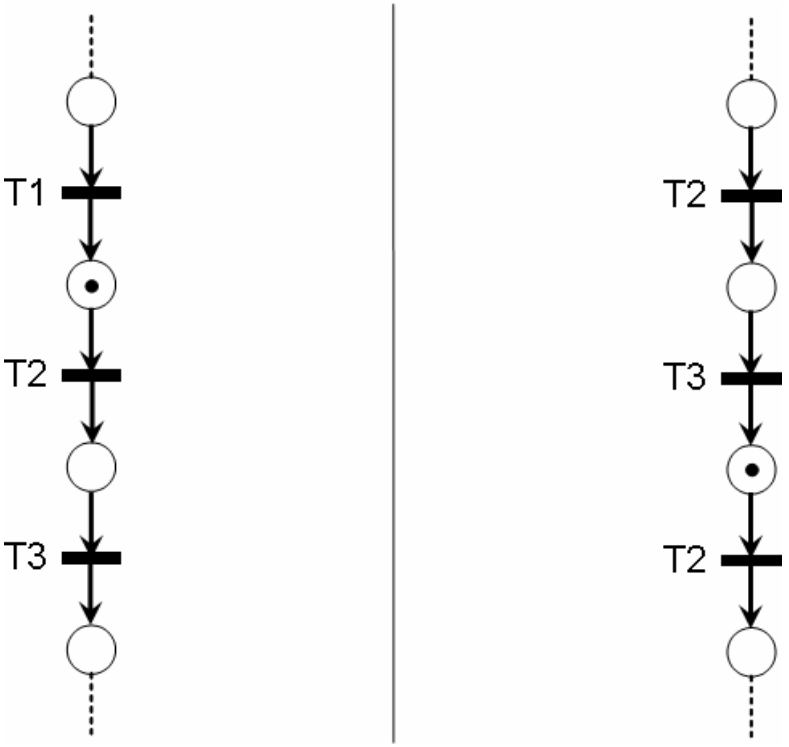
3. Principe de la synthèse de commande basée sur les RdP

L'approche que nous proposons est une approche inspirée par la théorie du « Supervisory Control » de Ramadge et Wonham. Elle consiste à distinguer le modèle RdP du procédé du procédé en boucle ouverte et le modèle RdP des spécifications. Le modèle RdP de spécification traduit des spécifications positives. C'est-à-dire qu'il exprime un comportement souhaité et non pas un comportement interdit. Il est construit à l'aide de transitions du modèle RdP du procédé. Les transitions communes dans les deux modèles sont supposées être synchronisées. Une transition du modèle du procédé peut apparaître plusieurs fois dans le modèle de spécification.

Afin de préciser le comportement de l'ensemble des deux modèles, nous proposons les deux règles suivantes :

Règle 2 : Une transition du modèle de spécification peut être franchie seulement quand elle est franchissable dans les deux modèles.

La Règle 2 implique que si l'occurrence d'une transition dans le modèle de spécification est franchissable et que la même transition est franchissable dans le modèle du procédé, alors la transition en question est franchissable. Donc, nous n'avons pas besoin que toutes les occurrences dans le modèle de spécification d'une transition du modèle du procédé soient franchissables pour que cette transition soit effectivement franchissable. Ce cas est illustré par la transition 'T2' dans la Figure 24. Cette transition étant validée dans les deux modèles, elle est donc franchissable.

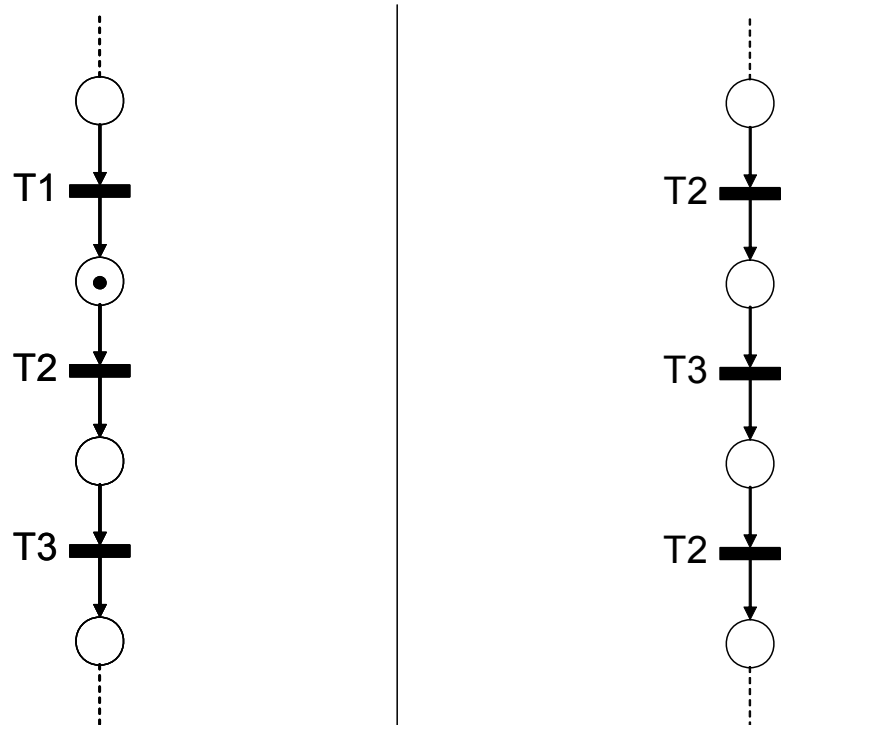


A. Modèle du procédé B. Modèle de la spécification

Figure 24. Illustration de la condition d'application de la Règle 2

Règle 3 : Une transition du modèle du procédé est inhibée si cette transition existe dans les deux modèles mais qu'elle n'est pas franchissable dans le modèle de spécification.

Cette deuxième règle traduit plusieurs comportements. D'abord si une transition du modèle du procédé n'est pas utilisée dans la spécification, son franchissement n'est pas contraint par la spécification. D'autre part, lorsqu'une transition du modèle du procédé n'a aucune occurrence franchissable dans le modèle de spécification, alors cette transition doit être inhibée si elle est franchissable dans le modèle du procédé (Figure 25).



A. Modèle du procédé B. Modèle de la spécification

Figure 25. Illustration de la condition d'application de la Règle 3

Afin d'illustrer comment construire un modèle de spécification connaissant le modèle du procédé, nous allons nous intéresser à la synthèse du modèle de spécification d'un stock de capacité fini qui contraint le fonctionnement de machines.

4. Spécification des contraintes de comportement induites par un stock

Nous utilisons cet exemple illustratif, car il sera réutilisé au chapitre III, pour présenter la manière de réaliser la commande du système de transport d'un système d'un système manufacturier. Nous nous intéressons aux systèmes de transport de type convoyeur à bande qui est caractérisé par la présence de stocks intermédiaires entre les machines ou au cas de robots stockant ou retirant des produits d'un stock pour charger ou décharger des machines.

Dans cet exemple, nous supposons que nous généralisons le modèle du producteur-consommateur (Figure 26). Nous supposons que nous avons n machines productrices notées $M_\mu{}^i$ ($i = 1,\ldots n$) et m machines consommatrices notées $M_D{}^j$ ($j = 1,\ldots m$). Les machines

productrices déposent les pièces produites dans un stock de capacité K . Les machines consommatrices retirent une pièce du stock pour réaliser leur production (Figure 26). Les machines sont supposées être de capacité unitaire.

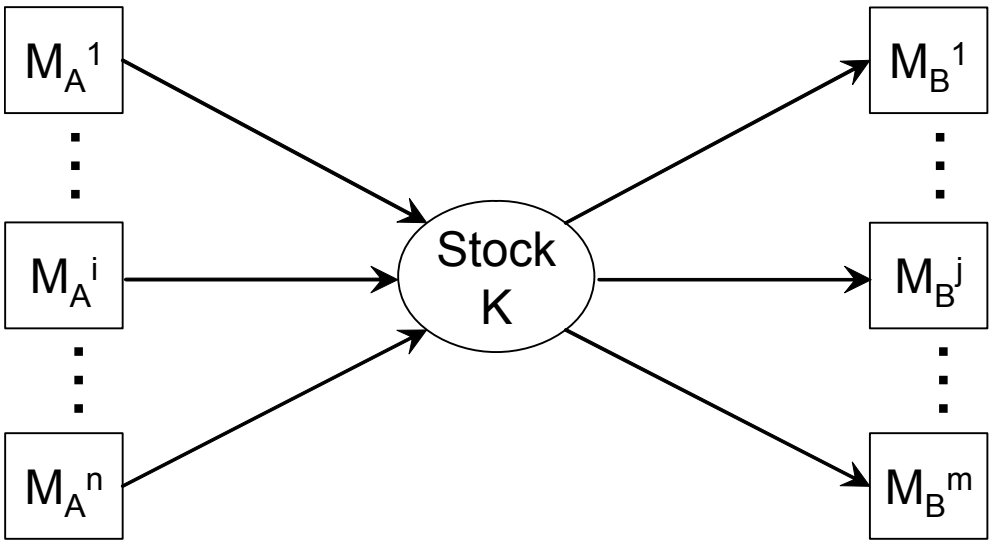
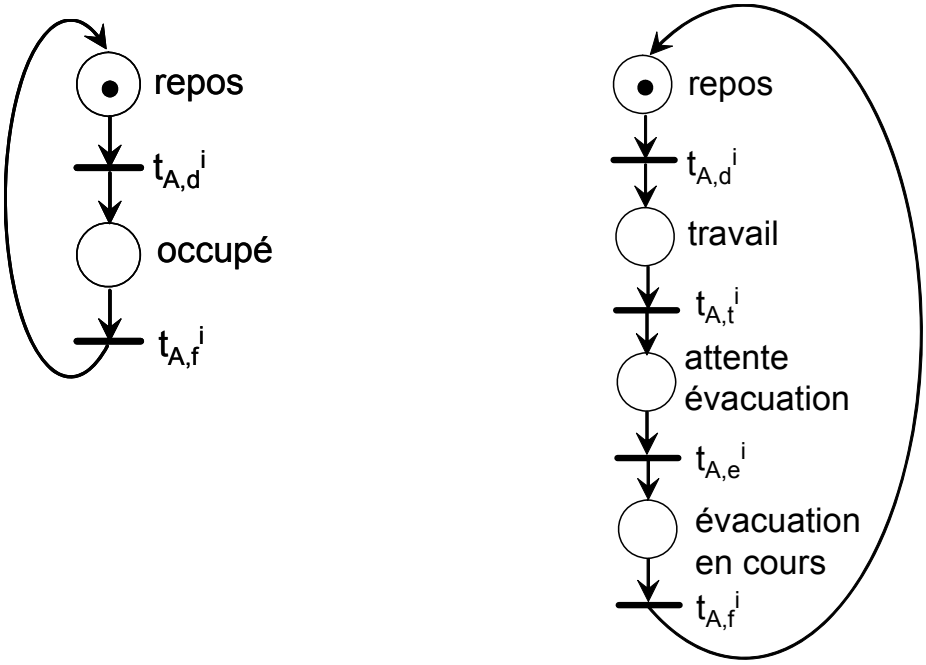


Figure 26. Système à n producteurs et m consommateurs

Dans un premier temps nous commençons par modéliser le comportement des machines fonctionnant de manière indépendante. Selon, le cahier des charges, on peut caractériser ce fonctionnement par un nombre plus ou moins important d'états de la machine. Par exemple, si on cherche à caractérisé uniquement l'état libre (état de repos) ou occupé de la machine, on peut modéliser son comportement par une RdP ordinaire à deux places et deux transitions (Figure 27.A). Si nous considérons la machine M_A^i , notons $t_{A,d}^i$ et $t_{A,f}^i$ les transitions qui caractérisent le début de l'occupation de la machine et la fin de son occupation. Si l'état occupé correspond à un état de travail à ce moment la transition $t_{A,d}^i$ modélise le début de l'opération de transformation et $t_{A,f}^i$ la fin de cette transformation. Mais l'état machine occupé est un état qui peut être raffiné. En effet, le cahier de charges peut nous amener à distinguer des états ou après transformation d'une pièce, la machine peut être en attente d'évacuation de la pièce. Ensuite même lorsque l'évacuation a commencé, la machine n'est pas disponible avant la fin de l'évacuation. Cet affinement possible est illustré par la Figure 27.B dans lequel on conservé les transitions qui caractérisent le début et la fin de l'occupation de la machine.



A. Modèle élémentaire d'une machine

B. Exemple de raffinement du modèle d'une machine

Figure 27. Exemple de modèles du comportement non contraint d'une machine

A présent, intéressons nous aux contraintes de fonctionnement imposées aux machines par la présence d'un stock de capacité *K*. La capacité du stock correspond au nombre total d'emplacements dont il dispose pour accueillir des produits. Les contraintes imposées par le stock se traduisent par une spécification suivante qui dépend du modèle de comportement des machines. Par exemple, si nous considérons le modèle élémentaire d'une machine (Figure 27.A), et que la place occupée correspond à un état de travail de la machine, alors la spécification due au stock serait la suivante :

Spécification 1 : Une machine productrice ne doit pas produire si le stock est plein. Une machine consommatrice ne doit pas produire si le stock est vide.

Si nous considérons comme de modèle du comportement de la machine, le modèle raffiné donné par la Figure 27.B, on aurait alors la spécification suivante :

Spécification 2 : Une machine productrice ne peut pas être évacuée si le stock est plein. Une machine consommatrice ne peut pas prélever de pièce si le stock est vide.

Notons que la spécification 2 permet à chaque machine d'avoir une pièce produite temporairement stockée sur la machine.

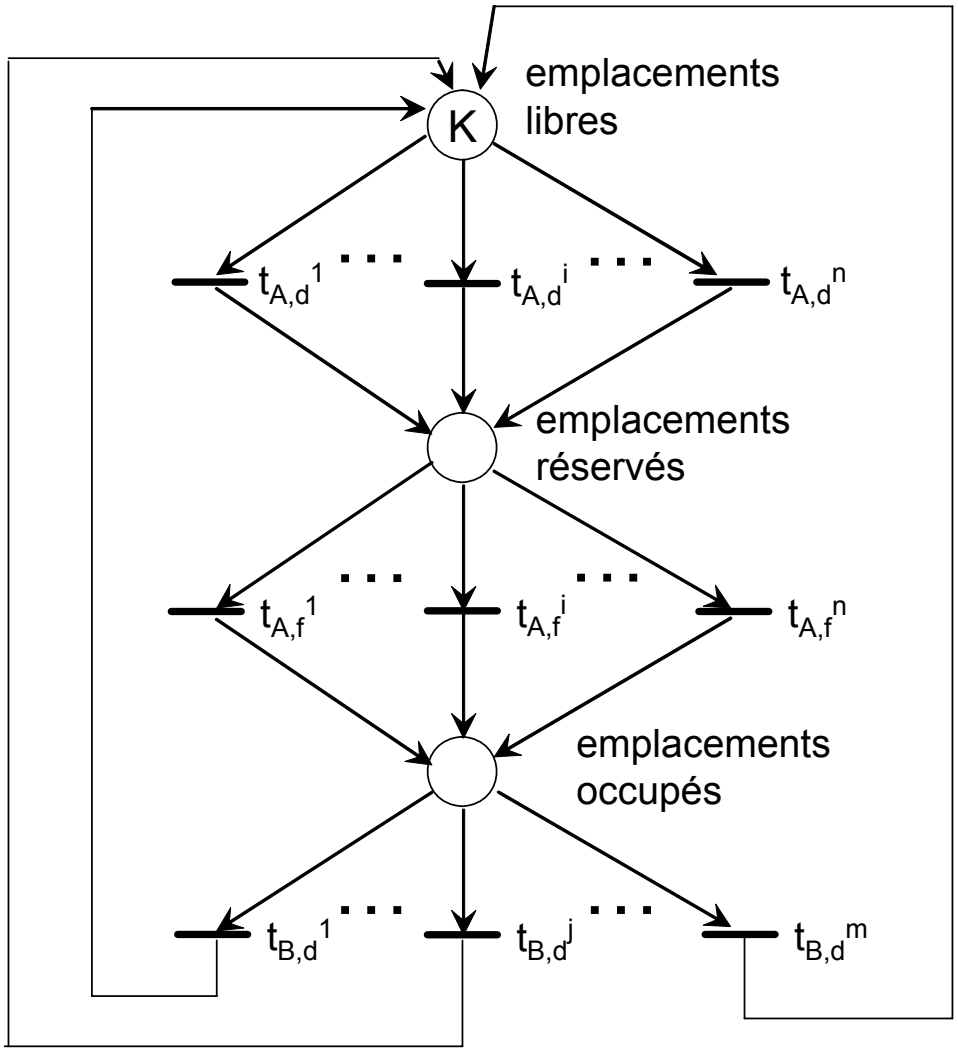


Figure 28. Modèle RdP générique de spécification dans le cas de modèles de machines correspondant à la Figure 27.A

Du point de vue de la machine, ce qui est important dans la spécification, c'est de déterminer s'il reste des emplacements libres ou pas dans le stock. Si on considère la spécification 1, on se rend compte que si le nombre de machines productrices *n* est supérieur à *K*, si on laisse les machines commencé à travailler avant de vérifier qu'il y a de la place dans le stock, on peut se retrouver une fois le stock plein avec une machine obligé de poursuivre l'opération de transformation quand bien même elle serait normalement terminé. Pour éviter cela, il faut que chaque machine, lorsqu'elle est mise en marche, elle réserve un emplacement dans le stock. De ce fait, on va caractériser le stock par trois états : emplacements libres du stock, emplacements réservés et emplacements occupés. La Figure 28 représente un stock considéré comme initialement vide. Nous avons donc K jetons dans la place 'emplacements libres'. Dans cet état, bien que les transitions $t^{j}_{B,d}$ des machines productrices soient validées elles doivent être inhibées car, ces transitions apparaissent dans la spécification sans être validées. Quand une machine commence sa production, une transition $t^{i}_{A,d}$ est franchie, ajoutant un jeton dans la place 'emplacements réservés'. A la fin de la production, le franchissement de la transition $t^{i}_{A,f}$ ajoute un jeton dans la place 'emplacements occupés'. Dès qu'il y a un jeton dans cette place, il est possible de démarrer une machine productrice. Si tous les jetons se retrouvent dans la place emplacements occupés, les transitions $t^{i}_{A,d}$ de démarrage des machines productrices ne sont plus validées dans le modèle de spécification, donc elles doivent toutes être inhibées dans les modèles des machines.

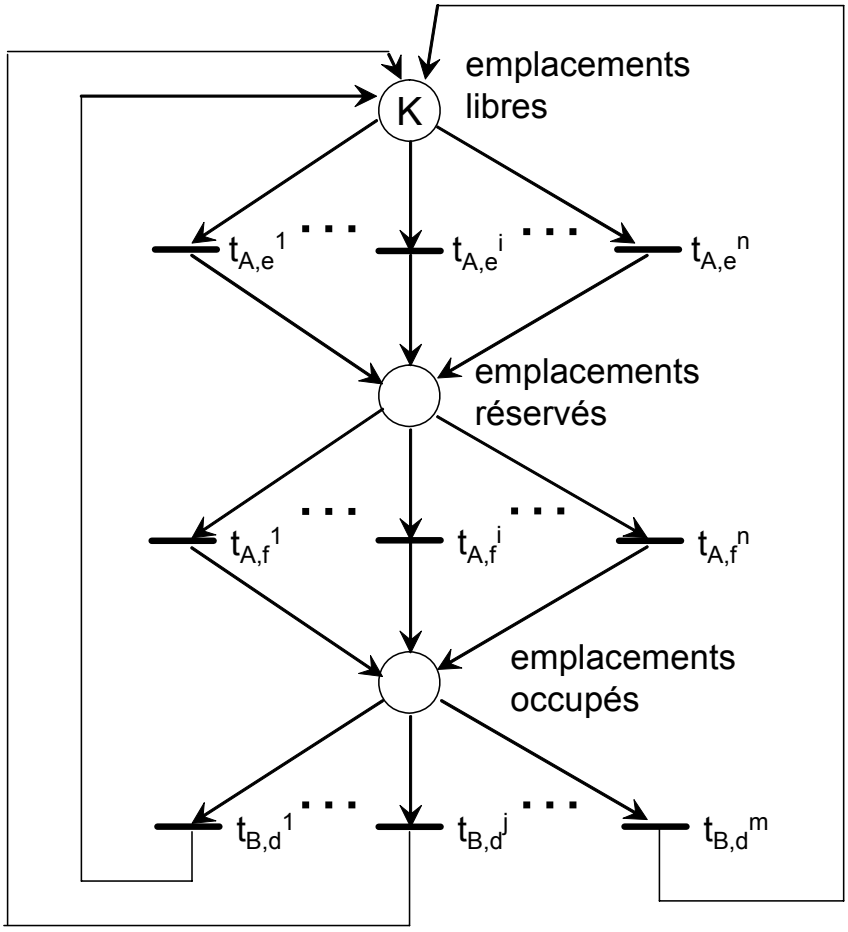


Figure 29. Modèle de spécification correspondant à la spécification 2 et aux modèles des machines donnés par la Figure 27.B.

Si nous considérons la spécification 2 associée au modèle de machines de la Figure 27B, alors nous obtenons par contre le modèle de spécification de la Figure 29. On peut remarquer que la seule différence entre les deux modèles et le choix des transitions qui permettent de réserver des emplacements dans le stock. En effet, dans ce modèle, la transition de début de transformation $t_{A,d}^{i}$ est remplacée par celle de début d'évacuation $t_{A,e}^{i}$. Cette spécification permet de lancer la transformation d'une pièce sur une machine même lorsque le stock est plein.

5. Synthèse de RdP par Graphe d'Accessibilité Synchrone Contraint

L'intérêt de la synthèse basée sur le formalisme des RdP peut être expliqué par le fait que, lorsque le comportement du SED est modélisé en utilisant les RdP, son espace d'état est naturellement représenté par son graphe d'accessibilité. Néanmoins, ce modèle peut devenir compliqué, notamment quand le modèle du système est représenté par un RdP non borné. Nous admettrons que notre méthode est développée pour des modèles RdP bornés. Pour réduire l'explosion combinatoire du graphe d'accessibilité, nous proposons une approche de synthèse basée sur le concept de Graphe d'Accessibilité Synchrone (GAS). Précisons tout de suite que notre intention n'est pas la construction de ce graphe, mais son exploitation implicite pour déterminer les équations nécessaires à l'application de la théorie des régions (cf. § 6.1). Dans un premier temps nous allons présenter le problème de l'explosion combinatoire résultant de la construction du graphe d'accessibilité synthétisant le comportement de plusieurs RdP. Nous montrerons ensuite comment construire un Graphe d'Accessibilité Synchrone contraint permettant de limiter cette combinatoire dans le cadre de la synthèse de contrôleurs respectant des spécifications utilisateurs.

5.1 Graphe d'Accessibilité Synchrone

La construction d'un graphe d'accessibilité synchrone n'a de sens que si des RdP à connexité indépendante partagent des transitions qui doivent être franchies de manière synchrone. Dans ce cas, le graphe d'accessibilité résultant est réduit par rapport au graphe d'accessibilité standard obtenu en effectuant le produit cartésien des graphes correspondant à chacun des RdP initiaux. La réduction consiste en la fusion de certaines transitions entre marquages du graphe d'accessibilité standard. Cela entraîne la suppression de certains marquages du graphe d'accessibilité standard.

Définition 19 : $\forall\left(M_k^1 \xrightarrow{t_i^1} M_j^1\right) \in R\left(N_p^1, M_0^1\right)$ et $\forall\left(M_q^2 \xrightarrow{t_p^2} M_i^2\right) \in R\left(N_p^2, M_0^2\right)$ alors si $\left(\begin{bmatrix} M_k^1 \\ M_q^2 \end{bmatrix} \xrightarrow{t_i^1} \begin{bmatrix} M_j^1 \\ M_i^2 \end{bmatrix}\right) \in R\left(\begin{bmatrix} N_p^1 \\ N_p^2 \end{bmatrix}, \begin{bmatrix} M_0^1 \\ M_0^2 \end{bmatrix}\right)$ et $\left(\begin{bmatrix} M_j^1 \\ M_q^2 \end{bmatrix} \xrightarrow{t_p^2} \begin{bmatrix} M_j^1 \\ M_i^2 \end{bmatrix}\right) \in R\left(\begin{bmatrix} N_p^1 \\ N_p^2 \end{bmatrix}, \begin{bmatrix} M_0^1 \\ M_0^2 \end{bmatrix}\right)$ et si t_i^1 et

t_p^2 sont deux transitions synchrones telles que $t_i^1 = t_p^2$ alors $\left(\begin{bmatrix} M_k^1 \\ M_q^2 \end{bmatrix} \xrightarrow{t_i^1} \begin{bmatrix} M_j^1 \\ M_i^2 \end{bmatrix}\right) \in RS\left(\begin{bmatrix} N_p^1 \\ N_p^2 \end{bmatrix}, \begin{bmatrix} M_0^1 \\ M_0^2 \end{bmatrix}\right)$ avec RS le graphe d'accessibilité synchrone.

5.2 Algorithme de construction du graphe d'accessibilité synchrone contraint [LEE06a]

Nous utilisons le graphe d'accessibilité synchrone (GAS) pour obtenir la synthèse optimale du modèle du procédé et du modèle de spécification. Nous supposons que nos modèles initiaux sont des RdP possédant des transitions synchrones. Ainsi, en respectant les règles de validation donnée au paragraphe 4 (notion de contraintes), ces modèles peuvent être interprétés pour donner le GAS contraint (GASC) par fusion synchrone de leur graphe d'accessibilité.

Afin de définir formellement la construction d'extraction des règles de la théorie des régions, nous proposons les deux théorèmes suivants.

Sachant que le marquage M_i est composé des places des différents modèles RdP nous adoptons les notations suivantes :

M_0 est le marquage initial ;

$M_0[\sigma > M_n$ est un marquage accessible ;

σ est une séquence de transitions franchies ($\sigma = t_1 t_2 \cdots t_n$) ;

MA est l'ensemble des marquages autorisés ;

MT est l'ensemble des marquages traités ;

T_F est l'ensemble des marquages interdites ;

SA est l'ensemble des séquences autorisés dans le graphe d'accessibilité qui est global ;

SI est l'ensemble des séquences interdites de transitions d'état dans le graphe d'accessibilité qui est global;

Théorème 1 : S'il existe une transition t_{ik} / $M_i[t_{ik}>M_k$ et que t_{ik} vérifie la **Règle 2**, alors $\forall \sigma \in AS$ / $M_0[\sigma > M_i$, $\sigma * t_{ik} \in AS$.

Preuve :

Si t_{ik} vérifie la Règle 2, cela signifie que cette transition est franchissable à la fois dans le modèle du procédé et dans le modèle de la spécification et donc il existe M_k / $M_i[t_{ik}>M_k$. Etant donné une séquence $\sigma \in SA$ / $M_0[\sigma > M_i$, cela signifie que M_i appartient à l'ensemble des marquages autorisés MA. Par conséquences comme est $M_i[t_{ik}>M_k$ autorisée, cela implique que M_k appartient à MA et donc que $\forall \sigma \in SA, \sigma * t_{ik} \in AS$

Théorème 2 : S'il existe une transition t_{ik} / $M_i[t_{ik}>M_k$ and t_{ik} vérifie la Règle 3, alors $\forall \sigma \in AS$ / $M_0[\sigma > M_i$, $\sigma * t_{ik} \in SI$.

Preuve :

Si t_{ik} vérifie la Règle 3, cela signifie que cette transition n'est pas validée dans le RdP du modèle de spécification. Par conséquence, la transition de M_i à M_k est interdite et donc M_k appartient à MI. S'il existe une séquence $\sigma \in AS$ / $M_0[\sigma > M_i$ comme $M_k \in MI$, alors $\sigma * t_{ik}$ est nécessairement une séquence illégale et appartient donc à SI.

En fait, nous pouvons constater au travers de ces deux théorèmes que la fusion du modèle du procédé et la fusion du modèle de spécification aboutit à translater le problème des séquences interdîtes de transitions d'état défini dans le modèle du procédé en boucle ouverte en un problème de transition interdite dans l'espace d'états du modèle fusionné. Nous illustrerons par la suite cette transformation au travers d'un exemple didactique. Cette transformation nous permet donc d'envisager l'utilisation de la théorie des régions de A. Ghaffari [GHA02b] pour trouver le contrôleur du système en boucle fermé. Aussi sur la base des théorèmes précédents, nous proposons l'algorithme suivant permettant la construction du GASC de RdPs synchronisés par des transitions communes.

Algorithme 1 : Algorithme de construction du Graphe d'Accessibilité Synchrone Contraint des modèles du procédé en boucle ouverte et de spécification.

Etape 1. *Etant donné le modèle du procédé et un modèle de spécification caractérisés par deux RdP, définir* M_0 *le marquage initial,*

Etape 1.1. *Ajouter* M_0 *à MA ;*

Etape 2. *Tant que MA n'est pas vide faire.*

Etape 2.1. *Retirer le marquage M_i de MA ;*

Etape 2.2. *Si M_i n'existe pas dans MT alors*

Soit T_{Mi} , l'ensemble des transitions franchissables à partir de M_i .

Etape 2.2.a *Tant que T_{Mi} n'est pas vide, retirer une transition t de $T_{Mi} / M_i [t > M_n$;*

Etape 2.2.1. *Si t est seulement une transition de modèle du procédé et t peut être franchie alors M_n est ajouté à l'ensemble de marquages autorisés MA et σ de $M_0 [\sigma > M_n$ est ajouté à l'ensemble de séquences autorisés SA ; fin_si;*

Etape 2.2.2. *Si t est une transition à la fois du modèle du procédé et du modèle de spécification et si t peut être franchie dans les deux modèles RdP alors M_n est ajouté à l'ensemble de marquage autorisé MA et σ de $M_0 [\sigma > M_n$ est ajouté à l'ensemble de séquences autorisés SA ; fin_si;*

Etape 2.2.3. *Si t est une transition du modèle du procédé et du modèle de la spécification et si t peut seulement être franchie dans le modèle du procédé alors la transition $M_i [t > M_n$ est ajoutée à l'ensemble des transitions d'état interdites T_F et σ de $M_0 [\sigma > M_n$ est ajouté à l'ensemble de séquences interdites de transitions d'état SI ; fin_si;*

Etape 2.2.4. *Si t est une transition du modèle du procédé et du modèle de la spécification et si t peut être franchie seulement dans ce dernier alors t ne doit pas être franchie et le marquage M_n n'est pas considéré comme un marquage légal ; fin_si;*

Etape 2.2.b *: Aller à l'étape 2.2.a*

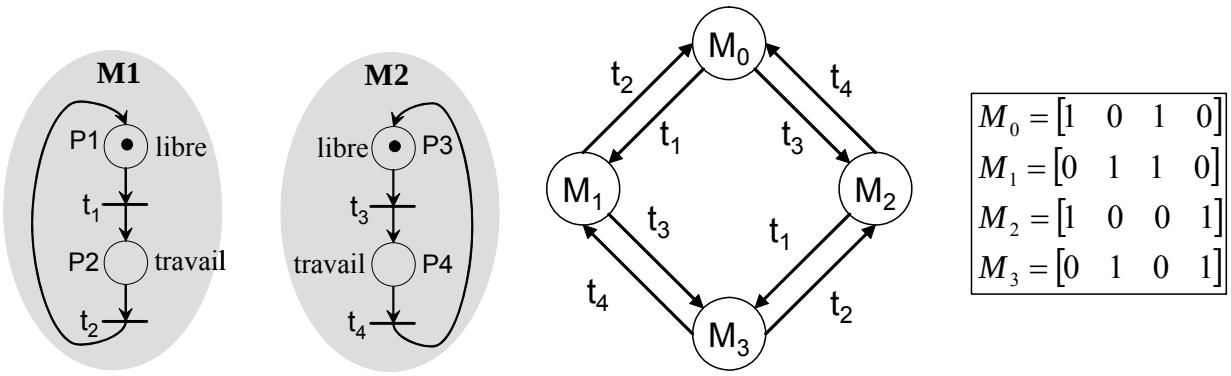
Etape 2.3. *Ajouter M_i à MT ;*

Fin_faire;

Fin.

Pour illustrer la transformation du PSITE en PTEI et le fonctionnement de l'algorithme de construction d'un GASC, nous considérons comme exemple, un système de

type producteur-consommateur (Une machine M1 productrice et une machine M2 consommatrice) dont le fonctionnement est contraint par un stock de capacité unitaire. Afin que cet exemple soit simple à comprendre, nous utilisons le modèle élémentaire de représentation du comportement d'une machine (Figure 30.A). Dans l'approche proposée, le concepteur doit ensuite traduire ses spécifications informelles en un ou plusieurs modèles RdP.



A. Modèle du procédé en boucle ouverte B. Graphe d'accessibilité en boucle ouverte

Figure 30. Modèles RdP du producteur-consommateur (M1 et M2)

Dans le problème du producteur consommateur, les spécifications consistent à contrôler le système de manière à ce que la machine 1 (M1) puisse travailler seulement quand le stock est vide et la machine 2 (M2) travaille seulement quand le stock contient une pièce sachant que la capacité du stock est d'une pièce. Comme il n'y a qu'un seul producteur, il n'est pas nécessaire que la machine effectue une réservation. Aussi, comme dans le modèle de spécification, ce qui est important ce sont les transitions et non pas les places, nous pouvons fusionner les places 'emplacements libres' et 'emplacements réservés' du modèle générique de la Figure 28 pour obtenir le modèle simplifié de la Figure 31. Cette simplification nous permettra de réduire la taille du graphe d'accessibilité construit.

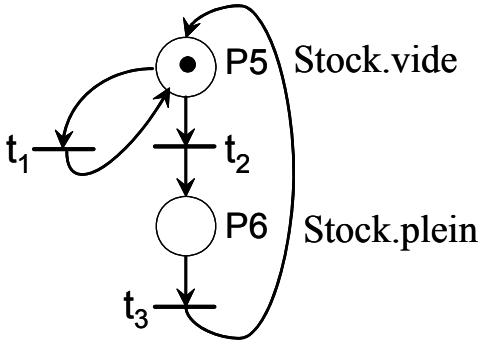


Figure 31. Modèle simplifié de spécification pour un stock de capacité unitaire

Le bouclage de la transition ‘t1’ sur la place P5 traduit donc que la machine M1 ne peut être mise en marche que si le stock est vide. Donc après la production d’une pièce, tant que la machine M2 n’aura pas récupéré cette pièce dans le stock, la machine M1 ne pourra pas être démarrée une nouvelle fois.

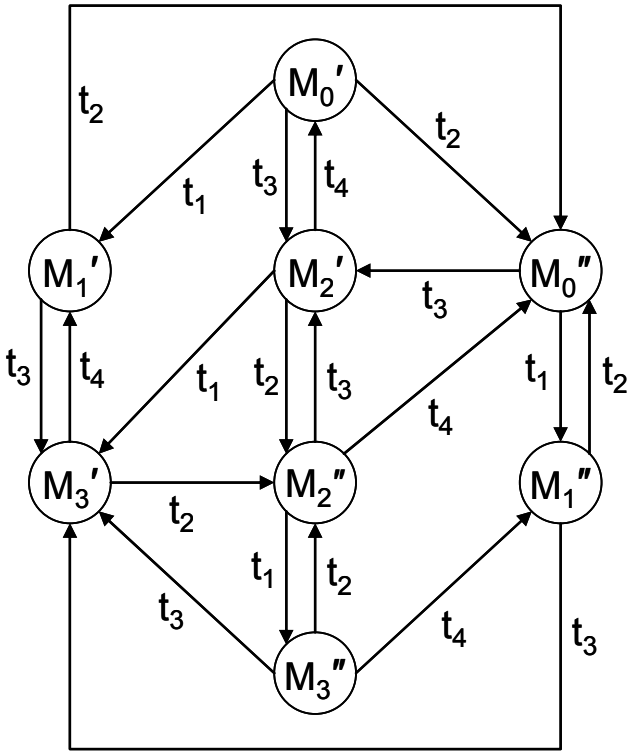
Afin de bien montrer la différence entre un graphe d’accessibilité standard et celui obtenu de manière synchrone, nous construisons le graphe d’accessibilité qui synthétise les trois RdP des Figure 30 et Figure 31. La Figure 32 illustre le graphe d’accessibilité synchrone du modèle global. Celui-ci est obtenu par le produit des sous-systèmes élémentaires entre le modèle du procédé et le modèle de spécification, en prenant en considération la synchronisation par les transitions communes des différents modèles. Chaque marquage est représenté par le vecteur $M = [P1 \quad P2 \quad P3 \quad P4 \quad P5 \quad P6]$.

Remarque : Afin de pouvoir faire la correspondre entre les graphes d’accessibilité du système en boucle ouverte et celui du système en boucle fermé, les indices des états après fusion correspondent aux indices des états avant fusion. Comme la spécification introduit crée deux états associé à l’état Mi du modèle en boucle ouverte, nous les nommerons respectivement M’0 et M’’0.

Le marquage initial est donc $M'_0 = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]$. A partir du marquage initial, trois transitions sont franchissables : ‘t_1’, ‘t_2’ et ‘t_3’. Par exemple, le franchissement de ‘t_1’ donne le marquage ‘M'_1’. Normalement, comme dans chacun des états nous avons

systématiquement trois places de marquées, trois transitions sont potentiellement franchissables. Pourtant, à partir de ' $M_1^{'}$ ' seules deux transitions sont franchissables à cause de la synchronisation due à la transition ' t_2 '.

Considérons à présent l'application de l'algorithme 1 pour la construction du GASC. Il est construit comme suit : au début, nous produisons le marquage initial $M_0^{'} = [1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]$ du modèle de synthèse (Étape 1) et nous l'ajoutons dans l'ensemble des marquages autorisés ' *MA* ' (Etape 1.1). A partir de $M_0^{'}$, normalement les transitions ' t_1 ', ' t_2 ' et ' t_3 ' peuvent être franchies. Mais, la transition ' t_2 ' ne peut pas être franchie par application de la Règle 2 (Etape 2.2.4). De même, la transition ' t_3 ' ne doit pas être franchie par application de la Règle 3 car si elle est franchissable dans le modèle du procédé, elle existe dans le modèle de la spécification mais elle n'y est pas franchissable. Donc la seule transition qui peut réellement être franchie est la transition ' t_1 '. Dans l'Étape 2.2.2, $M'_0 [t_1 > M'_1 = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0]$ est franchissable et donc M'_1 est ajouté à ' *MA* '. $M'_0 [t_3 > M'_2 = [1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$ n'étant pas franchissable la transition ' t_3 ' est ajoutée à l'ensemble interdit ' T_F ' par l'Étape 2.2.3.



$$M'_0 = [1\ 0\ 1\ 0\ 1\ 0]$$
$$M'_1 = [0\ 1\ 1\ 0\ 1\ 0]$$
$$M''_0 = [1\ 0\ 1\ 0\ 0\ 1]$$
$$M'_2 = [1\ 0\ 0\ 1\ 1\ 0]$$
$$M'_3 = [0\ 1\ 0\ 1\ 1\ 0]$$
$$M''_2 = [1\ 0\ 0\ 1\ 0\ 1]$$
$$M''_1 = [0\ 1\ 1\ 0\ 0\ 1]$$
$$M''_3 = [0\ 1\ 0\ 1\ 0\ 1]$$

Figure 32. Graphe d'accessibilité synchrone correspondant aux modèles de la Figure 30 et la Figure 31

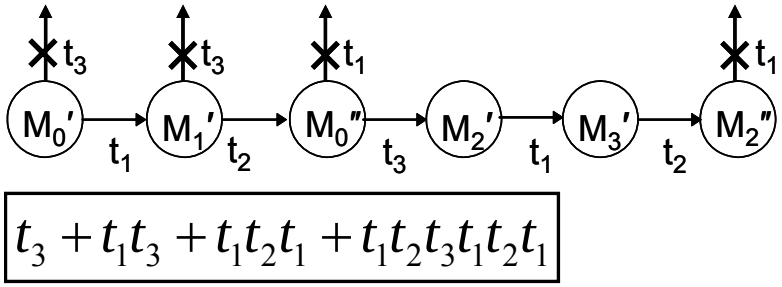
Par la suite, les transitions potentiellement franchissables à partir de 'M'_1' sont 't_2' et 't_3'. $M'_1[t_2 > M''_0 = [1\ 0\ 1\ 0\ 0\ 1]$ est franchissable donc M''_0 est ajouté à 'MA' par l'Étape 2.2.2 parce que 't_2' peut être franchie dans les deux modèles. $M'_1[t_3 > M'_3 = [0\ 1\ 0\ 1\ 1\ 0]$ n'est par contre pas franchissable et donc 't_3' est ajouté à 'T_F' par l'Étape 2.2.3. En poursuivant ce raisonnement, nous voyons que les transitions 't_1' et 't_3' peuvent être franchies depuis 'M''_0'. $M''_0[t_1 > M''_1 = [0\ 1\ 1\ 0\ 0\ 1]$ est ajouté à 'T_F' par l'Étape 2.2.3 tandis que $M''_0[t_2 > M'_2 = [1\ 0\ 0\ 1\ 1\ 0]$ est ajouté à 'MA' par l'Étape 2.2.2. Par la suite, $M'_0[t_3 > M'_2 = [1\ 0\ 0\ 1\ 1\ 0]$ est ajouté à l'ensemble 'T_F' et $M''_0[t_3 > M'_2 = M'_0[t_1 t_2 t_3 > M'_2 = [1\ 0\ 0\ 1\ 1\ 0]$ est ajouté à l'ensemble 'MA', conformément aux spécifications de l'utilisateur. Nous pouvons chercher les marquages accessibles à, partir de 'M'_2' parce que si $M'_0[t_3 > M'_2$ n'est pas un élément de l'ensemble autorisé 'MA', $M''_0[t_3 > M'_2$ est par contre un élément de 'M_A'. Les transitions 't_1' et 't_4'

par conséquent peuvent être franchies depuis ‘ M'_2 ’. $M'_2\left[t_1 > M'_3 = \left[0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0\right]\right.$ et $M'_2\left[t_4 > M'_0 = \left[1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0\right]\right.$ sont ajoutés à ‘ MA ’ par l’Étape 2.2.2.

Tableau 3. Les autorisations et les interdictions

SA : séquences autorisés	SI : séquences interdites de transitions d’état
- ($M^T{}_0 = \left[1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0\right]$)	
t_1	t_3
t_1t_2	t_1t_3
$t_1t_2t_3t_1t_2$	$t_1t_2t_1$
$t_1t_2t_3$	$t_1t_2t_3t_1t_2t_1$
$t_1t_2t_3t_1$	

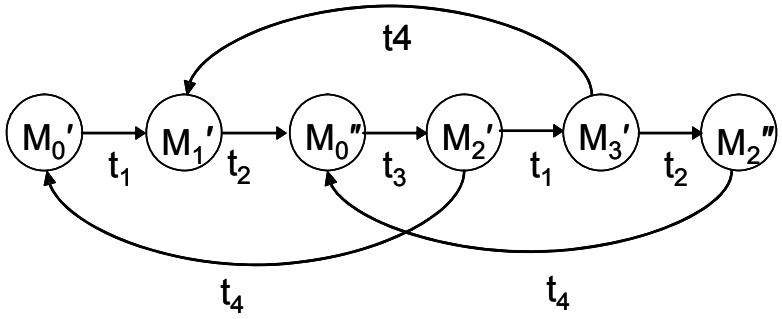
Ce scénario montre que nous pouvons trouver un cycle de transitions retournant au marquage initial : c’est l’ensemble des séquences définis par l’expression régulière (t1.t2.t3.t4)*. A cette équation de cycle, s’ajoute deux catégories de séquences résumées par le Tableau 3. Dans ce tableau, nous avons quatre séquences interdites de transition d’états (ou SITE) su point de vue du graphe d’accessibilité du procédé en boucle ouverte (Figure 30). Ces séquences sont définies à partir de l’état initial. Ainsi, la séquence ‘ t_3 ’ signifie qu’on interdit le franchissement de la transition ‘t3’ quand on est dans l’état initial. La séquence ‘t1.t3’, signifie que si on arrive dans l’état M1 à partir M0 en ayant franchit la transition ‘t1’, on s’interdit de franchir la transition ‘t3’ pour ne pas réaliser cette séquence interdite. Si on projette la séquence sur le graphe d’accessibilité de la Figure 32, on voit qu’en fait le problème de séquence interdite définie sur l’espace d’état du modèle du procédé en bouclez ouverte peut être résolu dans cet espace en un problème de transition interdite. Pour empêcher la séquence, notre algorithme supprime la possibilité de transition de M’$_1$ vers M’$_3$ (cf. Figure 33). En appliquant systématiquement ce principe, on voit que le graphe d’accessibilité synchrone contraint qui serait construit correspond à celui de la Figure 34.



$$t_3 + t_1 t_3 + t_1 t_2 t_1 + t_1 t_2 t_3 t_1 t_2 t_1$$

Figure 33. Illustration de la transformation du problème de séquences interdites en problème de transition interdites

Rappelons ici que notre objectif n'est pas la construction du graphe d'accessibilité synchrone contraint mais la détermination des équations exploitées par la théorie des régions.



$$\left(t_1 t_2 \left(t_3 t_1 \left(t_2 t_4 + t_4 t_2\right)\right)^* t_3 t_4\right)^*$$

Figure 34. Le graphe et la séquence de comportement souhaité

6. Synthèse à l'aide de la théorie des régions

Le problème de synthèse d'un réseau à partir de la donnée d'un automate, consiste à déterminer si cet automate est isomorphe au graphe d'accessibilité correspondant du RdP. Ce problème est résolu dans la littérature pour des réseaux de types divers allant des réseaux simples aux réseaux de Petri. La finalité est de déterminer les RdP purs (N, M_0), caractérisés par une matrice d'incidence 'C' et ayant un ensemble de transitions 'T'. Les transitions de 'T'étiquettent les arcs de 'G' l'automate qui est sensé être isomorphe au graphe d'accessibilité du RdP recherché.

Au départ, la théorie des régions a été développée pour synthétiser des RdP purs pour des systèmes de transitions donnés. Dans [BAD95], les auteurs proposent un algorithme qui résout ce problème de synthèse de RdP purs à partir d'automates finis. Cet algorithme opère en temps polynomial par rapport à la taille de l'automate pris en entrée.

A. Ghaffari a proposé une nouvelle application de la théorie des régions. Etant donné un RdP modélisant le comportement du procédé en boucle ouverte, l'objectif est de déterminer des places de contrôles à ajouter au modèle initial pour l'obtention du modèle en boucle fermé respectant les spécifications de l'utilisateur [GHA02b]. Nous commençons cette partie par une présentation de l'approche d'A. Ghaffari. Ensuite nous exploiterons cette théorie pour la résolution des équations du GASC déterminées à l'aide de l'algorithme 1.

6.1 La théorie des régions [GHA02b]

L'approche de théorie des régions d'A Ghaffari est formalisée à l'aide du théorème suivant :

Théorème 3: Etant donné G un automate à états finis, il existe un réseau de Petri (N, M_0) l'ayant pour graphe d'accessibilité si et seulement si il existe un ensemble P de places $\{(M_0(p), C(p,.))\}$ tel que :

1. Chaque place P de (N, M_0) satisfait les équations des cycles (2.1) de G :

$$\sum_{t \in T} C(p,t)\vec{\delta}[t] = 0\,,\ \forall \delta \in \Delta\,, \qquad (2.1)$$

où $\vec{\delta}[t]$ désigne la somme algébrique de toutes les occurrences de t dans δ et Δ est l'ensemble des cycles non orientés de G ;

2. Chaque place P de (N, M_0) vérifie les conditions d'accessibilité (2.2) de marquage de G :

$$M(p) + C(p,.)\vec{\Gamma}_M \geq 0\,,\ \forall M \in R\,; \qquad (2.2)$$

3. Pour chaque couple (M,t) où t n'est pas franchissable à partir de M ; il existe au moins une place de contrôle p_c qui satisfait la condition de séparation d'événement (2.3) :

$$M(p_c) = M_0(p_c) + C(p_c,.)\vec{\Gamma}_M + C(p_c,t) < 0 . \qquad (2.3)$$

Supposons que, pour chaque nœud du graphe, il existe au moins un chemin non orienté le reliant à chacun des autres nœuds de ' G '. Soit ' p ' une place quelconque du RdP (N, M_0) à déterminer. Une place ' p ' est entièrement déterminée par le vecteur de ses coordonnées $C(p,.)$ dans la matrice d'incidence ' C ' de (N, M_0). Elle doit donc vérifier l'équation (2.4) pour tous les marquages ' M ' appartenant à R(N, M_0) et toutes les transitions ' t ' validées par le marquage ' M ' ; c'est-à-dire que ' t ' est l'étiquette d'un arc sortant du nœud ' M ' :

$$M'(p) = M(p) + C(p,t), \ \forall (M, M') \in G \text{ tel que } M[t > M', \qquad (2.4)$$

où ' M' 'est le nœud destination de l'arc étiqueté par ' t '.

Considérons à présent un cycle non-orienté quelconque ' δ ' de ' G '. ' $\vec{\delta}$ ' est le vecteur d'occurrences de transitions de 'T' dans ' δ '. L'application de l'équation (2.4) le long du cycle conduit à la relation de l'équation (2.1) qui est une équation de cycle.

D'autre part, pour chaque nœud ' M ' du graphe ' G ', il existe au moins un chemin non orienté le reliant à chacun des autres nœuds du graphe. Soit, en particulier, ' Γ_M ' un chemin arbitraire joignant ' M_0 ' à ' M '. En appliquant l'équation (2.4) à tous les arcs de ' Γ_M ' on obtient :

$$M(p) = M_0(p) + C(p,.)\vec{\Gamma}_M \qquad (2.5)$$

avec ' $\vec{\Gamma}_M$ ' le vecteur d'occurrences de ' Γ_M ' défini comme ' $\vec{\delta}$ '. Le marquage ' M ' est atteignable dans ' G ' s'il vérifie l'équation (2.2). Cette équation traduit la condition d'accessibilité de ' M '. Une place ' p ' du réseau recherchée (N, M_0) doit donc nécessairement vérifier les équations de cycle et les conditions d'accessibilité induites par la structure du graphe ' G '. Mais ces conditions ne permettent pas d'obtenir ' G ' comme graphe d'accessibilité. Une condition supplémentaire implique que, pour chaque couple (M,t) tel que ' M ' est un état de ' G ' et ' t ' une transition non validée par ' M ', ' t ' soit inhibée par le

marquage d'une certaine place ' p '. Puisque le réseau est supposé pur, cette condition est réalisée si et seulement si l'équation (2.3) est vérifiée pour au moins une place ' p '.

La relation (2.3) est appelée condition de séparation d'événement relative au couple (M,t) . L'ensemble des couples (M,t) où ' M ' est accessible et ' t ' est non validée par M est appelé l'ensemble des instances de séparation d'événement.

Par la suite, nous allons appliquer cette approche de la théorie des régions pour déterminer les places de contrôle à ajouter au réseau de Petri initial pour inhiber les séquences interdites de transitions d'état.

6.2 Résolution des équations obtenues par la théorie des régions

Pour expliquer cette résolution, nous nous appuyons sur l'exemple du producteur-consommateur de la Figure 30.

Chaque place de contrôle (que nous noterons 'p_c') ajoutée doit vérifier les 3 équations définies par le Théorème 3. Nous pouvons commencer par établir les équations de cycles en utilisant le graphe de la Figure 33.A. Le graphe d'accessibilité modélisant la synthèse du modèle du procédé et des spécifications contient trois cycles : le cycle 1 est caractérisé par la séquence 't1*t2*t3*t4' obtenue à partir du marquage ' M_0 ' ; le cycle 2 est caractérisé par la séquence 't2*t3*t1*t4' obtenue à partir du marquage ' M_2 ' ; et le cycle 3 est caractérisé par la séquence 't3*t1*t2*t4' qui est également obtenue à partir du marquage ' M_2 '. Ces cycles étant composés du même ensemble de transitions avec le même nombre d'occurrences pour chacune, ils se traduisent donc par la même équation de cycle :

$$C(p_c,t1)+C(p_c,t2)+C(p_c,t3)+C(p_c,t4)=0 \qquad (2.6)$$

Ensuite, nous appliquons l'équation (2.2) du Théorème 3 pour obtenir les conditions d'accessibilité des marquages autorisés. Par rapport à notre exemple illustratif, ces conditions sont calculées à partir des séquences souhaitées déterminées par l'algorithme 1 et stockée dans moitié gauche du Tableau 3. Nous obtenons les équations suivantes :

$$\left.\begin{array}{l} a.M_0(p_c) \geq 0 \\ b.M_1(p_c) = M_0(p_c) + C(p_c, t1) \geq 0 \\ c.M_2(p_c) = M_0(p_c) + C(p_c, t1) + C(p_c, t2) \geq 0 \\ d.M_3(p_c) = M_0(p_c) + C(p_c, t1) + C(p_c, t2) + C(p_c, t3) \geq 0 \\ e.M_4(p_c) = M_0(p_c) + C(p_c, t1) + C(p_c, t2) + C(p_c, t3) + C(p_c, t1) \geq 0 \\ f.M_5(p_c) = M_0(p_c) + C(p_c, t1) + C(p_c, t2) + C(p_c, t3) + C(p_c, t1) + C(p_c, t2) \geq 0 \end{array}\right\} \quad (2.7)$$

L'équation (2.7a) exprime que le marquage initial des places de contrôle doit être positif ou nul comme le marquage de toutes places dans un RdP ordinaire. Les équations (2.7b) à (2.7f) traduisent effectivement les séquences autorisées. Par exemple, l'équation (2.7b) exprime que la transition 't1' doit être franchissable à partir du marquage initial pour atteindre le marquage ' M_1 '.

Finalement, les séquences interdites de séquences de transition d'états sont formulées à l'aide des séquences stockées dans la partie droite du Tableau 3. Il y a donc quatre équations de séparation d'événements à satisfaire. Leur résolution donnera au plus quatre nouvelles places de contrôle ' p_c ', à savoir :

$$M_0(p_{c1}) + C(p_{c1}, t3) < 0 \quad (2.8)$$

$$M_0(p_{c2}) + C(p_{c2}, t1) + C(p_{c2}, t3) < 0 \quad (2.9)$$

$$M_0(p_{c3}) + C(p_{c3}, t1) + C(p_{c3}, t2) + C(p_{c3}, t1) < 0 \quad (2.10)$$

$$M_0(p_{c4}) + C(p_{c4}, t1) + C(p_{c4}, t2) + C(p_{c4}, t3) + C(p_{c4}, t1) + C(p_{c4}, t2) + C(p_{c4}, t1) < 0 \quad (2.11)$$

Ces équations spécifient les conditions d'inhibition des séquences interdites de transition d'état. L'équation (2.8) signifie qu'une place de contrôle ' p_c ' doit inhiber la possibilité de franchissement de la transition ' t_3 ' à partir du marquage initial. L'équation (2.9) exprime que la séquence (t1.t3)* n'est pas permise car dans ce cas le stock serait vide lors du démarrage de la machine 2. L'équation (2.10) spécifie que la transition ' t_1 ' ne doit pas être franchissable après un cycle limité à la machine 1 (M1). Elle correspond en fait aux séquences définies par (t1.t2)*. La dernière équation (2.11) signifie qu'en cas de fonctionnement en parallèle des deux machines, si la machine 1 a produit une nouvelle pièce

avant que la machine 2 n'est terminée sa pièce précédente, elle ne doit pas être remise en production avant que sa dernière pièce n'est été retirée du stock.

Comme nous choisissons ici de trouver une solution basée sur les RdP ordinaires, toutes les places de contrôle ajoutées doivent vérifier la relation (2.12). Cette équation traduit le poids des arcs liant ces places aux différentes transitions du modèle de synthèse.

$$\forall t \in T \quad C(p_c, t) \in \{-1, \quad 0, \quad 1\} \qquad (2.12)$$

Remarque : Travailler dans les RdP ordinaires permet de réduire l'espace de recherche. Mais toutes les synthèses de contrôleurs non pas forcément de solutions dans les RdP ordinaires. Donc en cas d'échec, il faut relaxer cette contrainte en cherchant une solution dans les RdP généralisés.

Comme les transitions ' t_2 ' et ' t_4 ' sont définies comme incontrôlables, l'équation (2.12) nous permet de poser :

$$C(p_c, t2) \geq 0 \qquad (2.13)$$

$$C(p_c, t4) \geq 0 \qquad (2.14)$$

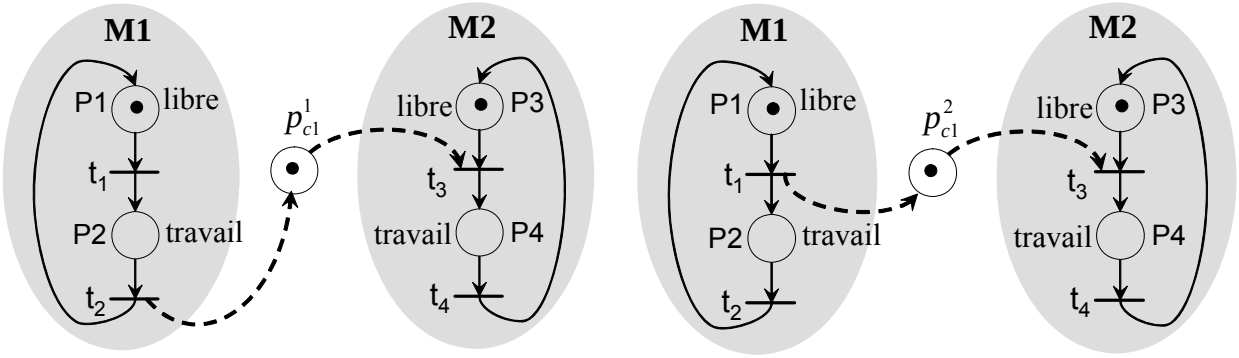
Pour effectuer la résolution le principe consiste à considérer itérativement chacune des équations de séparation d'événement avec toutes les autres équations. Nous raisonnons donc au départ comme si nous avions à déterminer 4 places de contrôles.

A. Calcul de ' p_{c1} '

Ce calcul est fait sur la base de l'équation (2.6), de l'équation (2.7), de l'équation (2.8) et de l'équation (2.12). En raison de l'équation (2.12), il n'existe qu'une seule possibilité pour vérifier l'équation (2.8), $M_0(p_{c1}) + C(p_{c1}, t3) < 0$. Il faut imposer que $M_0(p_{c1}) = 0$ et $C(p_{c1}, t_3) = -1$. Ensuite, on applique $M_0(p_{c1}) = 0$ et $C(p_{c1}, t_3) = -1$ à l'équation (2.7). Dans ce cas, trois connexions de réseaux sont possibles pour la place de contrôle ' p_{c1} ' :

- $C(p^1_{c1}, .) = (0 \quad 1 \quad -1 \quad 0)$: Cette solution définit une place qui contrôle que la transition ' t_3 ' n'est franchissable que si le franchissement de ' t_2 'a déjà eu lieu (Figure 35.A).

- $C\left(p_{c1}^{2},.\right)=\begin{pmatrix}1 & 0 & -1 & 0\end{pmatrix}$: Cette solution définit une place qui contrôle que la transition ‘ t_3 ’ n’est franchissable que si le franchissement de ‘ t_1 ’ a déjà eu lieu (Figure 35.B).
- $C\left(p_{c1}^{3},.\right)=\begin{pmatrix}1 & 1 & -1 & -1\end{pmatrix}$: Cette solution n’est pas possible car elle est contraire à l’équation (3.14), la transition ‘ t_4 ’ ayant été définies comme incontrôlable.
-



A. Place de contrôle ‘ P_{c1}^{1} ’ B. Place de contrôle ‘ P_{c1}^{2} ’

Figure 35. Connexions de la place de contrôle **‘** p_{c1} **’** au modèle du procédé

B. Calcul de ‘ p_{c2} **’**

Pour calculer la place de contrôle ‘ p_{c2} ’, nous considérons l’équation (2.6), l’équation (2.7), l’équation (2.9) et l’équation (2.12). La recherche de solutions relatives aux équations (2.7b), (2.7c), (2.9) et (2.12) entraîne que $M_0\left(p_{c2}\right)+C\left(p_{c2},t1\right)=0$, $C\left(p_{c2},t3\right)=-1$ et $C\left(p_{c2},t2\right)\geq 0$. Ensuite, nous pouvons déterminer les valeurs des autres paramètres de la place. Quatre solutions sont envisageables en fonction des valeurs de $M_0\left(p_{c2}\right)=0$ et de $C\left(p_{c2},t2\right)$:

- $M_0\left(p_{c2}\right)=0$ et $C\left(p_{c2},t2\right)=0$: Dans ce cas $C\left(p_{c2},t1\right)=0$ et $C\left(p_{c2},t4\right)=1$. Nous rejetons cette solution car en raison du marquage initial de ‘pc2’, elle rend impossible le franchissement de la transition ‘ t_3 ’ ;

- $M_0(p_{c2})=1$ et $C(p_{c2},t2)=0$: Dans ce cas, cela entraîne que $C(p_{c2},t1)=-1$. En conséquence, l'équation de cycle (3.6) ne peut plus être vérifiée. Donc ce cas également ne donne aucune solution;
- $M_0(p_{c2})=0$ et $C(p_{c2},t2)=1$: Dans ce cas $C(p_{c2},t1)=0$ et $C(p_{c2},t4)=0$. Nous obtenons donc la solution $C(p_{c2}^1,.)=(0 \quad 1 \quad -1 \quad 0)$ qui est analogue à la solution obtenue par p_{c1}^1 (Figure 35.A) ;
- $M_0(p_{c2})=1$ et $C(p_{c2},t2)=1$: Dans ce cas $C(p_{c2},t1)=-1$ et $C(p_{c2},t4)=1$. Nous obtenons donc la solution $C(p_{c2}^2,.)=(-1 \quad 1 \quad -1 \quad 1)$. Cette place de contrôle servirait en fait à garantir que les deux machines ne fonctionnent pas simultanément. Cela ne correspond pas vraiment à notre spécification. Néanmoins conservons cette solution partielle. Nous verrons ultérieurement que la seconde étape de la résolution l'élimine pour sélectionner la première solution.

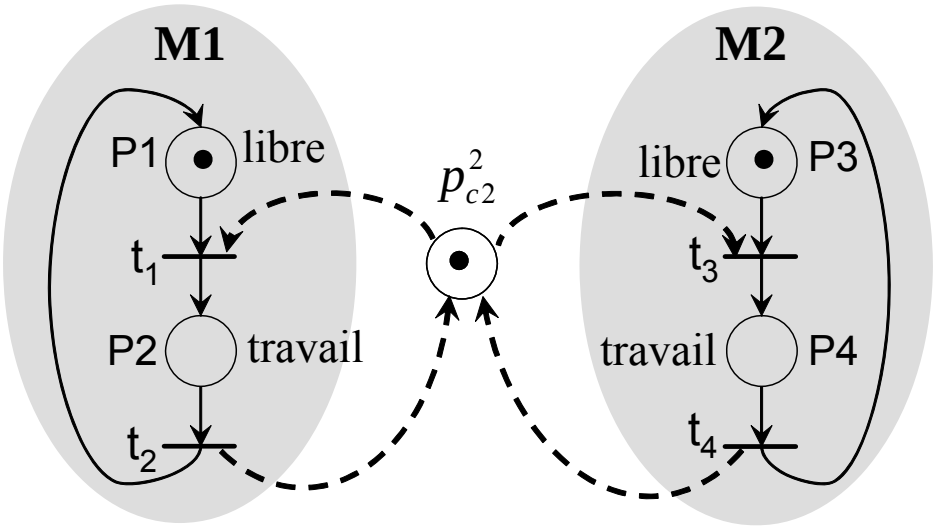


Figure 36. Contrôleur correspondant à la solution p_{c2}^2

C. Calcul de ' p_{c3} '

Pour déterminer ' p_{c3} ', nous considérons d'abord les équations (2.10), (2.6), (2.7c) et (2.12). Nous obtenons alors $M_0(p_{c3})+C(p_{c3},t2)=1$ et $C(p_{c3},t1)=-1$. Nous intégrons

ensuite ces résultats dans l'équation (2.7e), ce qui entraîne que $C(p_{c3},t3)=1$. La résolution de l'équation $M_0(p_{c3})+C(p_{c3},t2)=1$, offre trois possibilités :

- si $M_0(p_{c3})=0$ et $C(p_{c3},t2)=1$: ce cas n'est pas possible car l'équation (2.7b) et $C(p_{c3},t1)=-1$ imposent $M_0(p_{c3})\geq 1$;
- si $M_0(p_{c3})=1$ et $C(p_{c3},t2)=0$: ce cas impose que $C(p_{c3},t4)=0$ afin de pouvoir vérifier l'équation de cycle (2.6) ; il correspond donc à une place de contrôle raccordée par les connexions définies par $C(p_{c3}^1,.)=(-1 \quad 0 \quad 1 \quad 0)$;
- si $M_0(p_{c3})=2$, $C(p_{c3},t2)=-1$: ce cas n'est pas possible car il est contradictoire avec l'équation (2.13) qui impose de ne pas contrôler la transition 't_2' qui est incontrôlable par définition.

En conclusion la résolution du système d'équations intégrant l'équation de séparation d'événement (2.10) aboutit à une place de contrôle initialement marquée qui conditionne le fait que la transition 't_1' ne peut être franchie que s'il y a eu au préalable un franchissement de 't_3' (cf. Figure 37).

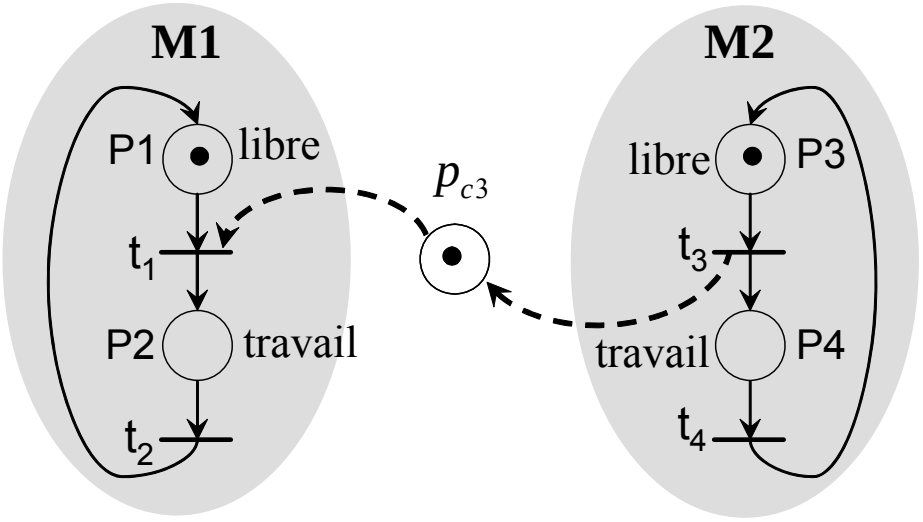


Figure 37. Place de contrôle correspondant à la résolution du système relatif à la place ' p_{c3} '

D. Calcul de ‘ p_{c4} ’

Finalement, nous déterminons ‘ p_{c4} ’ en utilisant les équations (2.11), (2.6), (2.7) et (2.12). Par les équations (2.11), (2.7f) et (2.12), nous établissons que $M_0(p_{c4})+2C(p_{c4},t2)+C(p_{c4},t3)=2$, $C(p_{c4},t1)=-1$ et $M_0(p_{c4})\geq 1$. Ensuite, nous intégrons ces résultats partiels avec les équations (2.7) et (2.6). Nous déterminons ainsi trois possibilités de connexions de réseaux pour la place de contrôle ‘ p_{c4} ’ :

- $C(p^1_{c4},.)=(-1 \quad 0 \quad 1 \quad 0)$ avec $M_0(p^1_{c4})=1$: cette solution est identique à p^1_{c3}
- $C(p^2_{c4},.)=(-1 \quad 0 \quad 0 \quad 1)$ avec $M_0(p^2_{c4})=2$: cette solution permet de franchir deux fois la transition ‘ t_1 ’ avant qu’il ne soit nécessaire de franchir la transition ‘ t_4 ’. Elle est conforme à l’équation (3.11) mais elle ne permet pas de contrôler les autres séquences interdites ;
- $C(p^3_{c4},.)=(-1 \quad -1 \quad 1 \quad 1)$ avec $M_0(p^3_{c4})=3$: cette solution nécessite le contrôle de la transition ‘ t_2 ’ qui a été définie comme incontrôlable ; elle est donc rejetée.

Pour déterminer à partir de ces résultats partiels les places du contrôleur optimal, nous privilégions les solutions vérifiant le plus grand nombre d’équations de séparations d’événements [LEE05b] (Figure 38).

a. $M_0(p^1_{c1})=0$	$C(p^1_{C1},.)=[0 \quad 1 \quad -1 \quad 0]$	c. $M_0(p_{c3})=1$	$C(p_{C3},.)=[-1 \quad 0 \quad 1 \quad 0]$
$M_0(p^2_{c1})=0$	$C(p^2_{C1},.)=[1 \quad 0 \quad -1 \quad 0]$		
b. $M_0(p^1_{c2})=0$	$C(p^1_{C2},.)=[0 \quad 1 \quad -1 \quad 0]$	d. $M_0(p_{c4})=1$	$C(p_{C4},.)=[-1 \quad 0 \quad 1 \quad 0]$
$M_0(p^2_{c2})=1$	$C(p^2_{C2},.)=[-1 \quad 1 \quad -1 \quad 1]$		

Figure 38. Synthèse des places de contrôles déterminées

Après cela, l’étape suivante consiste à chercher le contrôleur commun aux différents résultats partiels. Le contrôleur $C(p^1_{c1},.)=(0 \quad 1 \quad -1 \quad 0)$ contrôle le comportement interdit de l’équation (2.8) et le contrôleur $C(p^1_{c2},.)=(0 \quad 1 \quad -1 \quad 0)$ contrôle le comportement interdit de

l'équation (2.9). Donc le contrôleur $\begin{pmatrix}0 & 1 & -1 & 0\end{pmatrix}$ est sélectionné car il est permet d'interdire deux séquences : ' t_3 ' et ' $t_1 t_3$ '. Une même analyse donne le contrôleur $C\left(p_{c3}^1,.\right) = C\left(p_{c4}^1,.\right) = \begin{pmatrix}-1 & 0 & 1 & 0\end{pmatrix}$. Finalement, deux places de contrôle sont suffisantes pour obtenir un comportement du système en boucle fermé conforme aux spécifications de l'utilisateur (Figure 39).

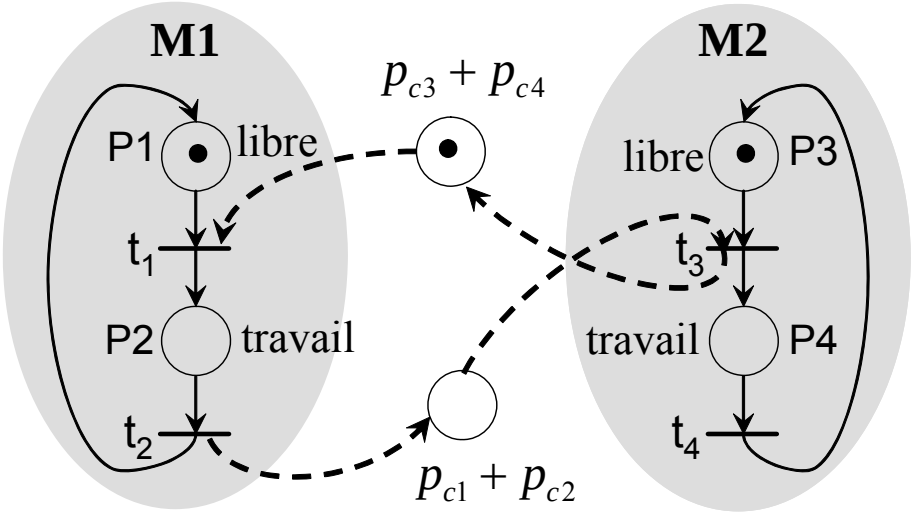


Figure 39. Modèle de comportement du procédé en boucle fermé

Notons ici que le choix de retenir les places de contrôle qui vérifient plusieurs équations de séparation est arbitraire. Il permet juste de minimiser le nombre de place de contrôle et donc de simplifier les problèmes d'implémentation. Tout autre choix de places de contrôle reste une solution valable. Dans notre exemple, les deux premières équations ayant chacune une solution distinct de la solution commune, nous avons deux autres solutions possibles à notre problème.

7. Conclusion

Dans ce chapitre, nous avons introduit un nouveau type de problème de pour la spécification du comportement souhaité d'un système à l'aide des Réseaux de Petri. Il s'agit du Problème des Séquences Interdites de Transitions d'Etats (PSITE). Nous avons montré que ce problème est un vrai problème de spécification, dans la mesure où lorsqu'on considère le

modèle du procédé en boucle ouverte, on ne peut le réduire à l'un des problèmes définis dans l'état de l'art du chapitre I. Pour traiter ce problème, nous avons proposé de le ramener à un problème d'interdiction de transition d'états dans l'espace du modèle du procédé fusionné de manière synchrone avec le modèle de spécification. Cela permet alors d'utiliser la théorie des régions pour calculer les places de contrôle à ajouter au modèle initial pour obtenir le comportement en boucle fermé. Nous avons ainsi proposé un algorithme qui permet l'extraction des équations de la théorie des régions. Notons ici que dans notre approche, la spécification permet uniquement de spécifier des contraintes de fonctionnement. Certains problèmes peuvent plus facilement se traduire par spécifications en termes de comportement non souhaité par rapport à un comportement possible. Il faudrait trouver une méthode de modélisation de ces comportements interdits en se basant sur une approche RdP.

Dans le prochain chapitre nous allons appliquer les résultats de ce chapitre à la synthèse de la commande coordination d'un système manufacturier.

Chapitre III : Application de la synthèse RdP à la conception de la commande d'un système manufacturier

1. Introduction

Dans ce chapitre, notre objectif est d'appliquer l'approche de synthèse proposée au chapitre II, à la conception d'une commande reconfigurable pour la conduite des systèmes manufacturiers. L'approche proposée est basée sur la méthodologie CASPAIM qui décompose la commande de coordination des systèmes manufacturiers en deux parties (cf. chapitre I) : la commande de coordination du système de transport et la commande des produits.

La commande de coordination du système de transport perçoit le système sous la forme d'un ensemble de ressources de transformation interagissant par le biais de stocks intermédiaires. Les ressources de transformations sont chargées ou déchargés en produits par ressources de transport qui peuvent stocker temporairement ces produits dans des stocks. Dans ce contexte, la prise en compte de l'ensemble des contraintes introduites par les différents stocks pour construire un modèle monolithique s'avère trop complexe et contraire aux exigences en modularité d'un système reconfigurable (cf. paragraphe I.2 au chapitre I). Dans ce chapitre, nous allons donc nous intéresser dans un premier temps à la proposition d'une approche modulaire basée sur des modèles entièrement RdP.

La commande des produits est basée sur le concept de gammes opératoires. Actuellement dans l'approche CASPAIM (cf. paragraphe I.4.1.4 du chapitre I), nous proposons des gammes opératoires dont l'exécution en exploitation n'est pas déterministe. L'indéterminisme résiduel est dû à la prise en compte de la flexibilité au niveau du codage des gammes opératoires. Notre objectif est d'étudier dans quel contexte nous pouvons obtenir une

exécution déterministe de la commande des produits en construisant des allocateurs de ressources mettant en œuvre l'ordonnancement prévisionnel de la production considérée.

2. Approche RdP et modularité pour la synthèse de la commande du système de transport d'un système manufacturier

2.1 Introduction

L'une des limites de la théorie du « supervisory control » est la difficulté à appréhender des procédés réels en raison de la complexité inhérente à la taille de ces systèmes. Pour y faire face, il existe plusieurs propositions permettant de combattre l'explosion combinatoire due à la manipulation d'automates à états finis. Deux catégories d'approches sont proposées dans la littérature : les approches modulaires ou décentralisées et les approches hiérarchiques (Figure 40). Dans cette partie, nous cherchons à adapter de telles approches à la synthèse de la commande à l'aide des RdP.

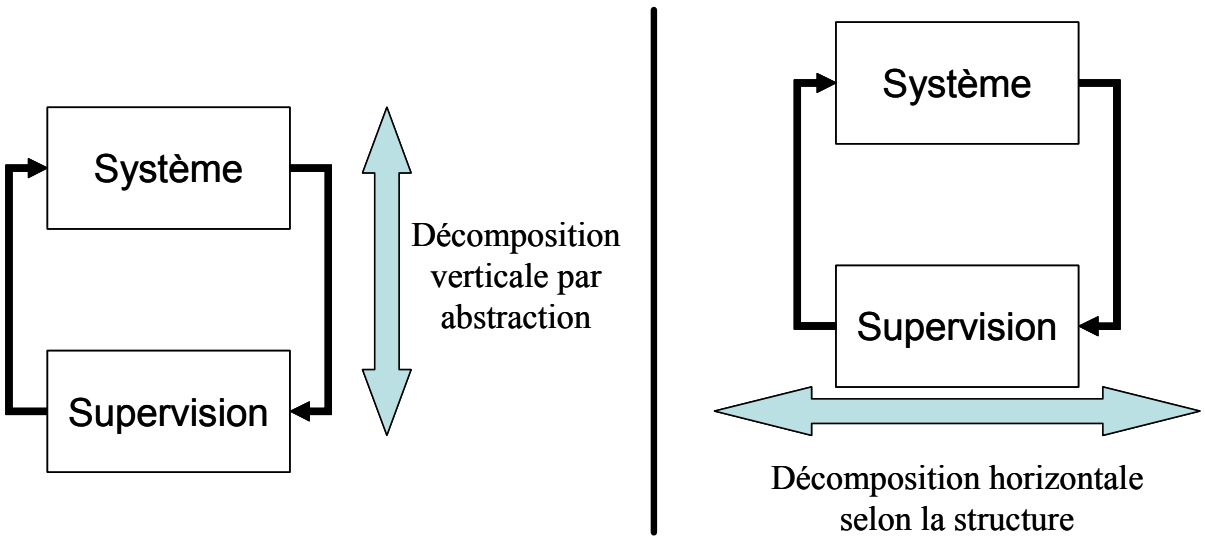


Figure 40. Schémas de décomposition

2.2 Supervision modulaire ou décentralisée de SED

La décomposition horizontale du contrôle supervisé d'un SED peut être réalisée soit par une supervision modulaire de SED (Figure 41), soit par une supervision décentralisée (Figure 42). La supervision modulaire de SED consiste à décomposer un superviseur monolithique en plusieurs superviseurs agissant en parallèle sur l'ensemble du procédé. Chaque module superviseur observe l'ensemble des événements produits par le procédé. L'objectif de cette approche est de réduire la complexité d'obtention des superviseurs tout en garantissant que l'ensemble applique un contrôle aussi permissif que celui qu'aurait appliqué un superviseur monolithique. Il faut notamment garantir que tous les modules superviseurs exercent sur le procédé des contrôles qui ne soient pas mutuellement incompatibles. Pour cela, il est nécessaire de vérifier que les superviseurs modulaires respectent des conditions préalables : contrôlables, non-bloquants et non-conflictuels.

Définition 20 : Langages non-conflictuels [WON88]

Soient deux langages L1, L2 ™ ∑*. Ils sont dits non conflictuels ou modulaires si $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$.

La définition précédente signifie que si deux langages partagent un préfixe commun ($\overline{L_1} \cap \overline{L_2}$), ils doivent partager un mot contenant ce préfixe ($\overline{L_1 \cap L_2}$).

Définition 21 : Système non-bloquant

Etant donné un système G, il est non-bloquant si $L_m(G) = L(G)$

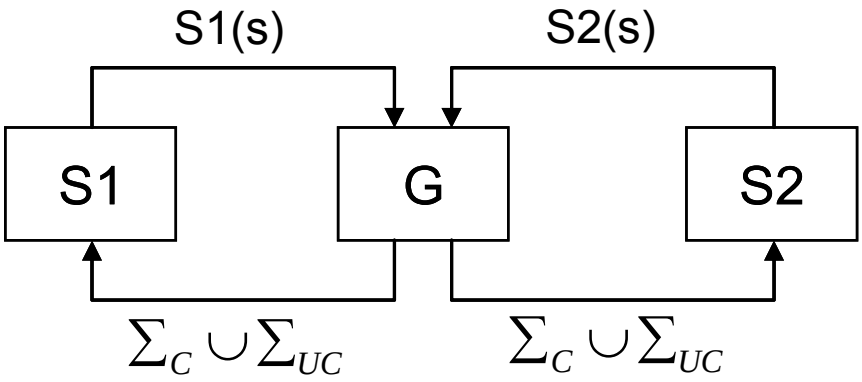


Figure 41. Supervision modulaire de SED

Proposition 1 [RAM88] : Etant donné deux superviseurs non-bloquants S1 et S2 d'un procédé G. Le superviseur S1∧S2 est non-bloquant si et seulement si les langages $L_m(S_1/G)$ et $L_m(S_2/G)$ sont non-conflictuels.

La proposition 1 donne une condition nécessaire pour que des superviseurs distincts appliqués à un même procédé, exercent des contrôles compatibles vis-à-vis de ce procédé.

Mais l'objectif pratique de la modularité est de pouvoir construire des superviseurs modulaires à partir de la donnée de spécifications distincts. Il faut que les spécifications en question soient compatibles pour que les contrôleurs résultants le soient.

Théorème 4 [WON88] : Etant donné deux spécifications K1, K2 ™ ∑*. Si SupC(K1, G) et SupC(K2,G) sont des superviseurs non-conflictuels de G, alors SupC(K1, G)∩SupC(K2,G)=SupC(K1∩K2, G).

Ce théorème nous indique que si à partir de spécifications données par l'utilisateur nous pouvons construire des superviseurs non-conflictuels, le contrôle exercé par l'ensemble de ces superviseurs est optimal car égal au contrôle que l'on aurait obtenu à l'aide d'un unique superviseur monolithique construit à partir de l'intersection de l'ensemble des spécifications.

Comme l'approche modulaire, la supervision décentralisée de SED a pour objectif de réduire la complexité relative à la construction d'un superviseur pour un procédé de taille importante [LIN88]. Son principe consiste donc à décomposer un procédé G en sous procédés virtuel Gi correspondant chacun à une spécification Ei donnée (i.e. $G = G_1 \| G_2 \ldots \| G_n$). L'idée est donc étant donné Ei de construire un superviseur Si qui permet le contrôle de Gi. Pour cela, deux questions doivent être résolus : comment définir Gi et peut-on garantir que le contrôle assuré par l'ensemble des superviseurs locaux Si est équivalent au contrôle exercé par un superviseur monolithique sur le procédé G ?

Pour définir Gi, on peut partir de la définition de Ei. L'idée de la supervision décentralisée est de travailler sur Σ_i un alphabet restreint par rapport à Σ l'alphabet du procédé global G. L'alphabet Σ_i est constitué des événements générés par tous les composants de G concernés par la spécification Ei. Ces composants constituent le procédé virtuel Gi qui est tel que $G = G_1 \| G_2 \ldots \| G_n$ et qui est donc défini sur l'alphabet local $\Sigma_i / \Sigma = \Sigma_1 \cup \Sigma_2 \ldots \cup \Sigma_n$.

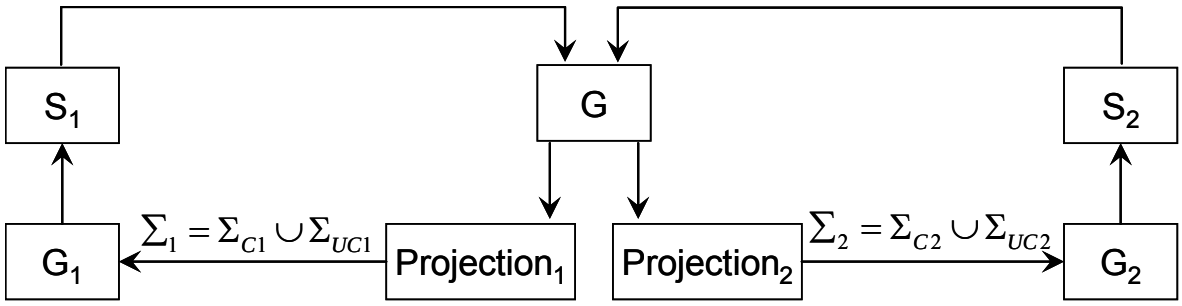


Figure 42 Supervision décentralisée de SED

De manière formelle, pour définir Gi on introduit la notion de projection. Soit $P_i : \Sigma^* \to \Sigma_i^*$ une projection naturelle. D'un point de vue pratique, dans toute chaîne appartenant à Σ^*, Pi efface les événements de $\Sigma - \Sigma_i$. De ce fait Gi est défini comme une projection de G sur $\Sigma_i /\ L(G_i) = P_i(L(G))$ (Figure 42).

Le superviseur local S_i est alors défini tel que $L(S_i / G_i) = SupC\left(E_i \cap P_i(L(G))\right)$. Pour déterminer si ce superviseur est optimal, il faut le comparer au superviseur $\widetilde{S}_i$ construit sur

l'alphabet Σ / $L(\widetilde{S}_i / G) = P_i^{-1}(L(S_i / G_i)) \cap L(G)$ avec P_i^{-1} la projection inverse de P_i . Le superviseur $\widetilde{S}_i$ (dit superviseur local étendu) est construit en prenant comme spécification le comportement en boucle fermé de Gi sous le contrôle de Si. Aussi par construction, le contrôle conjoint de l'ensemble des superviseurs locaux étendus $\widetilde{S}_i$ est équivalent à un superviseur monolithique obtenu en centralisé / $L(\wedge_{i=1..n} \widetilde{S}_i / G) = SupC((\wedge_{i=1..n} E_i) \cap L(G))$. Dans [LIN88], il est prouvé que S_i est optimal si $L(S_i / G_i) = L(\widetilde{S}_i / G)$.

Les deux approches que nous avons présentées ici ont pour objectif de réduire la complexité (en raison de l'explosion combinatoire) de la construction d'un superviseur monolithique. L'approche décentralisée semble plus prometteuse en termes de réduction de la complexité dans la mesure où à la fois la spécification et le procédé considéré sont restreints. Néanmoins, les deux approches nécessitent la vérification de condition a posteriori permettant de garantir l'optimalité du contrôle proposé par rapport à l'approche centralisée classique.

Comme nous l'avons vu, les deux approches partent des spécifications données par l'utilisateur. Nous qualifierons donc ces approches de fonctionnelles, à la différence d'approches structurelles partant d'une décomposition physique du procédé en sous-procédés. Dans le paragraphe suivant, nous nous intéressons à la supervision modulaire locale qui elle est une approche plus structurelle.

2.3 Systèmes composés et supervision localement modulaire

Les « *systèmes composés* » sont des SED composés de plusieurs sous-systèmes évoluant en parallèle. Les sous-systèmes peuvent être synchronisés à certain moment ou pas. Quand ils sont totalement asynchrones on dit qu'on a un « *produit de systèmes* » [QUE00a]. Les systèmes manufacturiers constituent un exemple type de systèmes composés. Nous nous intéressons aux systèmes composés car notre objectif à présent est de proposer une méthode systématique permettant de réaliser la synthèse de la commande de systèmes manufacturiers complexes en raison de leur taille. Nous nous intéressons en particulier aux travaux de M. H. Queiros et J. E. R. Cury [QUE00a] [QUE00b] sur les systèmes composés.

Un système composé est une représentation du procédé global *G* par la composition de sous-procédés G_i , $i \in I = \{1,\ldots n\}$ $\left(G = \|_{i=1}^{n} G_i, \Sigma = \cup_{i=1}^{n} \Sigma_i\right)$. On peut dire que deux

générateurs G_i et G_j sont asynchrones, si $\Sigma_i \cap \Sigma_j = \varnothing$. Tout système composé peut être représenté sous la forme d'un produit de systèmes asynchrones $G_i^{'}$, $i \in I^{'} = \{1,\ldots,n^{'}\}$, avec $G_i^{'} = \|_{j \in Ki} G_j$, où $Ki, i \in I^{'}$, est la limite d'une séquence définie par $Ki^0 = \{k\}(k \in I, k \notin Kl, l \neq i)$ et $Ki^j = Ki^{j-l} \cup \left(k \in I \middle| \Sigma_k \cap \Sigma_k{}^{j-l} \neq \varnothing\right)$.

Soit G un produit de systèmes composé des sous-procédés G_i, $i \in I = \{1,\ldots,n\}$. Soit E_a et E_b des spécifications génériques locales définies respectivement sur $\Sigma_a, \Sigma_b \subseteq \Sigma$. E_x avec $x \in \{a,b\}$ est considérée comme locale car les événements considérés par cette spécification concernent un sous-ensemble du système composé que nous noterons G_X avec $X \in \{A,B\}$. Le procédé local associé à la spécification E_x est défini par $G_X = \|_{j \in I_X} G_i$, avec $I_X = \{k \in I | \Sigma_k \cap \Sigma_x \neq \varnothing\}$. Donc, le procédé local G_X joint seulement des sous-systèmes du système composé original qui sont directement (ou indirectement) restreint par E_x. Pour $X = A$ ou X = B, définissons :

- le procédé restreint $G_R = \|_{X=A,B} G_X$, $\Sigma_R = \cup_{X=A,B} \Sigma_X$, $I_R = \cup_{X=A,B} I_X$;
- les procédés complémentaires $G_X^c = \|_{i \in I - I_X} G_i$, $G_X^{cR} = \|_{i \in I_R - I_X} G_i$, $G_R^c = \|_{i \in I - I_R} G_i$;
- les projections naturelles $P_X : \Sigma_R{}^* \to \Sigma_X{}^*$ et $P_R : \Sigma^* \to \Sigma_R{}^*$.

A ce moment là, les spécifications génériques E_x peuvent être explicitées par rapport au procédé local G_X , le procédé restreint G_R et, le procédé global G , respectivement par : $E_X = E_x \| L_m(G_X)$, $K_{XR} = E_x \| L_m(G_R)$ et $K_X = E_x \| L_m(G)$.

Définition 22 [QUE00a] : Modularité locale

Les langages E_A , $E_B \subseteq \Sigma_R^*$ sont dits être localement modulaires si $\overline{P_A^{-1}(E_A) \cap P_B^{-1}(E_B)} = \overline{P_A^{-1}(E_A)} \cap \overline{P_B^{-1}(E_B)}$ i.e., si $\overline{E_A \| E_B} = \overline{E_A} \| \overline{E_B}$

Théorème 5 [QUE00a] : Supposons un système composé G avec les sous-procédés G_i, $i = 1,\ldots,n$ et les spécifications E_a et E_b. Soit $E_A, E_B, K_{AR}, K_{BR}, G_A, G_B$, et G_R définis comme précédemment. Si $SupC(E_A, G_A)$ et $SupC(E_B, G_B)$ sont modulaires localement, alors $SupC(K_{AR} \cap K_{BR}, G_R) = SupC(E_A, G_A) \| SupC(E_B, G_B)$.

Le Théorème 5 étend l'approche modulaire de Ramadge et Wonham [WON88] au_ contexte d'une modularité locale. Ce théorème prouve que si les superviseurs locaux construits à partir des spécifications locales sont non-conflictuels, alors leur contrôle conjoint sur le procédé restreint est équivalent à celui qu'on obtiendrait à l'aide d'un superviseur monolithique construit à partir des spécifications restreintes. D'autre part, la modularité locale assure que l'action conjointe de superviseurs locaux résultants est optimale, c'est-à-dire, qu'elle ne présente pas de perte de performances par rapport au contrôle global centralisé le permissif réalisé sur le procédé restreint. D'un point de vue pratique, le concept de modularité locale signifie que l'on peut restreindre l'approche modulaire à un procédé restreint du procédé G, constitué par le produit synchrone de tous les systèmes composants partageant des événements avec les spécifications considérées. L'intérêt de cette restriction est que de faite elle réduit la complexité en réduisant la taille du système considéré pour la construction des superviseurs locaux.

En fait, la supervision modulaire locale de Queiroz et Cury tire profit de la supervision modulaire et de la supervision décentralisée présentées à la section 2.2. En effet, par rapport à l'approche décentralisée, dans le cadre des systèmes composés, elle propose une méthode pour définir les sous-systèmes Gi. Ensuite, moyennant une redéfinition des spécifications elle s'appuie sur l'approche modulaire pour proposer un cadre permettant de garantir un control modulaire optimal. Notons également ici que les auteurs proposent une méthode de transcription des contrôleurs en ladder pour l'implémentation sur automates programmables industriels [QUE 02]. Dans la suite de cette étude, nous allons appliquer cette approche à la conception RdP de contrôleurs pour la commande des systèmes manufacturiers.

2.4 Adaptation de la supervision localement modulaire à la commande des systèmes manufacturiers

Afin d'adapter la supervision modulaire localement au cadre de la commande des systèmes manufacturiers, nous proposons une approche en cinq étapes comme illustrée par la Figure 43. Nous supposons que le procédé est composé de machines évoluant en parallèle mais pouvant être synchronisé par des contraintes de fonctionnement structurelles ou fonctionnelles. Par exemple, la présence d'un stock intermédiaire de capacité limité sera considérée comme une contrainte structurelle. L'absence de blocage dans la réalisation d'une tâche par une machine sera considérée comme une contrainte fonctionnelle. La méthode proposée consiste donc à commencer par définir le modèle de procédé restreint Gx relatif à chaque spécification Ex. Une fois ce modèle obtenu, Gx et Ex étant spécifié sous forme RdP, nous pouvons calculer le superviseur local Sx en utilisant la méthode présentée au chapitre II.

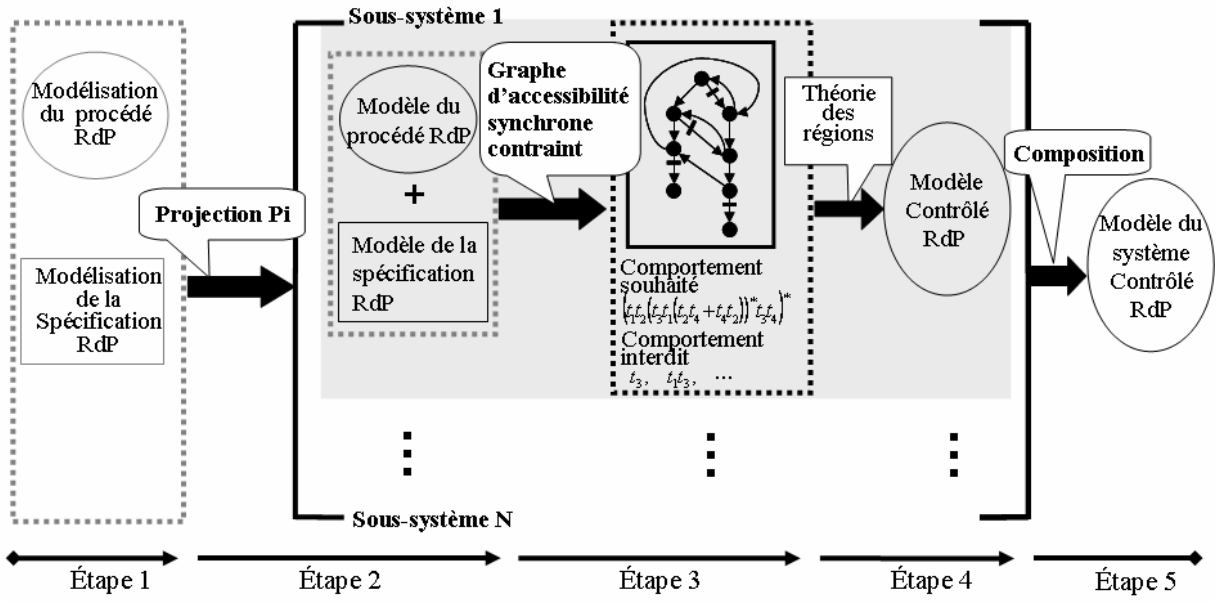


Figure 43. Approche en cinq étapes

Le rôle de chaque étape peut être résumé comme suit :

Etape 1 : Construction du modèle RdP du procédé et des modèles RdP des spécifications.

Initialement nous ne considérons que les machines constituant le système. Le comportement de chaque machine est représenté sous la forme d'un RdP $G_i = \left(N^i, M_0^i\right)$ connexe indépendant

des autres. Chaque modèle contient des transitions contrôlables et des transitions incontrôlables spécifiées par l'utilisateur. De ce fait, par construction, nous obtenons un système produit.

Etape 2 : Construction des modèles de procédé restreint et du modèle RdP par spécification.

On détermine l'ensemble G_X des machines qui sont associées à une spécification donnée. L'ensemble des transitions de cet ensemble de machines constitue l'ensemble de transitions pouvant servir à construire le modèle RdP de la spécification. Afin de pouvoir assurer le contrôle des machines du procédé restreint considéré, le modèle RdP de la spécification doit contenir au moins une transition synchrone vis-à-vis du modèle RdP de chaque machine.

Remarque : On constate que par construction, on retrouve la condition de définition de G_X telle que définie au §2.3 (le procédé local associé à la spécification E_x est défini par $G_X = \left\| _{j \in I_X} G_i \right.$, avec $I_X = \left\{ k \in I \middle| \Sigma_k \cap \Sigma_x \neq \varnothing \right\}$ avec $\sum_k$ l'alphabet composé par les événements associés aux transitions du RdP du système 'k'.

Etape 3 : Pour chaque procédé restreint G_X , construction de séquences interdites de transitions d'état et de séquences de comportement souhaité qui donnent le comportement en boucle fermé du procédé basé sur la synthèse du modèle du procédé G_X et du modèle de la spécification de l'utilisateur E_x .

Etape 4 : Pour chaque procédé restreint G_X , utilisation de la théorie des régions afin de déterminer les places de contrôle à ajouter au modèle RdP initial de G_X .

Etape 5 : Obtention du modèle RdP global du système contrôlé en boucle fermé : connexion des places de contrôle au modèle RdP du procédé en boucle ouverte (ensemble des RdP des machines considérées chacune de manière isolé) en se référant au Théorème 5.

Pour illustrer notre méthode, nous utilisons l'exemple d'une « ligne de transfert industriel » composée de six machines Mi (i = 1,…,6) reliées par les stocks S_a, S_b, S_c et S_d, travaillant comme le montre la Figure 44 [QUE00b]. Afin de simplifier le problème, les stocks sont supposés de capacité unitaire. Nous devons synchroniser la commande des machines pour tenir compte des capacités finies des stocks. De ce fait, à chaque stock est associé une spécification. Le flux des pièces traverse le système des machines M1 et M3 vers la machine M6. Les spécifications considérées sont les suivantes : pour un stock donné, la

(les) machine(s) amont Mi ne peut produire que si le stock n'est pas plein ; la machine aval Mi+1 ne peut produire que si le stock n'est pas vide.

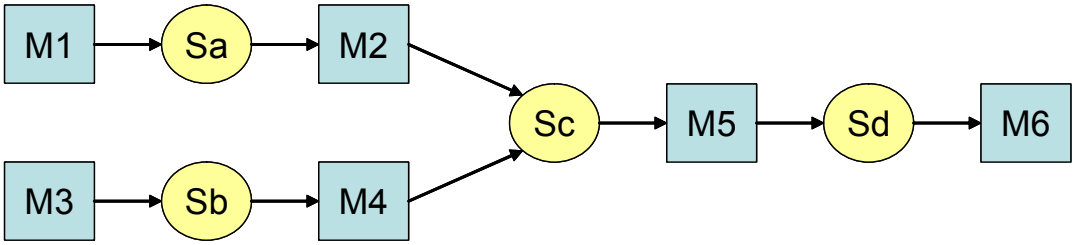


Figure 44. Ligne de transfert industriel

Étape 1. Nous supposons les 6 machines identiques. Par souci pédagogique, nous retenons pour chaque machine Mi le modèle RdP générique illustré par la Figure 45. La place P2i-1 modélise la machine à l'arrêt et place P2i, la machine en marche. La transition t2i-1 modélise le démarrage de la machine ; c'est une transition contrôlable. La transition t2i modélise l'arrêt de la machine ; c'est une transition incontrôlable. En résumé, les transitions contrôlables sont $T_C = \{t_1, t_3, t_5, t_7, t_9, t_{11}\}$ et les transitions incontrôlables sont $T_{UC} = \{t_2, t_4, t_6, t_8, t_{10}, t_{12}\}$.

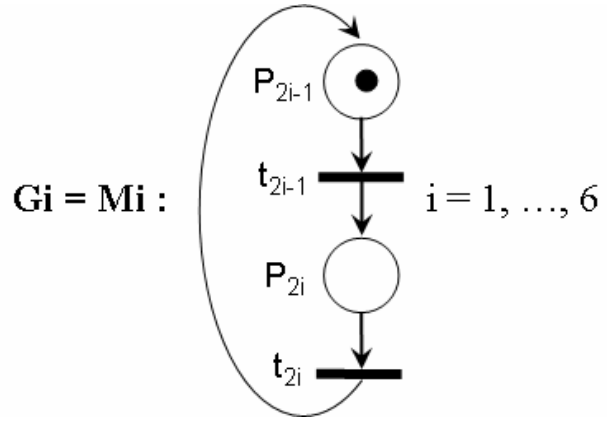


Figure 45. Modèle du procédé en boucle ouverte des six machines : $G_1, G_2, \ldots, G_6$

Étape 2. Comme le montre la Figure 46, le système considéré peut être décomposé en quatre sous-systèmes en fonction des machines en interaction avec chacun des stocks. Les trois premiers sous-systèmes sont évidents à obtenir. Ainsi, par rapport au stock Sa, nous pouvons composer les modèles des machines M1 et M2 pour obtenir le modèle du sous-système A. Par

abus de langage, nous le noterons $G_A = G_1 \| G_2$ comme dans le cas des opérations sur les automates. De même nous obtenons $G_B = G_3 \| G_4$ associé au stock Sb et $G_D = G_5 \| G_6$ associé au stock Sd. Le modèle associé au stock Sc est un peu plus complexe. En effet, deux possibilités sont offertes au concepteur :

Cas 1 : $G_C = G_2 \| G_4 \| G_5$: On reste dans le cadre du système produit de départ et on considère l'association de machines indépendantes ;

Cas 2 : $G_C = G_A \| G_B \| G_D$: On considère un nouveau système produit constitué par les trois premiers sous-systèmes créés. Ces trois systèmes sont en interaction avec le stock Sc donc ils peuvent être associés.

Nous retenons comme modèle pour la première machine le cas 1, i.e. $G_C = G_2 \| G_4 \| G_5$. En effet, la première raison est que ce choix est plus conforme à l'esprit de la modularité locale dans la mesure où les modèles des machines étant plus petits, la complexité est moindre. La deuxième raison est que le cas 2 peut être assimilé à une hiérarchisation des spécifications et donc des superviseurs. Ce cas ne correspond ni à une approche modulaire, ni à une approche décentralisée. Il s'apparenterait à une approche hiérarchique [WON96a], [WON96b]. Nous montrerons en annexe 2 que le cas 2 aboutit à une solution fausse.

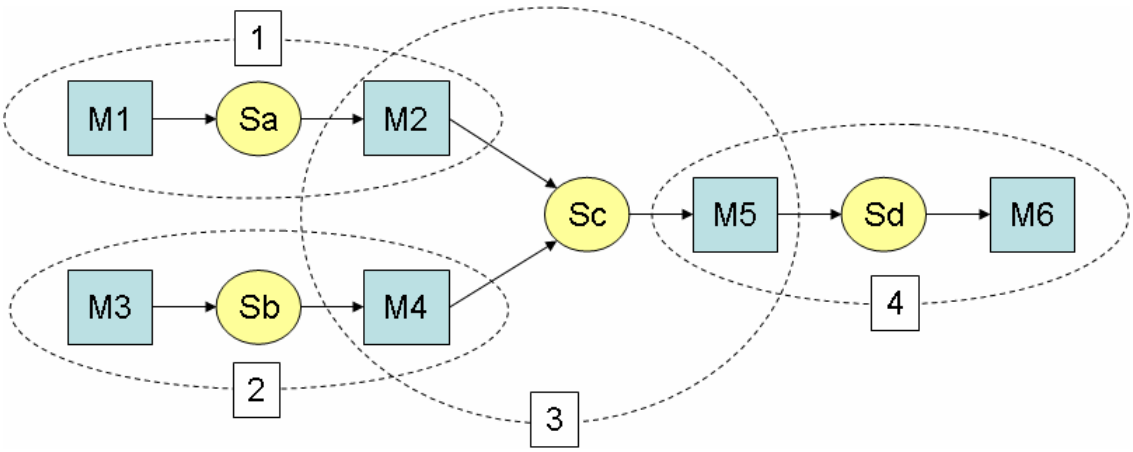
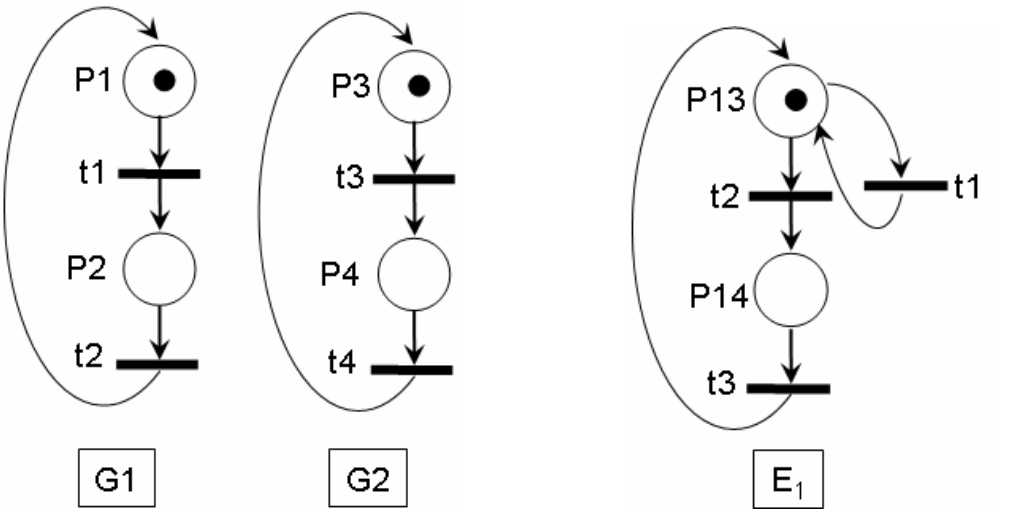


Figure 46. Regroupement des machines en sous-systèmes

Une fois les regroupements en sous-procédés effectués, nous connaissons alors les modèles RdP des machines inclus dans le regroupement. Cela nous permet de déterminer

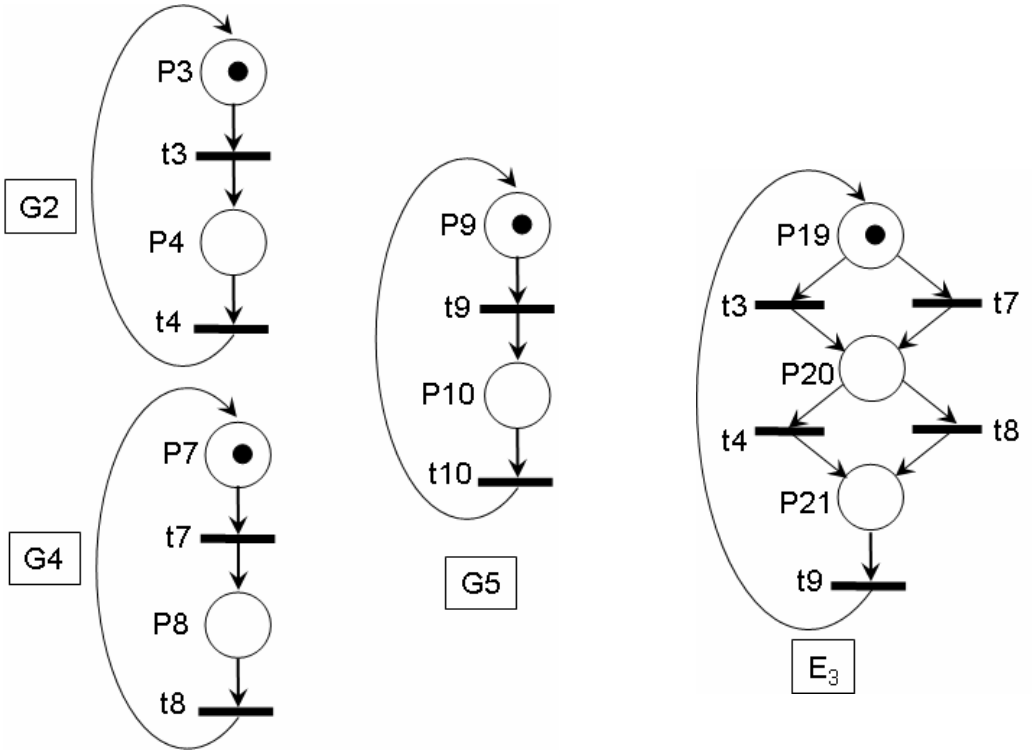
l'ensemble des transitions apparaissant dans ces modèles afin de pouvoir les utiliser pour construire le modèle RdP de la spécification. Par exemple, la Figure 47.A nous montre que l'ensemble des transitions de G_A est $T_{G_A} = \{t_1, t_2, t_3, t_4\}$. Nous pouvons donc en déduire le modèle de spécification donné par la Figure 47.B. Des démarches analogues permettraient de construire les modèles RdP des spécifications relatives aux stocks S_b et S_d.



A. Modèle du procédé B. Modèle de la spécification

Figure 47. Modèle du procédé et modèle de la spécification de sous-système 1

Pour construire la spécification du stock Sc, il faut tenir compte qu'il a deux machines en amont (M2 et M4) et une machine en aval (M5). Le stock peut être dans 3 états : vide, réservé pour une pièce en cours de fabrication et plein. Les transitions de $T_{G_C} = \{t_3, t_4, t_7, t_8, t_9, t_{10}\}$, on en déduit le modèle de spécification de la Figure 48.B.



A. Modèle du procédé B. Modèle de la spécification

Figure 48. Modèle du procédé et modèle de la spécification du sous-système 4

Étape 3. Pour chaque modèle du procédé et chaque spécification associée, nous établissons les équations de la théorie des régions à l'aide de l'algorithme I présenté au § II.4.2. Ainsi les états du modèle du procédé et du modèle de spécification sont fusionnés par la construction d'un Graphe d'Accessibilité Synchrone Contraint (GASC). Afin de faciliter la détermination des équations de la théorie des régions, nous avons réalisé l'outil NetMDI qui est une application développé en langage C (Microsoft Visual C++ 6.0). Nous reviendrons sur la conception de cet outil dans le chapitre IV. Après extraction du comportement souhaité (équations d'accessibilité et équations de cycle) et de l'ensemble de séquences interdites calculés à l'aide l'algorithme 1, l'outil fait appel à CPLEX pour résoudre le système de programmation linéaire que représente l'ensemble de ces équations.

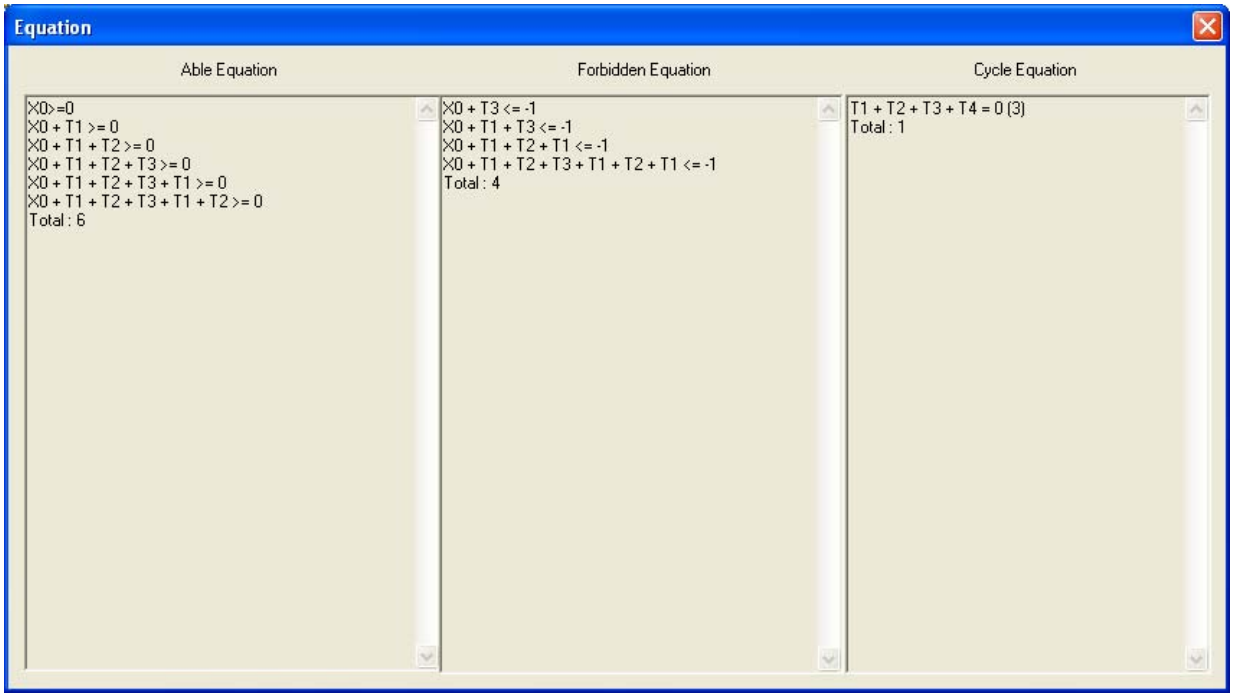


Figure 49. Equations du comportement souhaité et équations interdites pour le procédé G_A

- Dans le cas du procédé G_A, six équations expriment le comportement souhaité sur le plan accessibilité, quatre équations codent le comportement interdit et nous avons en plus une équation de cycle. La Figure 49 donne ces équations obtenues à l'aide de l'outil NetMDI. Notons que dans cette figure X_0 correspond au marquage initial de la place de contrôle associée et T_i est une notation qui correspond à l'arc connectant la transition 't_i' à la place de contrôle P_c (équivalent à $C(P_C, T_i)$). Les transitions 't_2' et 't_4' étant définies comme incontrôlables, l'équation (2.12) nous permet de poser en plus : $C(p_c, t2) \geq 0$ et $C(p_c, t4) \geq 0$. Les cas des procédés G_B et G_D sont similaires.

Dans le cas, du procédé G_C, nous avons huit équations correspondant aux conditions d'accessibilité, deux équations de cycle et neuf équations correspondant au comportement interdit (Figure 50.B). Les transitions 't_4', 't_8'et 't_{10}' étant définies comme incontrôlables, l'équation (3.12) nous permet de poser : $C(p_c, t4) \geq 0$, $C(p_c, t8) \geq 0$ et $C(p_c, t10) \geq 0$.

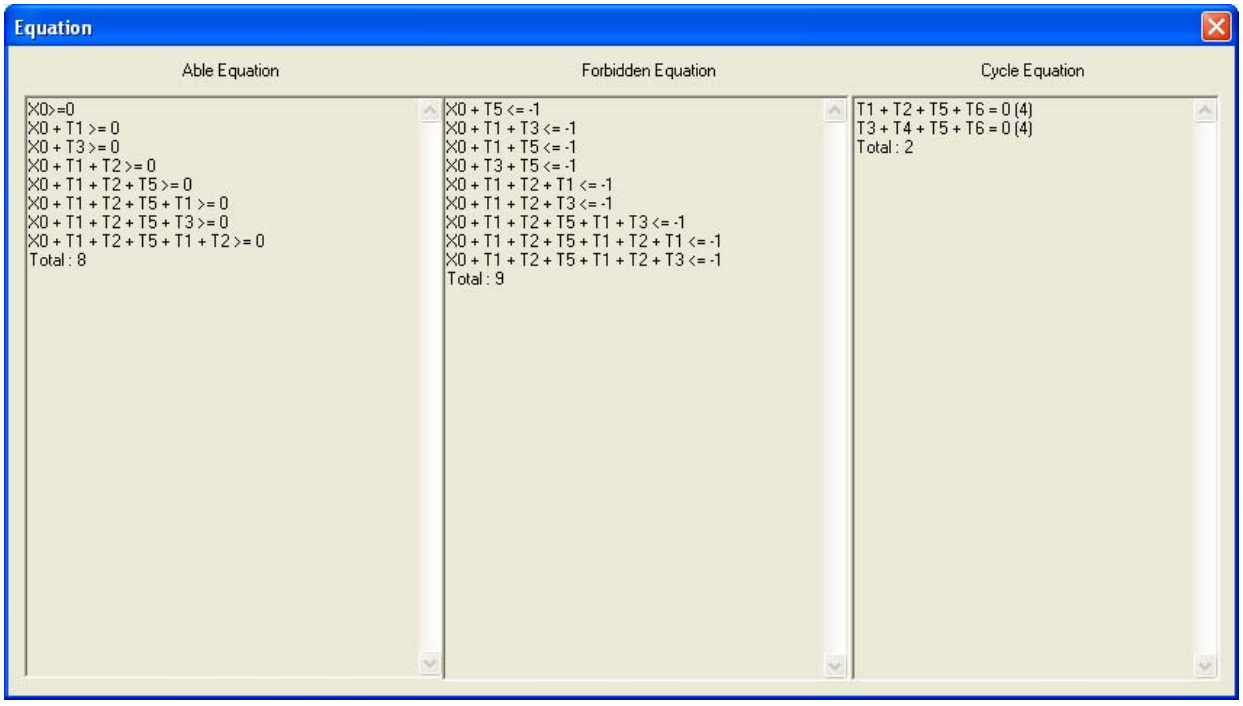


Figure 50. Equations du comportement souhaité et équations interdites pour le procédé G_C

Étape 4. Nous effectuons itérativement la résolution des différents systèmes d'équations à l'aide d'un solveur comme CPLEX. Nous développerons le principe de cette résolution assistée dans le quatrième chapitre. Le cas des procédés G_A, G_B et G_D a déjà été traité au paragraphe II.5.2. La résolution aboutit à des contrôleurs du type de celui donné par la Figure 51.

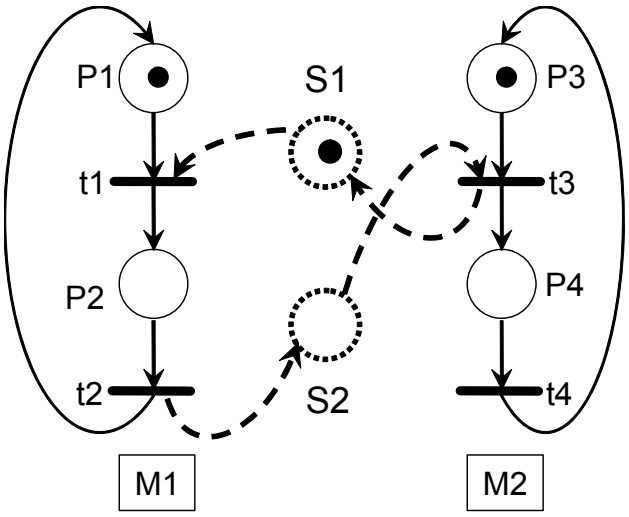


Figure 51. Commande en boucle fermée du sous-système G_A

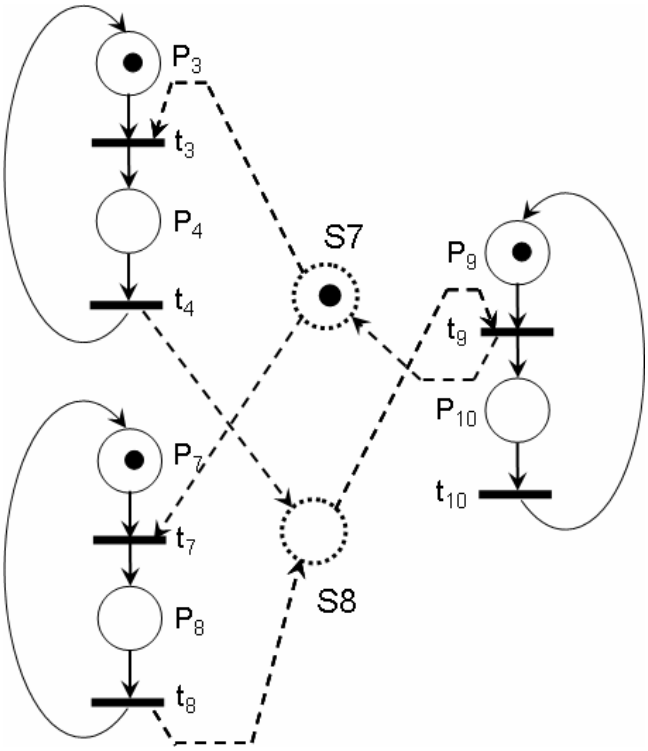


Figure 52. Commande en boucle fermée du sous-système G_C

Dans le cas du procédé G_C, la résolution du système d'équations associé aboutit à deux places de contrôles définies respectivement par les solutions $C(S_8,.)=(0 \quad 1 \quad 0 \quad 1 \quad -1 \quad 0)$ et $M_0(S_8)=0$ d'une part et, $C(S_7,.)=(-1 \quad 0 \quad -1 \quad 0 \quad 0 \quad 0)$ et $M_0(S_7)=0$ d'autre part. On obtient alors le contrôleur de la Figure 52.

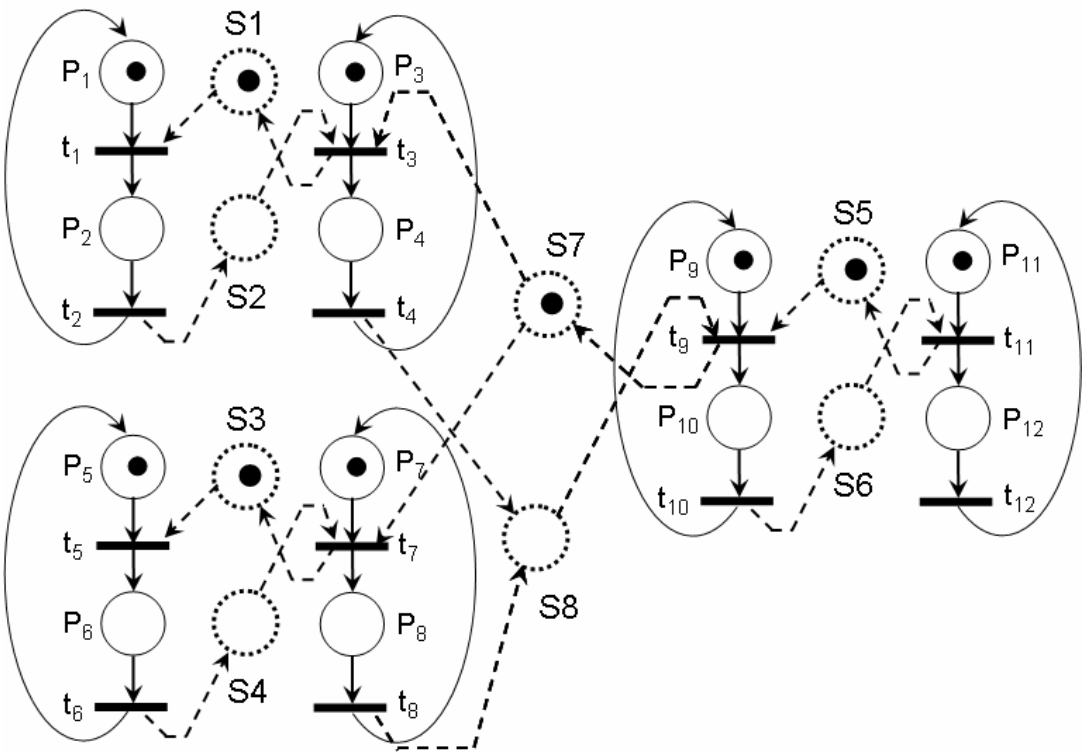


Figure 53. Contrôleur global de la ligne de transfert industriel de la Figure 44

Étape 5. Pour obtenir le modèle du contrôleur du système global, il suffit de fusionner les différents modèles partiels obtenus à l'étape 4 (Figure 53). Le modèle en boucle fermé de la ligne de transfert se traduit dont par l'ajout de 8 places de contrôles permettant de connecter les modèles initiaux des machines.

3. Synthèse de contrôleurs pour l'allocation des machines aux gammes opératoires

3.1 Introduction

Comme nous l'avons rappelé dans le premier chapitre, dans la méthodologie de commande des systèmes manufacturiers développée au LAGIS, les gammes opératoires permettent de découpler le contrôle par rapport aux produits, du contrôle à proprement parler des ressources de productions. La gamme opératoire d'un produit est élaborée en faisant abstraction des stratégies d'exploitation du système de production. De ce fait, en exploitation, pour que la commande respecte l'ordonnancement prévisionnel, nous avons besoin

d'allocateurs de machines permettant de définir à chaque instant quel produit peut utiliser une ressource donnée. Notre objectif dans cette partie est de contribuer à la construction systématique en RdP, d'allocateurs de machines dans le cadre d'une production cyclique. Nous nous plaçons dans un contexte où la stratégie de pilotage du système de production peut changer en fonction des modes de marche. Nous allons essayer de répondre à deux questions liées à la stratégie de pilotage. Etant donné une séquence de produits définie sur un cycle de production, peut-on trouver des contrôleurs permettant un contrôle déterministe ? Etant donné un ordonnancement cyclique, peut-on trouver des contrôleurs permettant un contrôle déterministe ?

3.2 Problématique

Pour poser la problématique abordée dans cette partie, considérons l'exemple du système manufacturier donné par la Figure 54. Supposons que sur ce système manufacturier on fabrique deux types de pièces OS1 et OS2. Le système étant un « flowshop », les deux familles de pièces passent à tour de rôle d'abord sur la machine 1 puis sur la machine 2. Supposons d'autre part que la stratégie d'exploitation consiste à fabriquer 50% de pièces de chaque type sur l'horizon de production. Un moyen d'y parvenir consiste à sortir en alternance une pièce de chaque type, soit la séquence OS1-OS2-OS1-OS2 … . Cette stratégie de pilotage montre l'émergence d'un cycle élémentaire de production caractérisé par la séquence de production OS1-OS2. L'horizon de production peut être découpé en ces cycles élémentaires ce qui assure l'obtention de la production souhaitée dans les délais définis.

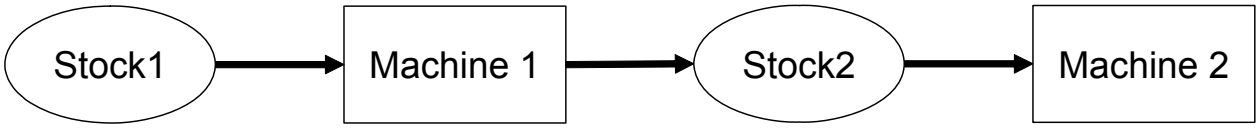


Figure 54. Exemple de système manufacturier

Dans cette problématique, nous ne nous intéressons pas contrairement aux sections précédentes à la gestion des contraintes imposées par les stocks ni au type d'opération exécutée par les machines sur chaque type de pièce. Aussi, nous supposerons que les stocks sont de capacité infinie. Notre objectif est de modéliser les contrôleurs permettant de

coordonner la production de chaque type de pièce. Ces contrôleurs seront modélisés par des RdP ordinaires. Les modèles initiaux de ces contrôleurs seront des gammes opératoires qualitatives. Les opérations de transfert seront négligées par soucis de simplicité et dans la mesure ou l'allocation des ressources de transport s'effectue au niveau du Graphe de Commande du Système de Transport (voir chapitre 1). Moyennant ces hypothèses, et compte tenu d'un encours d'une pièce pour chaque type par cycle élémentaire, nous pouvons modéliser la commande de ce système par le réseau de petri de la Figure 55. La gamme opératoire OS1 (respectivement OS2) correspond aux places P1 à P4 (respectivement P5 à P8). Les places d'une gamme modélisent la localisation d'une pièce dans le système de production. Par exemple, la place P1 modélise une pièce OS1 dans le stock 1. La place P2 modélise une pièce OS1 en cours de production sur la machine 1. Pour matérialiser que l'accès aux machines est conflictuel entre les pièces des deux gammes, nous avons protégé les sections critiques à l'aide de mutex (place modélisant une exclusion mutuelle). Ainsi la place nommée M1, correspond à la mutex modélisant que la machine 1 est libre. La transition 't1' modélise donc le début de l'opération de transformation d'une pièce OS1 par la machine M1 et la transition 't2' la fin de cette opération.

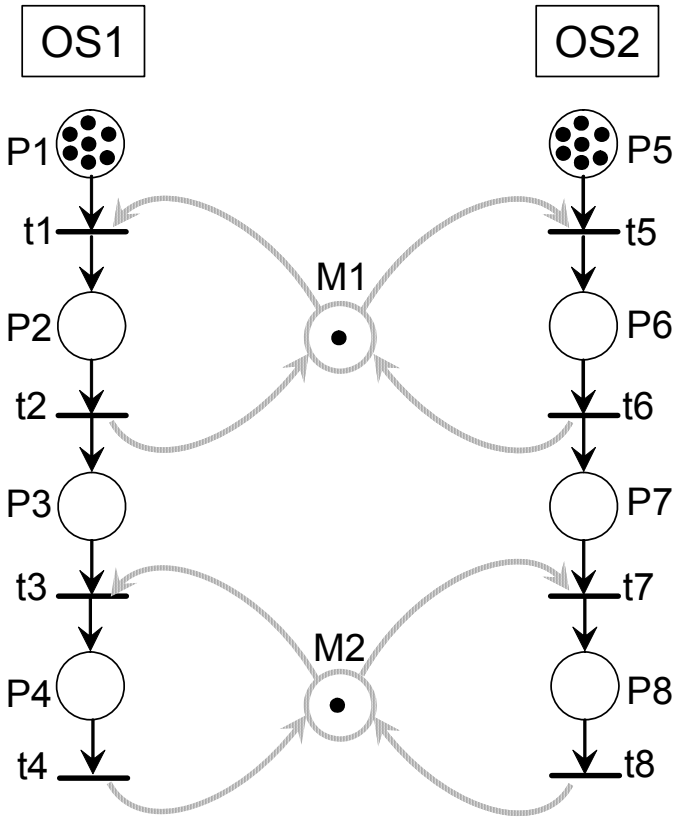


Figure 55. Gammes opératoires des pièces OS1 et OS2 dans le cadre du problème initial

La présence des mutex ‘M1’ et ‘M2’ entre les deux gammes ne permet pas d’avoir le contrôle déterministe garantissant d’avoir en sortie les ratios de production souhaités. Les stocks étant infinis nous avons placé initialement dans P1 et P5 un nombre infinis de jetons modélisant la présence d’un nombre infini de pièces de chaque type dans le stock. Le franchissement aléatoire des transitions ‘t1’ et ‘t3’ par exemple peut très bien conduire à ne produire que des pièces OS1 en sortie. Cela montre la nécessité de modifier notre modèle pour se placer dans le contexte cyclique évoqué au début de cette partie. Il montre également la nécessite de définir une séquence d’allocation de machines conforme à la stratégie de production qui a été définie. Dans un cycle, nous voulons d’abord attribuer les ressources aux pièces de type OS1, puis à celle de type OS2. Ce problème bien connu se résout par l’introduction de séquences alternées entre les deux gammes cycliques. La place $M_i - OS_j$ modélise que la machine ‘i’ ($i \in \{1,2\}$) est allouée à la gamme OS_j ($j \in \{1,2\}$).

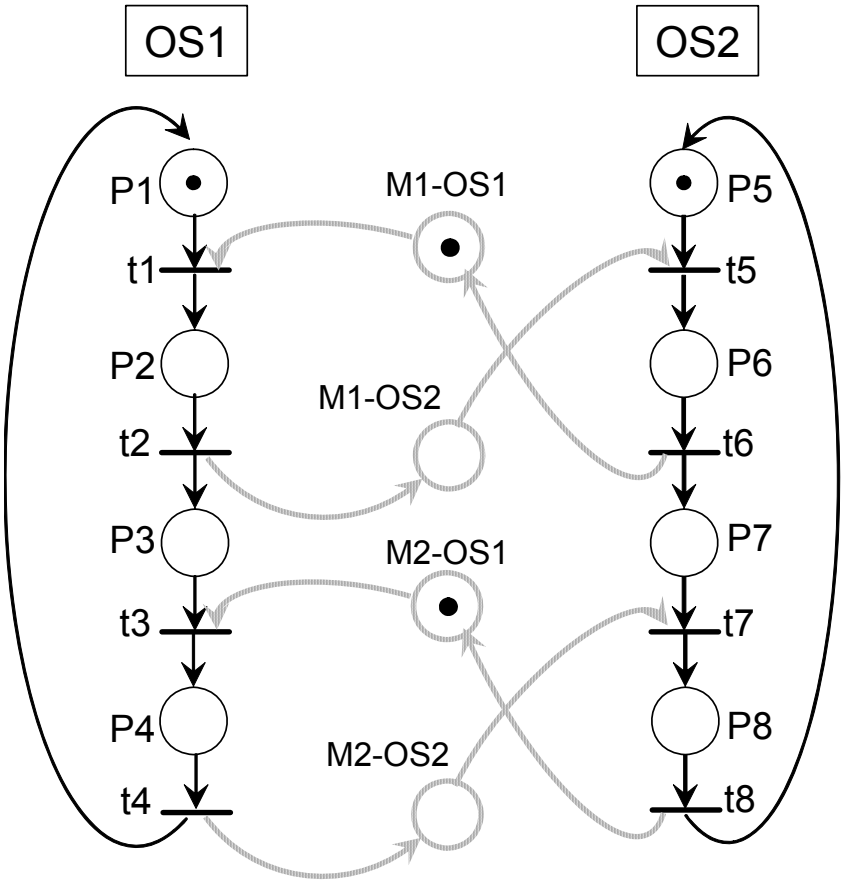


Figure 56. Allocateurs modélisés par des séquences alternées.

Supposons à présent que l'on change de stratégie de pilotage et que l'on veuille obtenir en fin de production 66,66% de pièces OS1 et 33,33% de pièces OS2. Peut-on générer de manière systématique des nouveaux allocateurs de machines afin que les contrôleurs des pièces soient déterministes ?

Les gammes opératoires modélisant les opérations à réaliser pour obtenir un produit fini, elles constituent des invariants que nous devons retrouver dans les contrôleurs mettant en œuvre une stratégie particulière de pilotage. Notre objectif est donc de conserver la structure des gammes mais de changer la stratégie d'allocation (partie en grisée dans les Figure 55 et Figure 56). Dans un fonctionnement cyclique, nous pouvons obtenir les nouveaux ratios à l'aide d'un en cours de deux pièces OS1 et d'une pièce OS2. Supposons que dans chaque cycle nous définissons l'ordre de passage suivant sur la machine 1, OS1-OS1-OS2 et sur la machine 2, OS1-OS2-OS1. Comment déterminer si nous pouvons obtenir un contrôleur mettant en œuvre la stratégie de pilotage définie par l'utilisateur ?

3.3 Principe de la résolution : approche logique

Pour répondre à la problématique énoncée dans le paragraphe précédent, l'idée est d'utiliser notre approche de synthèse de contrôleurs basée sur la théorie des régions. Le principe de cette synthèse est le suivant. Les gammes opératoires des produits modélisent le comportement du procédé en boucle ouverte. Notre problème est donc de construire des spécifications RdP qui modélisent la stratégie d'allocation de chaque contrôleur de machine. Pour cela il est nécessaire de caractériser les événements associés aux transitions du modèle du procédé qui sont utilisables pour spécifier le fonctionnement d'un allocateur de ressource.

Définition 23. Opération de transformation

A l'aide des RdP ordinaires, les opérations de transformations sont modélisées par une séquence composée d'une transition de début d'opération, une place modélisant l'opération en cours et une transition de fin d'opération.

Comme dans la théorie du Supervisory Control, nous associons aux transitions de notre RdP des événements contrôlables ou pas. Ainsi à une transition de début d'opération est associée un événement contrôlable tandis qu'à une transition de fin d'opération est associée un événement incontrôlable.

L'idée de base est donc d'utiliser le motif définissant une opération de transformation dans une gamme opératoire pour définir la stratégie d'allocation de chaque allocateur. On peut ainsi établir des séquences d'allocation et de libération d'une machine selon un modèle de fonctionnement cyclique. Considérons de nouveau la production de deux pièces de type OS1 et une pièce de type OS2 sur le système de production représenté par la Figure 54. Dans le cas d'une approche logique, nous nous intéressons uniquement à l'ordre de réalisation des opérations de chaque pièce par la ressource considérée. Ainsi nous supposons l'utilisateur désire réaliser la séquence OS1-OS1-OS2 sur la machine 1 et la séquence OS1-OS2-OS1 sur la machine 2. Pour chaque machine, nous écrivons cette spécification informelle sous forme RdP. Par exemple dans le cas de la machine 1, la transformation d'une pièce de type OS1 est caractérisé dans le modèle du procédé par l'allocation de la machine lors du franchissement

de la transition ‘t1’ (début transformation d’OS1) et par sa désallocation lors du franchissement de ‘t2’ (fin de la transformation d’OS1). L’alternance de ces deux transitions est reprise deux fois lors de la spécification de l’allocateur de la machine 1 afin de modéliser le passage consécutif de deux pièces de type OS1. On termine le modèle en utilisant les transitions ‘t5’ et ‘t6’ qui caractérisent la transformation d’une pièce OS2 par la machine 1 dans le modèle du procédé. On procède de même avec la deuxième machine. On obtient alors comme modèle de spécification les deux RdPs représentés par la Figure 57.

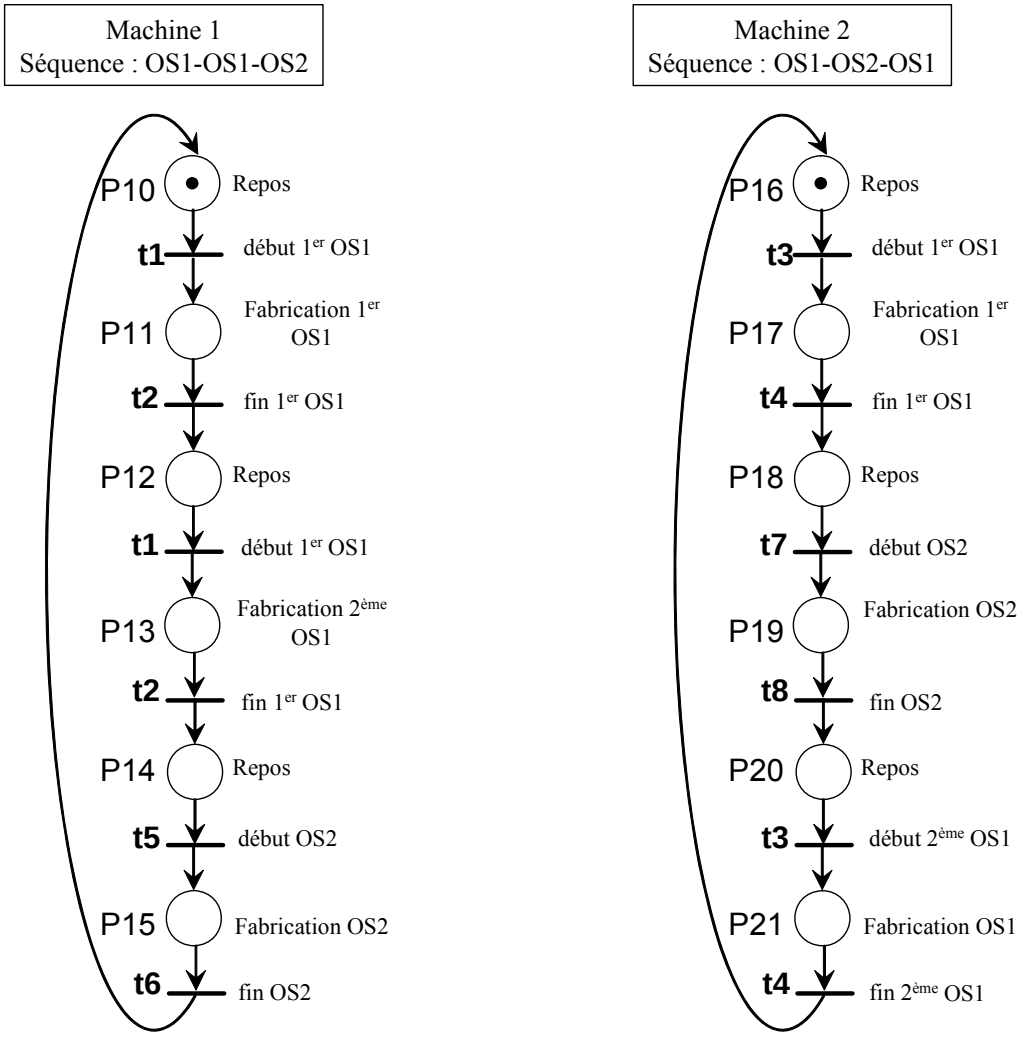


Figure 57. Exemple de spécification d’allocateurs de ressource dans le cas de l’approche logique

Notons qu’ici également les transitions du modèle de spécification sont des transitions synchrones avec celles du modèle du procédé.

Une fois le modèle de spécification obtenu, on en fait la synthèse avec le modèle du procédé pour déterminer si on obtient un modèle déterministe. Dans le cas de notre exemple, nous obtenons le modèle RdP de la Figure 58. Nous pouvons constater que le résultat n'est pas un modèle intuitif ce qui prouve l'intérêt d'avoir une approche systématique.

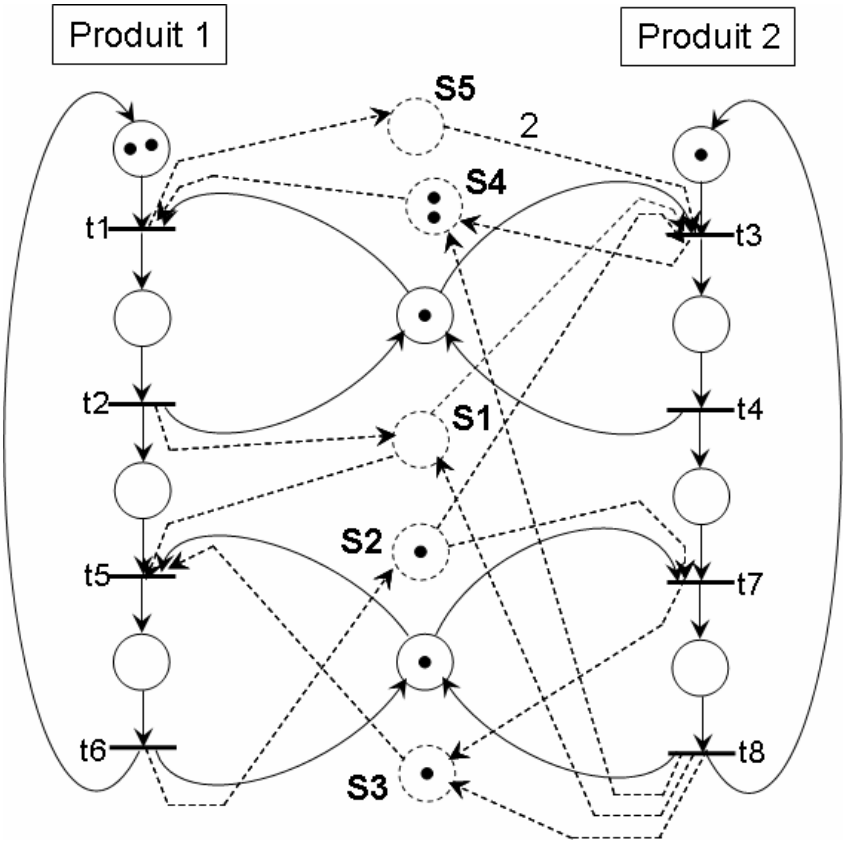


Figure 58. Contrôleur déterministe correspondant aux spécifications de l'utilisateur

Ce n'est pas parce que l'on spécifie l'allocation désirée de la part des ressources considérée que l'on peut obtenir un contrôleur déterministe la mettant en œuvre. Nous allons l'illustrer dans la section suivante relative à la recherche de la synthèse d'un contrôleur déterministe mettant en œuvre un ordonnancement cyclique.

3.4 Allocateur de ressource mettant en œuvre un ordonnancement cyclique

L'ordonnancement cyclique est connu pour être une technique adaptée à l'ordonnancement de productions correspondant à des demandes importantes [TOG05]. En effet, cette technique constitue une bonne approche pour éviter l'explosion combinatoire due à la complexité de l'ordonnancement de toutes les opérations nécessaires pour réaliser une production donnée dans un délai fixé. Les approches cycliques permettent de ramener le problème à la détermination et à l'optimisation d'un cycle répétitif. Des travaux récents [CAM87], [KOR88], montrent que ces techniques peuvent également être adaptées à la production de petites et moyennes séries de produits sous réserve d'étudier et d'optimiser également les modes transitoires. Parmi ces approches, nous pouvons citer l'ordonnancement par rapport à la machine critique, les techniques d'ordonnancement K-cyclique et l'ordonnancement 1-cyclique. Toutes ces approches ont pour objectif d'optimiser le temps de production tout en minimisant l'encours. Dans cette étude, nous nous intéressons en particulier à l'approche 1-cyclique [CAM97], [KOR98]. Elle est caractérisée par le fait que chaque machine réalise la même opération à la même date de chaque cycle élémentaire de production. Afin de simplifier l'étude, nous nous focalisons sur l'ordonnancement des opérations de transformation en négligeant les opérations de transfert.

Afin d'illustrer notre approche, nous réutilisant l'exemple bien connu de Valentin [VAL94]. Le système considéré est constitué de 3 machines ($M_i / i \in [1..3]$) mutuellement accessibles et fabriquant deux types de pièces : OS_1 et OS_2. Chaque type de pièce est caractérisé par sa gamme opératoire. Dans cet exemple, chaque opération d'une gamme est définie par la machine qui réalise l'opération et par sa durée (donnée ici en unité de temps générique noté u.t.). Nous noterons OP_{ij} la $j^{éme}$ opération de la gamme de OS_i. Les gammes opératoires considérées sont donc :

- OS_1 : OP_{11} (M_3, 2 u.t.), OP_{12} (M_1, 3 u.t.), OP_{13} (M_2, 2 u.t.)
- OS_2 : OP_{21} (M_1, 1 u.t.), OP_{22} (M_3, 2 u.t.)

Supposons que l'encours optimal compte tenu des spécifications précédentes est de 5, correspondant à la production de 3 pièces de type OS_1 et de 2 pièces de types OS_2 par cycle.

Même si les pièces dans chaque type sont identiques, afin de déterminer le motif optimal du cycle, il est nécessaire de les identifier pour les distinguer. Ainsi, nous noterons w1 à w3 les pièces relatives à la gamme OS_1 et w4 et w5 celles relative à la gamme OS_2.

L'approche 1-cyclique développées au LAGIS donne comme résultat le diagramme de Gantt double illustré par la Figure 59. La partie haute de ce diagramme correspond à l'ordonnancement des opérations des différentes pièces. La partie basse correspond à l'ordonnancement dual des opérations exécutées par chaque machine. Dans la suite de cette étude nous nous intéresserons plus particulièrement à la construction de contrôleurs permettant de mettre en œuvre cette partie basse. La durée optimale du cycle est de 11 u.t. Cette valeur est obtenue grâce au chevauchement de cycle. En effet, nous pouvons constater que l'opération OP_{22} sur la pièce w4 commence sur le cycle n et se termine sur le cycle n+1. La machine critique dans cet ordonnancement est la machine M1 qui travaille à 100% de la durée du cycle. Cette caractéristique est très intéressante pour détecter des défaillances progressives sans qu'il soit nécessaire d'instrumenter particulièrement le procédé ([TOG05]).

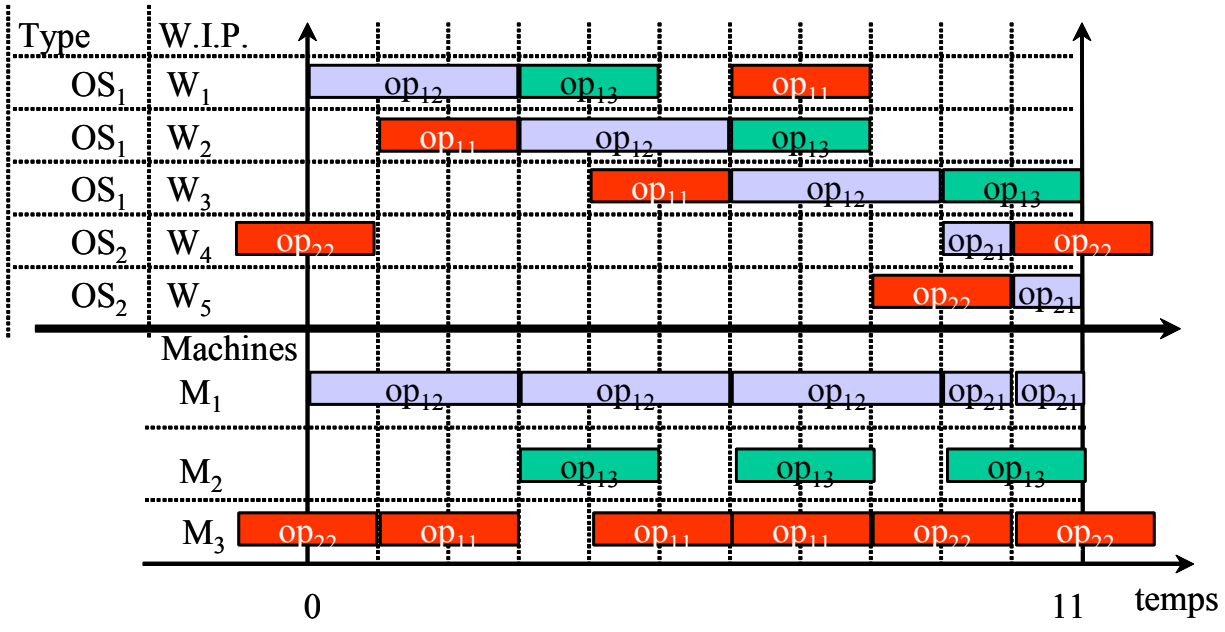


Figure 59. Ordonnancement 1-cyclique de l'exemple de Valentin

Etant donné un ordonnancement cyclique, pour déterminer s'il existe un contrôleur déterministe le mettant en œuvre, il suffit de transcrire en RdP l'ordonnancement des opérations par rapport aux machines. Pour cela, il est nécessaire de se référer au modèle de

commande en boucle ouverte afin de déterminer les transitions modélisant l'allocation et la désallocation de chacune des ressources. Dans le cas de l'exemple de Valentin, la commande en boucle ouverte est modélisée par le RdP de la Figure 60.

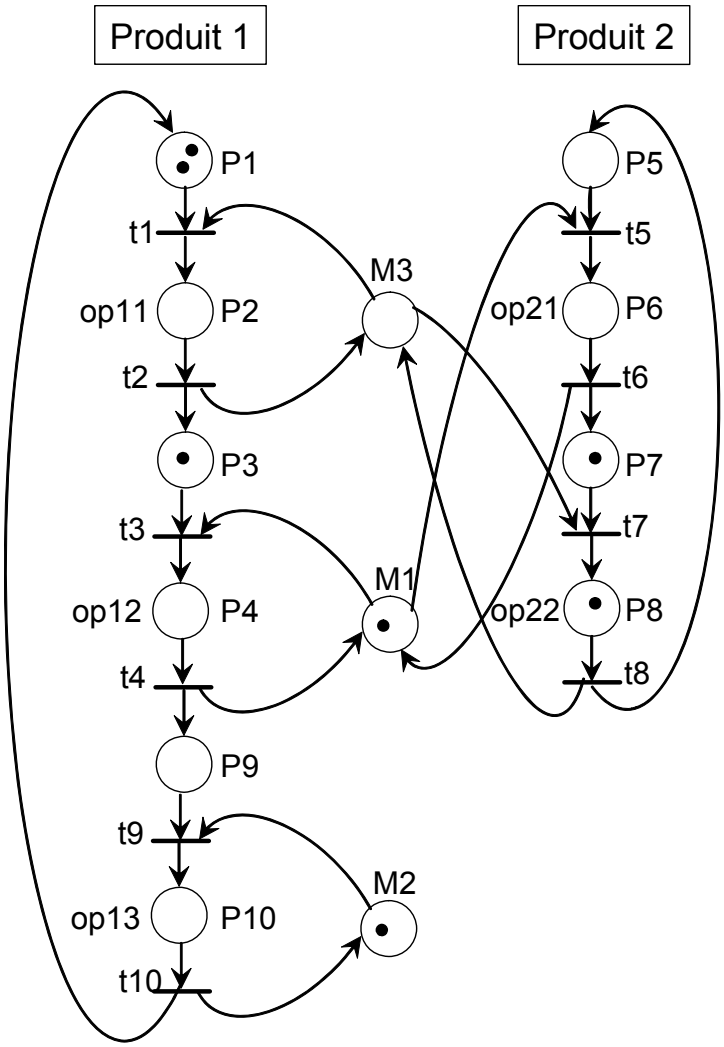


Figure 60. Gammes opératoires correspondant à l'ordonnancement cyclique de la Figure 59.

L'une des difficultés est d'établir le marquage initial des modèles RdP. Le marquage initial de chaque RdP de spécification doit correspondre à l'état de la machine correspondante au début du cycle. Le marquage initial des gammes opératoires doit également refléter cet état initial des machines. Le gantt de la Figure 59 montre qu'à l'instant t=0 de début de chaque cycle, la pièce w1 a déjà subit l'opération OP11 lors du cycle précédent est doit commencer l'opération OP12. Nous marquons ainsi la place P3 pour modéliser cet état de w1. w2 et w3 sont toutes deux en attentes de leur première opération. Nous marquons donc la place P1 de la

gamme OS1. Les pièces du type OS2, ont toutes deux subis la première opération de la gamme lors du cycle précédent. La pièce w4 est en cours de réalisation de l'opération OP22 et w5 est en attente de la ressource M3. D'où le marquage initial des RdPs de la Figure 60. Dans cette figure nous avons représenté les mutexes modélisant le partage des ressources machine pour la réalisation des opérations de transformation. On peut remarquer que la machine 2 n'est utilisée que par la gamme OS1 afin de gérer son parallélisme de flux. Aussi, dans un premier temps, afin de simplifier l'étude de ce problème nous ne considérons que le système composé de la machine 1 et de la machine 3. Nous pouvons leur associé les allocateurs de la Figure 61. Nous avons précisé dans cette figure le « cycle pièce » des opérations exécutées par chaque machine. Si nous supposons que le cycle courant est par rapport aux pièces, le cycle d'ordre 'n', ainsi la place P12 modélise l'opération 'OP12' sur la pièce w1 devant sortir du système de production dans le cycle courant. La place P28 modélise par contre l'opération 'OP11' réalisée sur la pièce w1 du cycle 'n+1'.

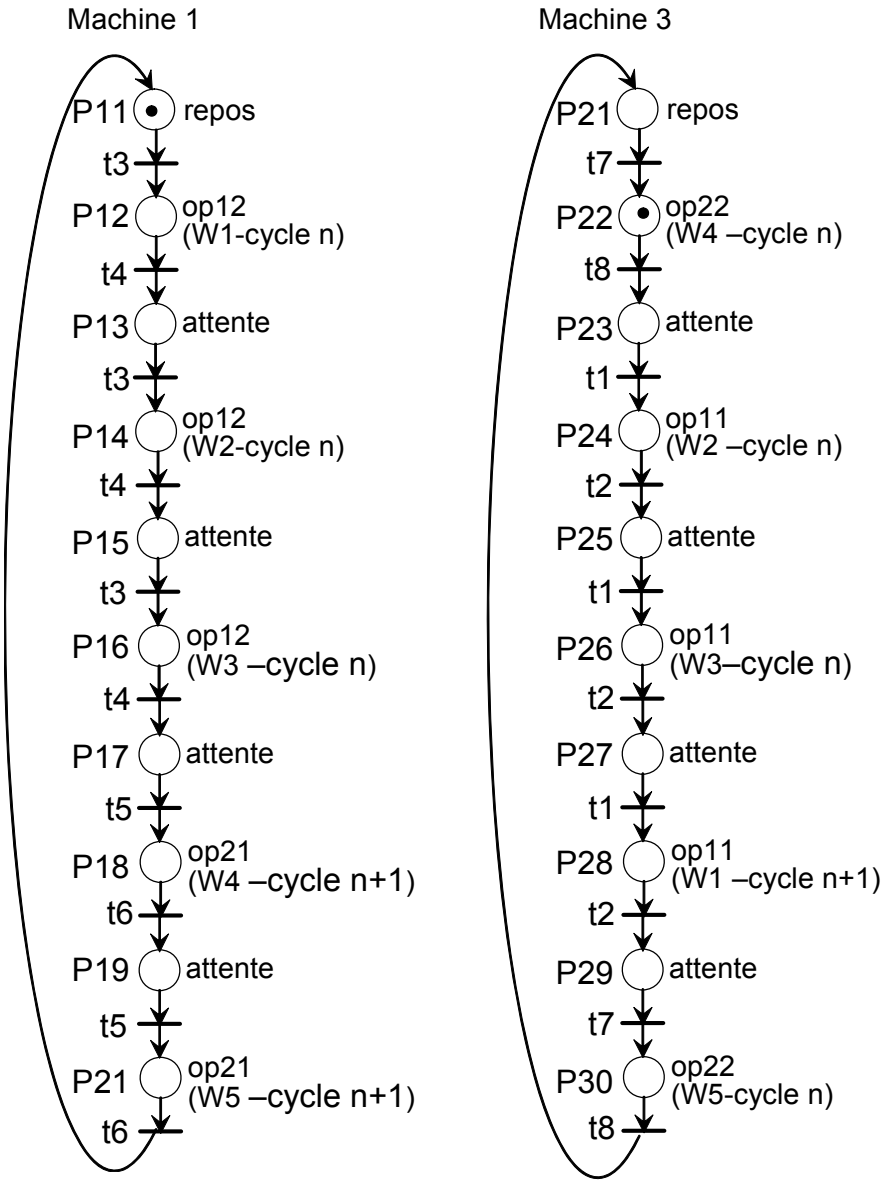


Figure 61. Allocateurs correspondant aux machines 1 et 3.

D'après les règles de synthèse énoncées au paragraphe II.3, les transitions 't3' et 't8' sont immédiatement franchissables car sensibilisées à la fois dans le modèle du procédé et dans le modèle de spécification. Le franchissement de 't8' modélise la fin de l'opération OP22 réalisée sur la pièce w4 du cycle 'n'. Franchir 't8' avant 't3' correspond à un état du système qui n'est pas spécifié par l'ordonnancement cyclique de la Figure 59. Cela signifie-t-il que notre modèle RdP de spécification n'est pas le bon ? Ce problème est dû au fait que nos modèles ne codent pas les durées associées aux différentes opérations. Pour cela, il nous faudrait utiliser des RdP-P temporisés. Actuellement, l'outil logiciel que nous avons

développé est uniquement basé sur l'exploitation des RdP généralisés. Il en résulte que le comportement global du système modélisé à l'aide des RdPs ordinaires correspond à un sur-ensemble du comportement souhaité. En effet, la spécification modélisée sous forme de séquence des allocations/désallocations d'une machine revient à spécifier des ordres partiels entre les événements des différentes machines qui composent le système.

La première conséquence de cette spécification par ordre partiel est une explosion combinatoire des états du modèle de spécification. En effet dans l'exemple de la Figure 61, chaque RdP comptant 10 places, le graphe de marquage résultant compte 100 places. Si on considère l'ordonnancement de la Figure 59 rapporté aux machines 1 et 3, on peut établir qu'une séquence de 20 états est suffisante pour modéliser le comportement souhaité. Pour obtenir cette séquence, il suffit de ranger sous forme d'ordre total les événements correspondant aux débuts et fins d'opérations. Nous obtenons alors le RdP de la Figure 62. Normalement, comme certaines opérations se terminent ou commencent simultanément d'après notre ordonnancement (cf. Figure 59), nous n'avons pas réellement un ordre total de tous les événements produits. C'est par exemple le cas de l'opération OP13 sur w1 et de l'opération OP11 sur w2 qui se terminent simultanément (transitions 't2' et 't4'). Nous avons choisi arbitrairement dans le modèle RdP de la Figure 62, de placer la transition 't2' avant la transition 't4'. L'expérience montre que quelque soit l'ordre, on obtient le même résultat à partir du moment où ces deux transitions sont consécutives. Cela s'explique par le fait qu'il n'y a pas de notion de temps explicite dans notre modèle.

La deuxième conséquence est que certains des nouveaux états introduits par la spécification à base d'ordres partiels peuvent mener à des situations de blocage. En effet, les temps opératoires sont imposés par le procédé. Une désynchronisation due à un allongement d'une durée opératoire en raison d'une défaillance progressive peut amener le système à atteindre un état possible mais non prévu par l'ordonnancement cyclique.

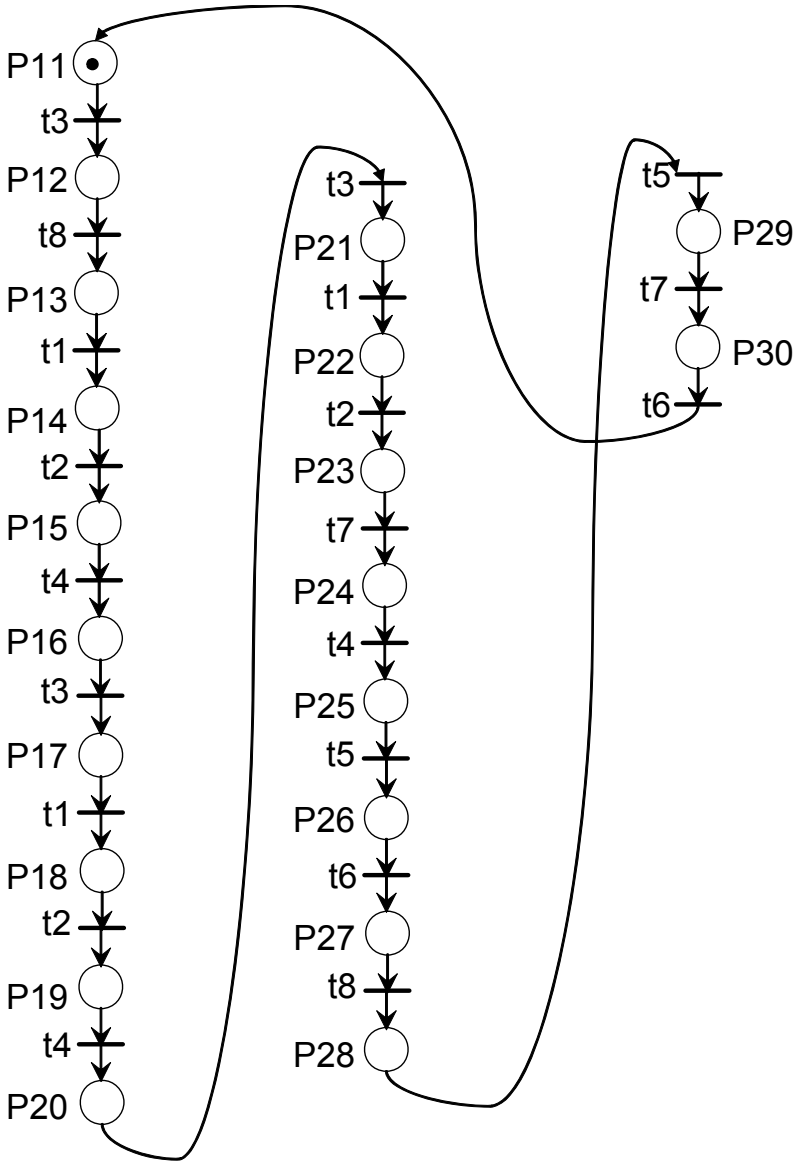


Figure 62. Spécification modélisant les événements produits par l'ordonnancement de la Figure 59

On ne peut toutefois pas trouver un contrôleur déterministe correspondant au modèle de spécification de la Figure 62. En effet, cette spécification ne permet pas d'imposer un comportement déterministe pour deux raisons. La première raison est due au fait que l'ordonnancement proposé n'est pas déterministe. En effet, d'une part toutes les opérations de la machine 3 à partir de OP11 sur w3 peuvent être anticipées d'une unité par rapport à l'ordonnancement considéré. D'autre part, l'opération OP22 sur w5 peut être permutée avec l'opération OP11 de w1. On peut trouver une équation de séparation d'événement pour interdire le franchissement de la transition 't7' pour la transformation de la pièce w5. Lors de la résolution on trouve que cette équation est incompatible avec une équation de

franchissement et que le système d'équations ne peut donc pas être résolu. Cela montre qu'en fait on ne peut pas trouver une spécification interdisant l'exécution de OP22. En fait, dans cette situation il faut choisir en temps réel de bloquer w5 et d'affecter la machine 3 à w1.

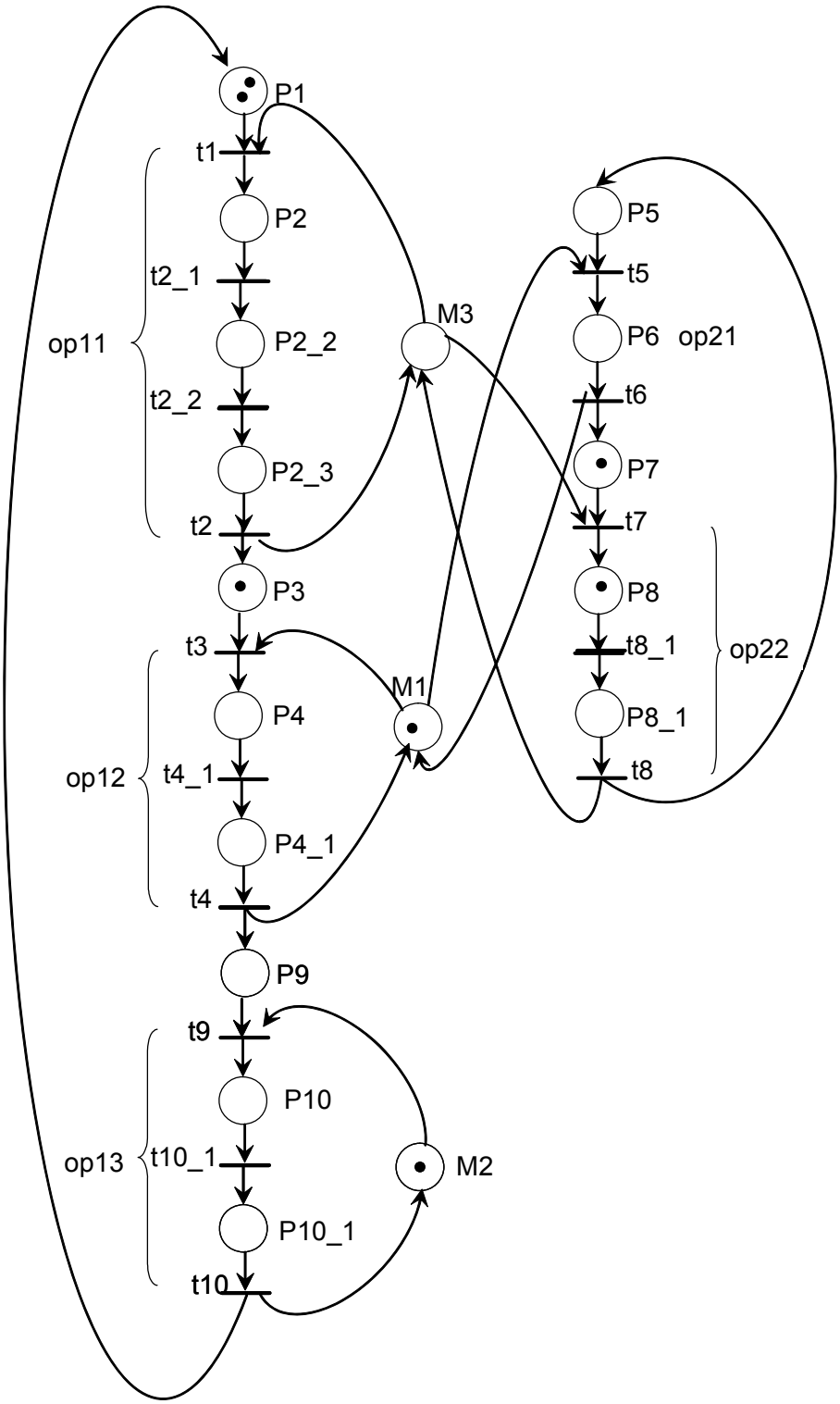


Figure 63. Modèle du procédé avec spécification des durées opératoires à l'aide des RdP ordinaires

Le non temporisation des opérations du modèle du procédé entraîne également des non-indéterminismes. En effet, du coup cette pièce passe plus vite d'une ressource à l'autre entraînant donc un indéterminisme d'allocation au niveau des machines suivantes. Afin de prendre en compte les durées opératoires, nous pouvons toutefois dupliquer les étapes modélisant une opération dans le modèle du procédé. En effet, si nous considérons que le franchissement de chaque transition correspond à une unité de temps, la modélisation d'une durée de 3 unités de temps correspond à une opération définie à l'aide de quatre transitions et 3 étapes (Figure 63). Notons sur cette figure que nous lors de l'extension du modèle du procédé, nous avons conservé à l'identique les noms des transitions qui modélisent le début d'une opération et sa fin. Par exemple la transition 't1' modélise toujours le début de l'opération 'op11' et 't2' sa fin. Cela permet de conserver exactement les mêmes modèles de spécification que pour le modèle du procédé standard. Mais cette approche en théorie intéressante ne l'est pas en pratique car elle provoque une explosion combinatoire due à la taille des modèles.

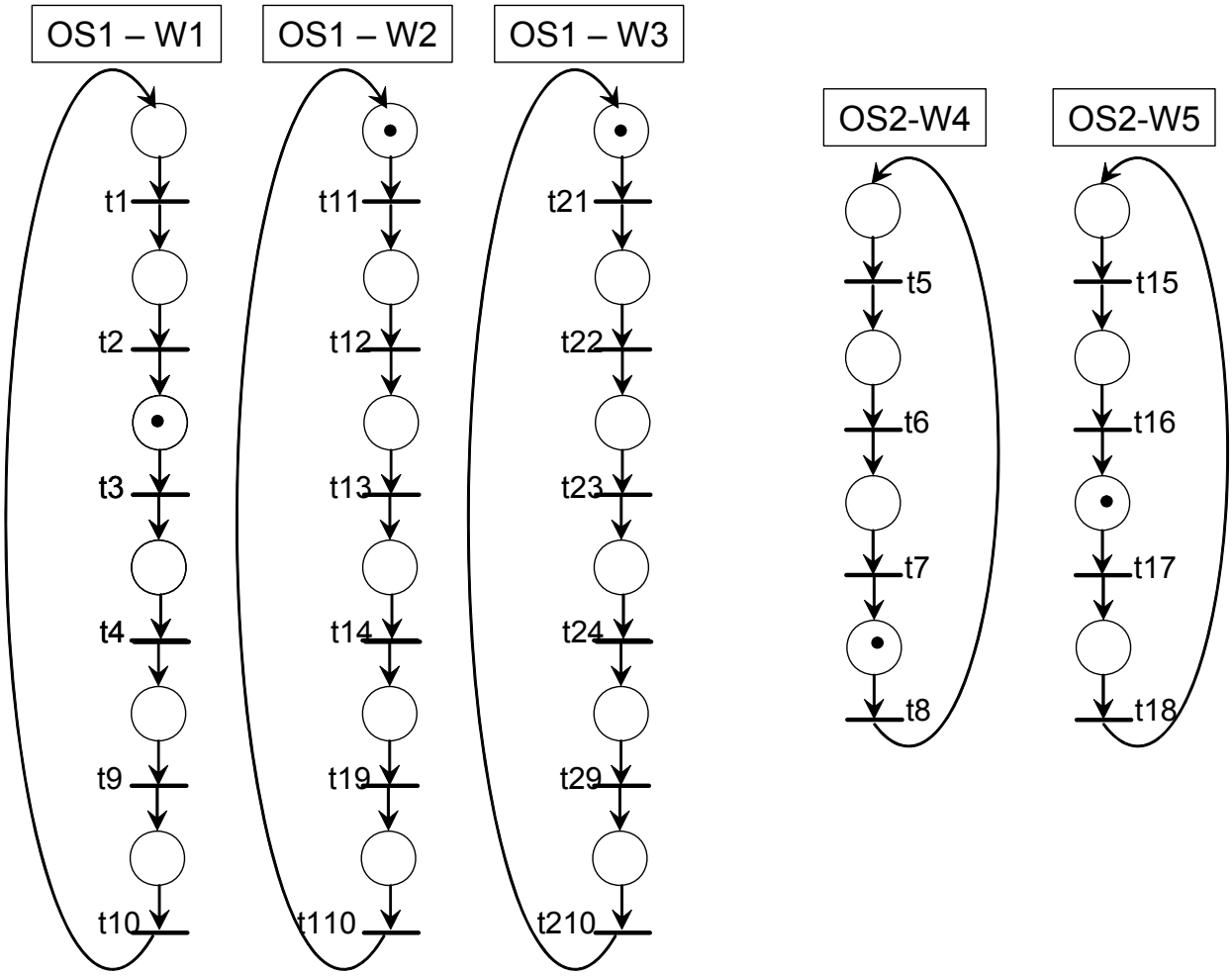


Figure 64. Dépliage du modèle du procédé

Une autre source d'indéterminisme est liée à la distinction des jetons du modèle du procédé. En effet, dans le diagramme de Gantt, nous avons nommé les pièces de manière à distinguer les opérations réalisées sur chacune d'elles. Dans la Figure 60, ce n'est pas le cas. Aussi, lorsque la pièce W2 a fini son cycle et est remise dans la place P1, la spécification permet de relancer sa fabrication dans le même cycle. Pour éviter cela, on peut réaliser un dépliage du modèle du procédé de la Figure 60 pour obtenir celui de la Figure 64. Dans ce cas, dans le modèle de spécification chaque transition ne pourra apparaître au plus qu'une seule fois permettant de vraiment quand l'opération réalisée sur une pièce doit être exécutée. Notons également que cette solution n'est applicable qu'à des modèles petites dimensions car elle entraîne une combinatoire importante.

4. Conclusion

Dans ce chapitre, nous avons montré que nous pouvons modéliser la commande des systèmes de manufacturiers en utilisant une approche complètement basée sur le formalisme des Réseaux de Petri afin de synthétiser la commande de coordination d'un système manufacturier. Pour la commande de son système de transport nous proposons d'utiliser le concept de supervision locale modulaire pour construire un modèle une commande modulaire facilement adaptable en fonction de la configuration matérielle du procédé. Pour la commande des produits implémentés sous-forme de gamme opératoire, nous avons montré que pour un ordonnancement prévisionnel déterminisme, nous pouvons de manière systématique des allocateurs de ressource mettant en œuvre de manière déterministe cette commande. Néanmoins l'utilisation des RdPs généralisés est une limite de nombre approche. En effet, elle nous oblige à aborder les problèmes de construction des allocateurs sous un angle purement logique. La difficulté à modéliser les spécifications temporelles du cahier de charges entraîne la présence d'indéterminismes qui rendent inconsistant le système d'équations de la théorie des régions. Nous avons proposé quelques solutions pour prendre en compte l'aspect temporel des opérations réalisées par les ressources, mais cela aboutit à des solutions trop combinatoires limitant l'intérêt de l'utilisation des RdPs. Une voie d'amélioration de l'approche serait donc l'utilisation de RdP temporisés et colorés.

Chapitre IV : Proposition d'un outil pour la synthèse à base de Réseaux de Petri

1. Introduction

L'objectif de la construction de cet outil est double. D'une part, il doit permettre de prouver de manière pratique la faisabilité des principales idées et concepts énoncés dans le chapitre précédent. Rappelons rapidement les idées principales. Dans ce chapitre, nous avons proposé un nouveau type de spécifications des utilisateurs modélisables à l'aide des Réseaux de Petri (ou RdP). Nous avons désigné le problème associé par le terme de « Séquences Interdites de Transitions d'État (ou SITE) ». Par la suite, nous avons proposé de réaliser une synthèse du comportement en boucle fermé d'un système à partir de deux modèles RdP synchronisés par certaines de leurs transitions : le modèle du procédé et le modèle des spécifications. Nous avons alors proposé la construction d'un graphe d'accessibilité synchrone contraint (ou GASC) qui sert de support à l'extraction des équations de la théorie de région. Cet outil doit donc prouver que la construction systématique du GASC est possible et que l'algorithme que nous avons proposé à la partie II.5.2 est correcte.

D'autre part, cet outil doit permettre de traiter rapidement de nombreux problèmes par le biais de synthèses automatiques. Cela nécessite la résolution automatique des équations de la théorie des régions associées à un problème. Dans cette partie, nous allons montrer comment résoudre ces problèmes à l'aide d'un solveur de problèmes linaires comme CPLEX.

Nous allons également illustrer l'intérêt de l'utilisation d'un tel outil de synthèse au travers de la synthèse de la commande en boucle fermée d'un exemple de système de production.

2. Présentation du problème de la recherche des équations obtenues par la théorie des régions

Lors de l'étude du contrôleur de l'exemple du producteur-consommateur au chapitre II, nous avons déterminé un système d'équations linéaires données par la théorie des régions. Ces systèmes d'équations comprennent d'une manière générale 3 sous-ensembles d'équations :

- Les équations correspondant aux séquences des marquages accessibles,
- Les équations des cycles,
- Les équations correspondant aux séquences interdites (séparation d'événement).

Chaque place de contrôle à ajouter au modèle initial doit vérifier toutes les équations des deux premiers sous-ensembles et au moins une équation du 3ème sous-ensemble. Définissons le système d'équations ainsi constitué de « sous-ensemble d'équations générateur » (d'une place de contrôle). A priori chaque équation de séquences interdites engendre une place de contrôle. La résolution du cas du producteur-consommateur au paragraphe II.6.2 a montré que la résolution d'une équation de séquences interdites peut donner plusieurs solutions. Un de nos objectifs est de minimiser le nombre de places de contrôle à ajouter au modèle du procédé pour obtenir le contrôleur en boucle fermé. Cela permet notamment en phase d'implémentation de limiter le nombre de variables nécessaires et donc d'optimiser les besoins en mémoires sur les calculateurs industriels. Le problème dans ce cadre est donc de déterminer quelle solution retenir pour chaque solution d'un « sous-système d'équations générateur » afin d'obtenir un nombre minimal de place de contrôles ? Lors de la résolution manuelle effectuée au paragraphe II.6.2 nous avions choisi de retenir les solutions redondantes inter-équations. Dans les paragraphes suivants nous allons proposer des solutions pour résoudre ce problème.

3. Recherche naïve de la solution optimale en nombre de places

Notre première idée a été d'implémenter un algorithme qui code le raisonnement que nous effectuons en manuel. L'idée est la suivante :

Algorithme 2 : Recherche naïve

S_{core} est l'ensemble composé des équations d'accessibilité et des équations de cycles,

S_{int} est l'ensemble des équations des séquences interdites,

$EQ_{int}[cpt]$ est la $cpt^{ème}$ équation interdite,

SG est le « sous-système d'équations générateur » composé de S_{core} et de $EQ_{int}[cpt]$,

ENSSOL[cpt][s] est la $s^{ème}$ solution de la $cpt^{ème}$ équation interdite,

***Etape 1** : (* Recherche des solutions de chaque sous-ensemble d'équations générateur *)*

Nb_{int}<- card(EQ_{int}) ; (Détermination du nombre d'équations interdites *)*

Pour cpt allant de 1 à Nb_{int} faire

SG<- S_{core} U EQ_{int} [cpt] ;

ENSSOL(cpt) <- solutions_de(SG) ;

fin_Pour ;

***Etape 2** : (* Pour chaque équation interdite détermination de sa solution ayant le plus de liens avec les autres équations *)*

Pour cpt allant de 1 à 'Nb_{int}' faire

Nb_{sol}[cpt] <- card(ENSSOL[cpt]) ; (Nombre de solutions de la $cpt^{ème}$ équation interdite *)*

fin_Pour ;

Pour cpt allant de 1 à 'Nb_{int}' faire

Pour s allant de 1 à Nb_{sol}(cpt) faire

Nb(cpt, s) <- 0 ; (Initialisation du compteur d'équations interdites ayant la même solution que la $cpt^{ème}$ équation interdite*)*

Pour i allant de 1 à Nb_{int} avec i <> cpt faire

Pour k allant de 1 à Nb_{sol}(i) faire

Si (ENSSOL(cpt,s)=ENSSOL(i,k)) alors Nb(cpt,s) <- Nb(cpt,s) + 1 ; fin_Si

fin_Pour ;

fin_Pour ;

fin_Pour ;

Indice_Max_SOL<-1 ;

Pour s allant de 1 à Nb_{sol}(cpt) faire

Si (Nb(cpt,s) > Nb(cpt, Indice_Max_SOL)) alors Indice_Max_SOL <- s ; fin_Si ;

fin_Pour ;

SOL(cpt)<- ENSSOL(cpt, Indice_Max_SOL)_;

fin_Pour ;

Etape 3 *: (* Indentification des solutions *)*

(Initialisation des solutions *)*

Pour cpt allant de 1 à Nb_{int} *faire inclus_SOL(j) <- faux ; fin_Pour ;*

(Choix itératif des solutions communes au plus grand nombre d'équations *)*

Pour cpt allant de 1 à 'Nb$_{int}$*-1' faire*

Si (Not inclus_SOL(cpt)) alors

SOLUTIONS<- SOLUTIONS + {SOL(cpt)} ;

Pour j allant de 'cpt+1' à Nb_{int} *faire*

Si (SOL(j)=SOL(cpt)) alors inclus_SOL(j) <- vrai ; fin_Si

fin_Pour

fin_Si

fin_Pour ;

Si (Not inclus_SOL(Nbint)) alors SOLUTIONS<- SOLUTIONS + {SOL(cpt)} ; fin_Si ;

Fin.

Cet algorithme est dit naïf car nous n'avons pas pu l'implémenter. En effet, nous avons implémenté sous CPLEX la fonction '*solutions_de*' résolvant chaque sous-système d'équations générateur. CPLEX est un environnement puissant permettant la résolution de systèmes en programmation linéaire. Malheureusement il donne uniquement la première solution qu'il trouve lors de la résolution d'un sous-système générateur et non pas l'ensemble des solutions possibles. Il nous a donc fallu chercher un algorithme adapté au type de fonctionnement de CPLEX. Dans la partie suivante, nous allons commencer par présenter le principe de fonctionnement de CPLEX.

4. Recherche d'un algorithme d'obtention d'un nombre minimal de solutions à l'aide de CPLEX

4.1 Modélisation d'un problème linéaire à l'aide de CPLEX

Pour résoudre un problème avec CPLEX, il est nécessaire de construire à partir des équations de la théorie des régions un problème d'optimisation linéaire. La formulation d'un tel problème nécessite la définition de trois parties :

- Une fonction objectif à minimiser ou maximiser du type $\sum_{i=1}^{m} d_i xi$ avec xi des variables du problème et les d_i des constantes ;
- D'un ensemble de contraintes : Chaque contrainte C_k peut s'écrire sous la forme : $\sum_{i=1}^{n} a_i^k xi \leq b^k$ avec les a_i^k et les b^k des constantes ;
- D'un espace de valeurs d'appartenance pour chacune des variables du problème : $e_i \leq x_i \leq E_i$ pour i allant de 1 à n.

Rappelons que notre problème consiste à trouver un nombre minimal de places de contrôles à ajouter au modèle RdP de la commande en boucle ouverte pour obtenir le fonctionnement en boucle fermé prenant en compte les spécifications de l'utilisateur. Dans notre cas les contraintes correspondent à un sous-ensemble générateur. C'est-à-dire qu'un ensemble de contraintes comprend les équations d'accessibilité, les équations de cycle et une équation de séparation d'événements. Dans ce cadre, afin de pouvoir coder notre problème sous CPLEX nous adoptons les notations suivantes :

- $x0$ est la variable désignant le marquage initial de la place de contrôle recherchée. Dans la notation matricielle du chapitre II, $x0$ est équivalent à $M0(pc)$. $x0$ est de valeur entière comprise entre 0 et +∞.
- t_i désigne l'arc reliant la place de contrôle pc à la i^{ème} transition du modèle RdP du procédé ; la notation matricielle équivalente est $C(pc,T_i)$. t_i est de valeur entière comprise entre -10 et 10 afin de spécifier que nous travaillons dans les RdP généralisés.

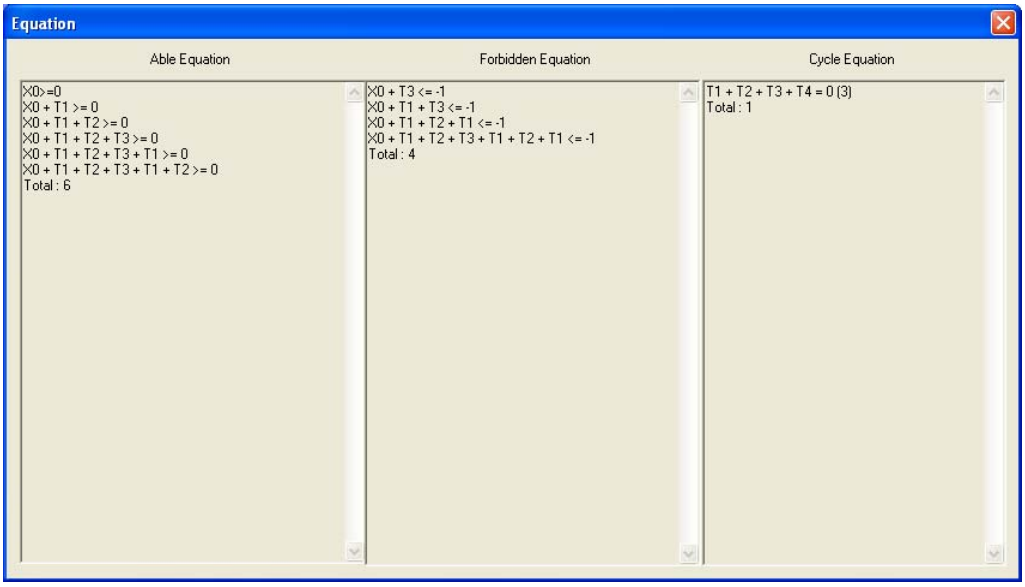


Figure 65. Équations de la théorie des régions obtenues par l'outil NetMDI

Pour spécifier les contraintes, CPLEX ne permet pas l'utilisation des opérateurs de comparaison stricte (< ou >). Aussi pour traduire les équations de séparation d'évènements qui sont de la forme « expression < 0 » nous les avons écrites sous la forme « expression <= -1 ». Nous avons choisi comme fonction objectif de minimiser le nombre de jetons dans la place de contrôle i.e. que notre fonction objectif correspond à la minimisation de la valeur de $x0$.

Considérons de nouveau l'exemple du producteur-consommateur utilisé au chapitre II (Figure 30). La Figure 65 donne l'écriture des différentes équations de ce problème obtenues à l'aide de notre outil en tenant compte qu'ensuite elles sont résolues avec CPLEX.

Nommons respectivement de 'f1' à 'f4' les quatre équations interdites de cet exemple. Si non considérons l'équation 'f1' correspondant à la place 'Pc1', nous pouvons construire le problème linéaire décrit sous l'application interactive de CPLEX par :

```
Minimize
obj: x0
Subject To
 c1: x0 >= 0
 c2: x0 + t1 >= 0
 c3: x0 + t1 + t2 >= 0
```

c4: x0 + t1 + t2 + t3 >= 0

c5: x0 + 2 t1 + t2 + t3 >= 0

c6: x0 + 2 t1 + 2 t2 + t3 >= 0

c7: t1 + t2 + t3 + t4 = 0

c8: x0 + t3 <= -1 (f1)

Bounds

-10 <= t1 <= 10

0 <= t2 <= 10

-10 <= t3 <= 10

0 <= t4 <= 10

Notons au niveau du domaine de définition des variables le fait que 't2' et 't4' sont définies comme étant forcément positif ou nul. Cette définition tient compte du fait que les transitions 't2' et 't4' sont définies en tant que transitions incontrôlables.

CPLEX nous donne alors comme solution [x0=0, t1=1, t2=0, t3=-1, t4=0]. Elle correspond à la deuxième solution trouvée au chapitre II pour la 1ère place de contrôle (solution notée $C\left(p_{c1}^{2},.\right)=\left(1 \quad 0 \quad -1 \quad 0\right)$ et $M_0\left(p^{2}{}_{c1}\right)=0$ dans la Figure 35 du chapitre II). Rappelons que la première solution déterminée lors de la résolution manuelle était $C\left(p_{c1}^{1},.\right)=\left(0 \quad 1 \quad -1 \quad 0\right)$ et $M_0\left(p^{1}{}_{c1}\right)=0$ soit [x0=0, t1=0, t2=1, t3=-1, t4=0]. C'est cette première solution que nous avions retenu car c'est une solution commune avec le problème linéaire construit à l'aide de l'équation interdite 'f2'. Le problème que nous avons ici est que CPLEX s'arrête à la première solution optimale qu'il trouve et nous ne pouvons pas le paramétrer pour qu'il trouve notre première solution. Donc, nous ne pouvons ni avoir toutes les solutions relatives à une équation de séparation, ni avoir la solution qui serait a priori commune à d'autres équations interdites.

Etant donné que la solution ainsi trouvé par CPLEX n'est pas une solution du système construit avec l'équation de séparation d'événement 'f2', nous avons cherché à déterminer si nous construisons d'emblée un problème linéaire avec 'f1' et 'f2', est-ce que nous aurons la solution commune? Nous avons donc soumis à CPLEX le problème linéaire suivant :

Minimize

obj: x0

```
Subject To
 c1: x0 >= 0
 c2: x0 + t1 >= 0
 c3: x0 + t1 + t2 >= 0
 c4: x0 + t1 + t2 + t3 >= 0
 c5: x0 + 2 t1 + t2 + t3 >= 0
 c6: x0 + 2 t1 + 2 t2 + t3 >= 0
 c7: t1 + t2 + t3 + t4 = 0
 c8: x0 + t3 <= -1          (f1)
 c9: x0 + t1 + t3 <= -1     (f2)
Bounds
-10 <= t1 <= 10
0 <= t2 <= 10
-10 <= t3 <= 10
0 <= t4 <= 10
```

La résolution par CPLEX de ce problème donne le résultat qu'on espérait, c'est-à-dire la première solution. La résolution du problème linéaire construit avec les équations de séparation d'événement 'f3' et 'f4' conduit à la même conclusion : on trouve la solution commune aux deux équations. Nous avons donc cherché une méthode d'utilisation de CPLEX permettant d'orienter CPLEX vers les solutions communes.

4.2 Détermination du nombre optimal de solutions par approche combinatoire

La manière la plus directe et la plus simple pour déterminer le nombre minimal de solutions consiste à construire des problèmes linéaires regroupant un nombre maximal d'équations de séparations d'événements. Le cas idéal c'est la détermination d'une solution pour un problème qui prend en compte toutes les équations de séparations d'événements. C'est-à-dire que si on *n* équations de séparations, au lieu d'avoir à ajouter *n* places de contrôles dans le modèle du procédé en boucle ouverte, il est suffisant de lui ajouter la place de contrôle, solution commune des *n* équations. L'idée est donc de partir des *n* équations

interdites et de faire des combinaisons d'équations pour un nombre décroissant. Cela signifie que si on a *n* équations interdites, dans l'absolu le nombre de combinaisons est :

$$\sum_{k=n}^{1} C_n^k = \sum_{k=n}^{1} \frac{n!}{k! \bullet (n-k)!}$$

Par exemple si nous avons un problème à 8 équations interdites, il faut donc dans le pire des cas (i.e. s'il n'y pas de solutions communes entre les équations) tester 255 problèmes linéaires résultants des combinaisons des différentes équations interdites, jusqu'au cas où chaque équation est testée séparément. Cet exemple montre que si *n* le nombre d'équations interdites est grand, la résolution prend beaucoup de temps et cela peut causer une saturation de la mémoire virtuelle.

Mais dans le cas général, il n'est pas nécessaire de tester toutes les combinaisons. En effet, si l'on trouve une solution commune à une combinaison de *m* équations, il suffit de poursuivre la recherche de solutions avec les *n-m* solutions restantes. Dans ce cas, si *m* est supérieur à $E\left(\frac{n}{2}\right)+1$, il ne peut plus y avoir une autre combinaisons de *m* équations parmi les *n-m* équations restantes et donc on peut arrêter la recherche des combinaisons de *m* équations. Il faut répéter cette recherche jusqu'à ce que chaque équation soit intégrée dans un regroupement (cf. Théorème 6).

Théorème 6 : Chaque équation de séparation ne doit apparaître qu'une seule et une seule fois dans l'un des regroupements des équations de séparation.

Preuve : On peut démontrer ce théorème par l'absurde. Si une équation apparaît dans deux regroupements distincts, cela signifie que cette équation engendre deux solutions en termes de places de contrôles. Donc les deux places contrôlent le franchissement de la transition correspondant à l'événement de séparation considéré. Pour que cette transition soit franchissable il faut que les deux places de contrôle soient marquées en même temps. Or comme l'équation apparaît dans deux regroupements distincts, il existe deux transitions distinctes entre elles et distinctes de la première transition qui sont respectivement contrôlées par chacune de ces deux places de contrôle. Si l'une de ces deux transitions a été franchie, donc la place de contrôle correspondant n'est plus marquée. Donc la première transition n'est

pas franchissable à ce moment. En fait rien ne garanti a priori que les deux places de contrôle puissent être marquées en même temps, donc la transition commune peut ne jamais être franchissable.

Nous avons donc proposé l'algorithme récursif suivant. Cet algorithme est construit en cherchant les regroupements d'équations interdites les plus grands possibles. On suppose que l'on part du plus grand regroupement (celui incluant toutes les équations) et on enlève au fur à mesure des équations jusqu'à l'identification d'une solution. On poursuit alors récursivement avec les reste des équations n'ayant pas été intégrée dans les regroupements précédents.

Algorithme 3 : Recherche combinatoire

S_{core} *est l'ensemble composé des équations d'accessibilité et des équations de cycles,*

S_{int} *est l'ensemble des équations des séquences interdites,*

SG *est le « sous-système d'équations générateur » composé de S_{core} et d'un sous-ensemble d'équations de S_{int},*

nb_{int} *est le nombre d'équations interdites dans SG*

k_{int} *est le nombre d'équations interdites à enlever pour obtenir une combinaison de $nb_{int} - k$ équations interdites,*

EQCombi *est une paire ensemble d'équations interdites et valeurs de la résolution associée,*

ResultCombi*[1.. nb_{int}] est un vecteur de dimension nb_{int} d'élément de type EQCombi, dont la composante d'ordre '$nb_{int} - k$' correspond à une liste de paire ensembles des équations interdites de cardinalité '$nb_{int} - k$' et valeurs associées à la résolution,*

EQCons*[1.. nb_{int}] est un vecteur de dimension nb_{int} dont l'indexe de chaque composante correspond à l'indexe d'une équation interdite ; par défaut toutes les composantes sont à 0 et quand l'équation interdite correspondante est intégrée à une combinaison solution, la composante correspondante est mise à 0.*

RestOf_EQCons*[1.. nb_{int}] est un vecteur de dimension nb_{int} donnant les équations non encore intégrées dans une combinaison.*

Etape 1 *: (* Initialisation du problème par rapport à l'ensemble des équations interdites *)*

SG <- S_{core} U S_{int} ;

nb_{int} <- card(SG) ;

Pour cpt=1 à nb_{int} faire EQCons [cpt] <- 0 ; fin_pour ;

***Etape 2** : (* On recherche l'ensemble des combinaisons ayant de solutions *)*

ChercherFeasibleCombi (EQCons, ResultCombi) ;

***Etape 3** : Afficher(ResultCombi) ;*

(La fonction suivante recherche récursivement les combinaisons maximales d'équations donnant une solution*)*

*Fonction **ChercherFeasibleCombi (EQCons, ResultCombi)**;*

k_{int} <- 0;

trouv <- faux;

Tant_que (k_{int} <= nb_{int} -1) et (not trouv)) faire

Si (ChercherFeasibleCombi2(nb_{int}, k_{int}, EQCons, EQCombi)=vrai) alors

Ajouter (EQCombi, ResultCombi) ;

RestOf_EQCons<- EQCons ;

Si (tous_a_un(RestOf_EQCons)=faux) alors

ChercherFeasibleCombi(RestOf_EQCons, ResultCombi)

fin_si;

trouv<-vrai;

sinon k_{int} <- k_{int} +1;

fin_si;

fin_TQ;

fin;

(Permet de déterminer les combinaisons comprenant 'n-k' équations interdites *)*

*boolean Fonction **ChercherFeasibleCombi2 (n, k, EQCons, ResultCombi)**;*

L'implémentation de cet algorithme a montré la faisabilité de notre approche. Malheureusement, confronté à des exemples donnant une trentaine d'équations interdites, il faut pratiquement une journée pour déterminer les solutions. D'autre part, l'algorithme ne donne pas forcément le nombre minimal d'équations. Pour l'illustrer, imaginons par exemple un système nous donnons une dizaine d'équations de séparations d'événement (ou équations interdites). Avec notre algorithme on peut trouver une première solution maximale commune à six équations interdites. Imaginons dans ce cas que les quatre équations ne puissent pas être combinées en nous conduisent à trouver une solution comprenant 5 places de contrôle. Imaginons maintenant une autre résolution permettant de trouver une combinaison à l'aide de 4 équations et deux autres de 3 équations. Cela nous donnerait une solution de 3 places de

contrôle meilleur que la précédente. L'idée exprimée ici est que des combinaisons d'une nombre maximal d'équations ayant une solution commune ne conduit pas forcément à un nombre minimal de places de contrôle.

5. Application à un exemple de système manufacturier

L'objectif de cette partie est d'illustrer l'application de la méthode proposée à un exemple de système manufacturier. Ainsi notre méthode de synthèse sera appliquée à une partie de la cellule flexible de l'Ecole Centrale de Lille.

5.1 Présentation du système de production

Pour illustrer l'application de notre démarche, nous considérons le système de production schématisé par la Figure 66. Ce système est composé de deux machines outils à commande numérique (CU1 et T1) et de deux robots permettant le chargement/déchargement de ces machines. Les caractéristiques de ces différentes ressources sont les suivantes :

- Un tour 'T1' à commande numérique intégrée. Ce tour possède deux axes et une tourelle pouvant comporter 12 outils. Ce tour peut réaliser deux types de tournage notés 'tournage 1' ou 't1' et 'tournage 2' ou 't2'.
- Une centrale d'usinage 'CU1' à commande numérique. Celui-ci est commandable sur quatre axes et est équipé d'un système de palettisation linéaire alternatif, composé de deux table-palettes (deux tampons d'entrée/sortie mono pièce). Un changeur automatique d'outils avec un magasin disques est intégré à la machine.
- Un robot de manutention universel 'R1' programmable par langage spécialisé (LM, NUM…). Il possède 6 axes de liberté.
- Un robot de manutention universel 'R2'. Ce robot possède 6 axes de liberté et dispose d'un système de changement d'effecteurs dont les déplacements sont enregistrés par apprentissage.
- Un tampon d'entrée des pièces notés INPUT et un tampon de sortie des pièces noté OUTPUT.

- Un stock intermédiaire A1 de capacité unitaire positionné entre les deux machines.

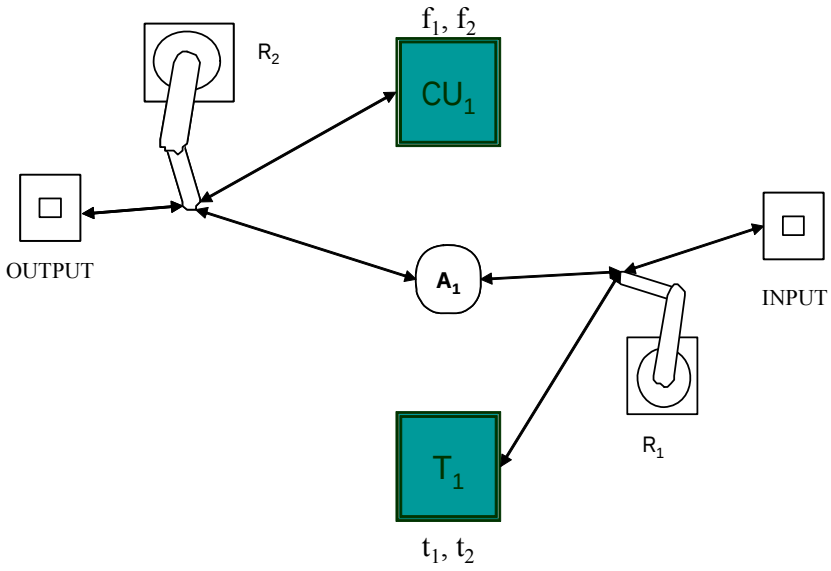


Figure 66. Système de production

Les relations d'accessibilités entre les différents lieux caractéristiques (cf. chapitre I) sont représentées par des flèches sur le schéma de la Figure 66. Ainsi le robot R1 permet de transférer les pièces du tampon d'entrée INTPUT directement vers la machine T1 ou vers le stock intermédiaire A1. Le robot R2 permet de transférer des pièces du centre d'usinage CU1 vers le (ou du) stock A1 ou vers le tampon de sortie OUTPUT. On voit ainsi que pour transférer une pièce du stock d'entrée INPUR vers le centre d'usinage, il faut d'abord que le robot R1 dépose la pièce dans le stock A1 et qu'ensuite le robot R2 transfert cette pièce sur CU1.

Nous considérons que la production courante consiste en trois types de pièces :

- Des éprouvettes de fatigue oligocycliques, référencées comme étant des pièces de type OS1. Pour obtenir ces pièces, la matière brute d'usinage subit un tournage de type 1 (tournage 1).
- Des pots de porte crayons, référencés comme pièces de type OS2. Pour obtenir ces pots, la matière brute subit une opération de fraisage (fraisage 1) permettant de creuser le pot et de l'amener aux dimensions définies par le cahier de charges.

- Des socles de porte crayon, référencés comme pièces de type OS3. Ces pièces sont obtenues par le l'enchaînement d'un tournage de type 2 (tournage 2) et ensuite un fraisage de type 2 (fraisage 2).

La Figure 67 illustre les gammes opératoires respectives de ces trois produits. Pour construire ces gammes opératoires ont été simplifiées en ne faisant pas apparaître les opérations de transfert

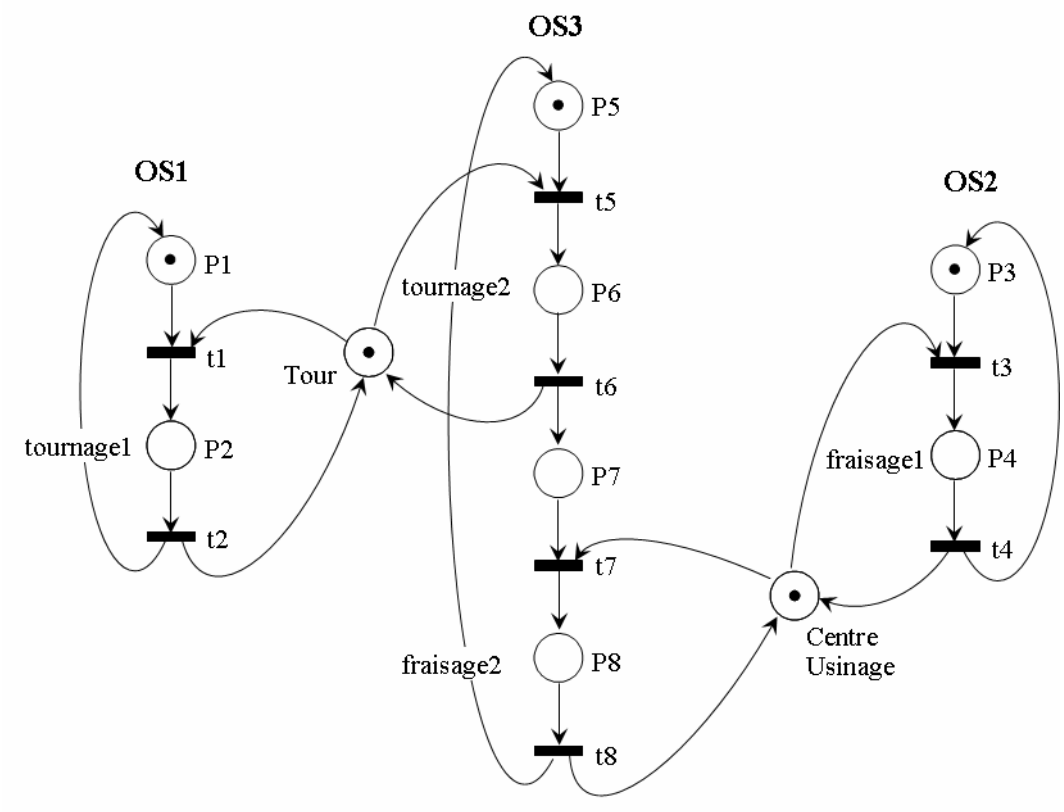


Figure 67. Gammes opératoires

5.2 Contrôleurs des pièces

Dans cette section, nous nous intéressons à la construction des contrôleurs des pièces. Pour obtenir ces contrôleurs nous définissons dans un premier temps le modèle du procédé en boucle ouverte. Pour construire ce modèle, nous partons du modèle des gammes opératoires de la Figure 67. En terme d'objectif de production, nous supposons que nous souhaitons à chaque cycle sortir une pièce OS1, une pièce OS2 et une pièce OS3.

Nous supposons qu'initialement toutes les pièces sont dans l'état brut. D'autre part, comme dans les gammes opératoires nous nous intéressons uniquement à la définition d'allocateur de machines et qu'ici le problème est posé en termes d'allocateur logique, nous simplifions le modèle des gammes en supprimant les places modélisant les opérations de transfert entre lieux caractéristiques. Afin de faciliter la compréhension de la construction des modèles de spécifications des allocateurs des machines, nous avons modélisé par des mutexs les ressources critiques partagées par les différentes gammes (Figure 67).

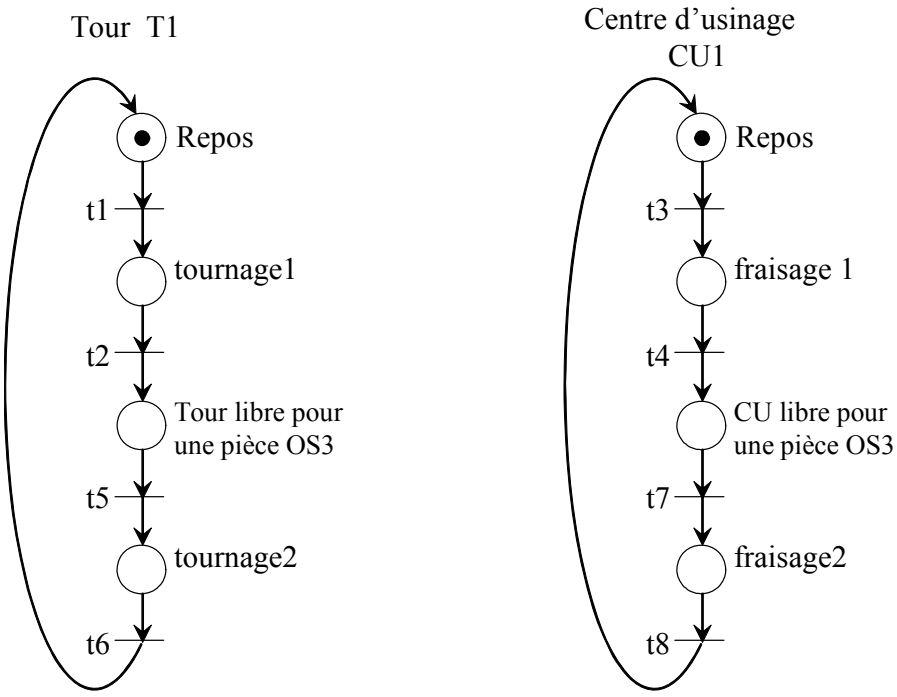


Figure 68. Modèle de spécification

Pour construire le modèle de spécification de chaque allocateur, nous modélisant la séquence des opérations exécutées par les machines d'après le modèle de procédé en boucle ouverte. Ainsi pour le tour T1, nous supposons que dans un premier temps nous allouons le tour à la pièce OS1 pour réaliser l'opération de tournage 1, et qu'ensuite le tour est alloué à la pièce OS3 pour réaliser l'opération de tournage 2. L'allocation du centre d'usinage CU1 est spécifiée de la même manière en allouant la machine d'abord à la pièce OS2 puis à la pièce OS2.

La réalisation de la synthèse à l'aide de l'outil NetMDI nous donne les ensembles d'équation illustrés par Tableau 4. Malgré l'encours minimal de 3 pièces, le système comprend 24 équations d'le système comprend 24 équations d'accessibilités, 32 équations interdites et 1 équations de cycle.

Tableau 4. Ensemble d'équations correspondant à la spécification OS1-OS2-OS3

Equations d'accessibilité (24)	**Equation interdites (32)**	**Equation de cycles (1)**
X0>=0 X0 + T1 >= 0 X0 + T3 >= 0 X0 + T1 + T2 >= 0 X0 + T1 + T3 >= 0 X0 + T3 + T4 >= 0 X0 + T1 + T2 + T3 >= 0 X0 + T1 + T2 + T5 >= 0 X0 + T1 + T3 + T4 >= 0 X0 + T1 + T2 + T3 + T4 >= 0 X0 + T1 + T2 + T3 + T5 >= 0 X0 + T1 + T2 + T5 + T6 >= 0 X0 + T1 + T2 + T3 + T4 + T5 >= 0 X0 + T1 + T2 + T3 + T5 + T6 >= 0 X0 + T1 + T2 + T5 + T6 + T1 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 >= 0 X0 + T1 + T2 + T3 + T5 + T6 + T1 >= 0 X0 + T1 + T2 + T5 + T6 + T1 + T2 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T7 >= 0 X0 + T1 + T2 + T3 + T5 + T6 + T1 + T2 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T7 >= 0 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 + T7 >= 0	X0 + T5 <= -1 X0 + T1 + T5 <= -1 X0 + T3 + T5 <= -1 X0 + T1 + T2 + T1 <= -1 X0 + T1 + T3 + T5 <= -1 X0 + T3 + T4 + T3 <= -1 X0 + T3 + T4 + T5 <= -1 X0 + T1 + T2 + T3 + T1 <= -1 X0 + T1 + T2 + T5 + T1 <= -1 X0 + T1 + T3 + T4 + T3 <= -1 X0 + T1 + T3 + T4 + T5 <= -1 X0 + T1 + T2 + T3 + T4 + T1 <= -1 X0 + T1 + T2 + T3 + T4 + T3 <= -1 X0 + T1 + T2 + T3 + T5 + T1 <= -1 X0 + T1 + T2 + T5 + T6 + T7 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T1 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T3 <= -1 X0 + T1 + T2 + T3 + T5 + T6 + T7 <= -1 X0 + T1 + T2 + T5 + T6 + T1 + T7 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T3 <= -1 X0 + T1 + T2 + T3 + T5 + T6 + T1 + T7 <= -1 X0 + T1 + T2 + T5 + T6 + T1 + T2 + T1 <= -1 X0 + T1 + T2 + T5 + T6 + T1 + T2 + T7 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T3 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T7 + T3 <= -1 X0 + T1 + T2 + T3 + T5 + T6 + T1 + T2 + T1 <= -1 X0 + T1 + T2 + T3 + T5 + T6 + T1 + T2 + T7 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 + T1 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 + T3 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T7 + T3 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 + T7 + T1 <= -1 X0 + T1 + T2 + T3 + T4 + T5 + T6 + T1 + T2 + T7 + T3 <= -1	T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 = 0

La résolution de ce système conduit au modèle donné par la Figure 69. Les places de contrôle S1 et S2 modélisent l'alternance de l'allocation du tour T1 entre la gamme du produit OS1 (place S1) et la gamme du produit OS3 (place S2). De même les places S3 et S4 modélisent la même de l'allocation du centre d'usinage CU1 entre respectivement la gamme OS2 et la gamme OS3.

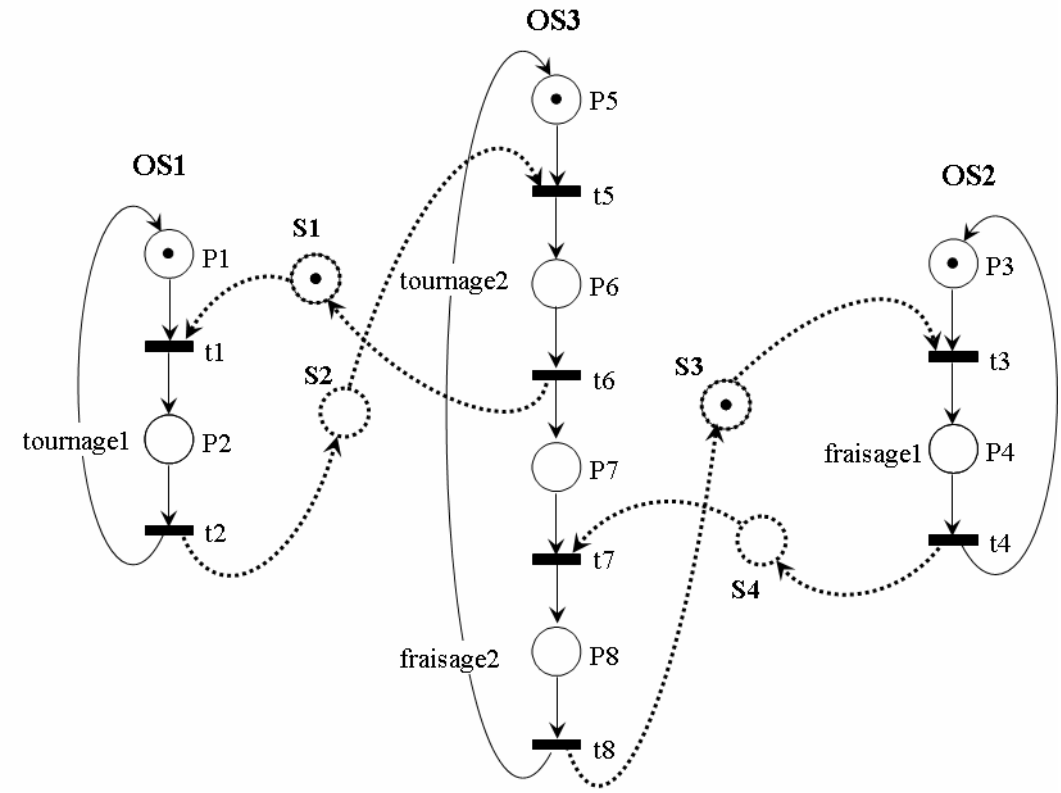


Figure 69. Contrôleur déterministe pour la réalisation du cycle OS1-OS2-OS3

5.3 Contrôleurs des ressources : coordination du système de transport

Le système de transport est composé des deux robots R1 et R2. Le fonctionnement en boucle ouverte de ces deux robots est donné par le schéma de la Figure 70. Sur cette figure nous donnons le modèle RdP de commande de chacun de ces robots. Prenons par exemple le cas du robot R1. D'après le schéma de la Figure 66, le robot R1 peut effectuer 4 transferts. Nous notons chaque transfert par 'L1->L2', notation dans laquelle 'L1' représente le lieu de départ et 'L2' le lieu d'arrivée. Ainsi la place P12 modélise un transfert du tampon INPUT vers le tour T1.

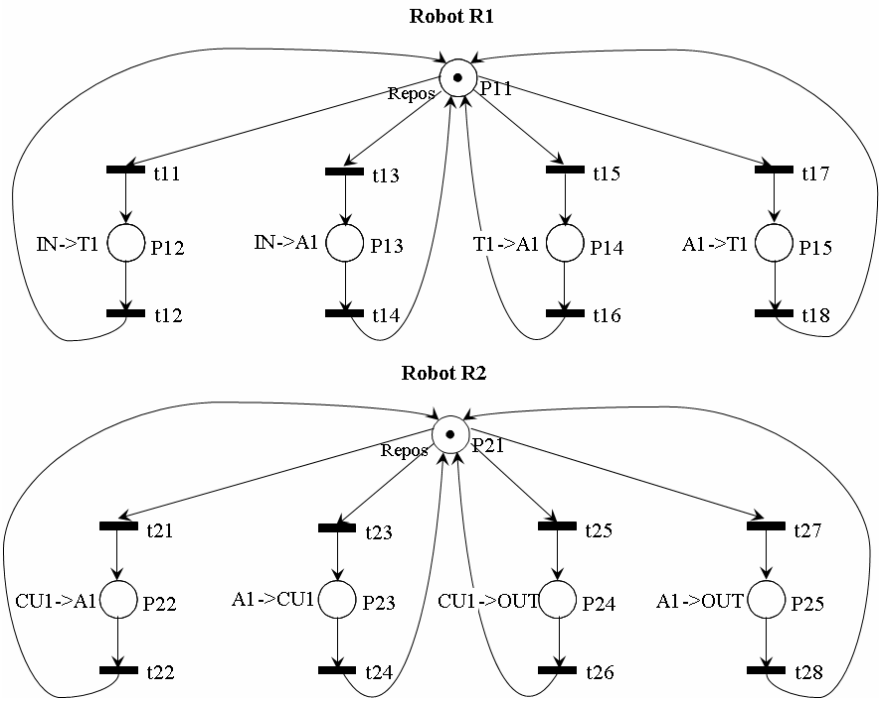


Figure 70. Modèle en boucle ouverte de la commande des deux robots

Comme nous pouvons le noter sur la Figure 70, les deux robots peuvent accéder au stock intermédiaire A1. Pour des raisons de sécurité, il n'est pas souhaitable que les deux robots accèdent en même temps à ce stock. Afin de limiter l'accès à ce stock, nous pouvons définir la spécification informelle suivante : on peut transférer une pièce dans le stock A1 que si il est libre ; on peut prélever une pièce du stock A1 que s'il contient une pièce. A partir de cette spécification, nous pouvons établir le modèle RdP donné par la Figure 71.

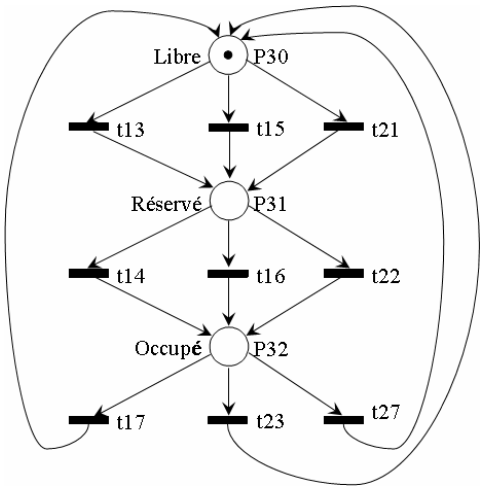


Figure 71. Modèle RdP de la spécification sur l'accès des robots R1 et R2 à la zone A1

Le modèle RdP traduisant la spécification d'accès au stock A1 est caractérisé par trois états : libre (place P30), réservé (place P31) en occupé (place P32). On passe de l'état 'libre' à l'état 'réservé' dès que l'un des robots commence un transfert vers la zone A1. Ainsi le robot R1 peut effectuer deux transferts vers A1 (IN->A1 et T1->A1) dont le début d'opération est caractérisé par les franchissements des transitions 't13' et 't15'. De même le robot R2 peut effectuer le transfert de CU1->A1 caractérisé par le franchissement de la transition 't21'. Dès le franchissement d'une de ces trois transitions, le stock A1 est réservé. De même à la fin de l'opération de transfert ('t14', 't16' et 't22'), le stock est occupé.

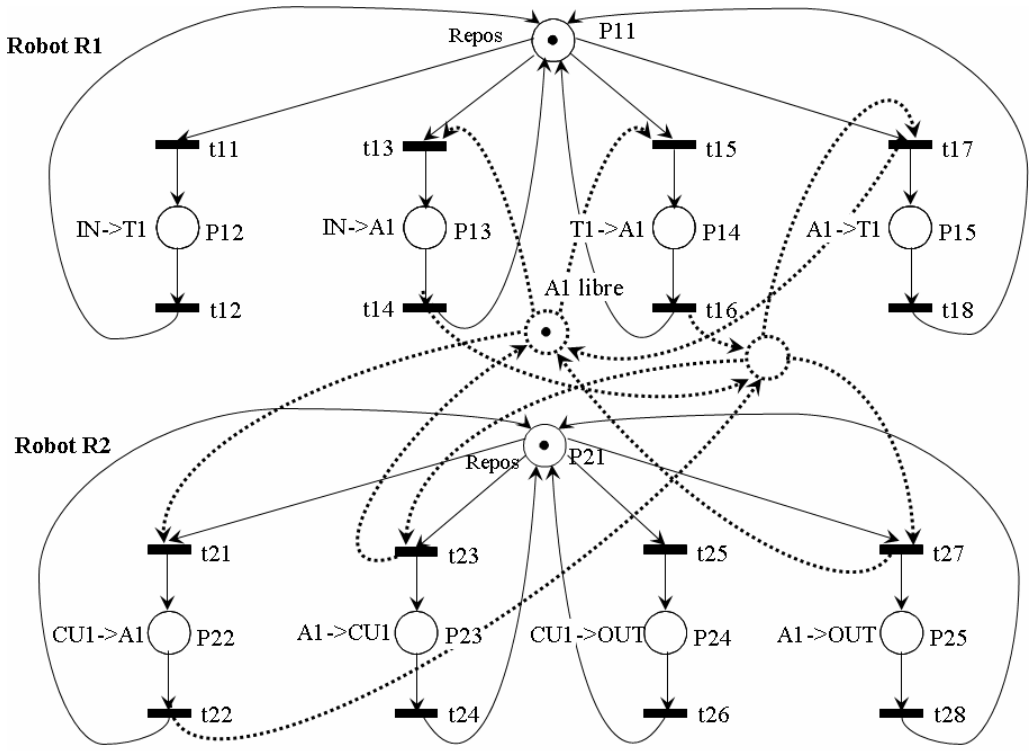


Figure 72. Modèle en boucle fermé de la commande des robots par rapport à l'accès à A1

La synthèse du modèle en boucle ouverte et du modèle de spécification donne le RdP de la Figure 72. On peut constater sur cette figure que la place 'A1 libre' contrôle les transitions de début de transfert vers A1 et la place 'A1 occupé' contrôle les transferts à partir de A1.

6. Conclusion

Dans ce chapitre nous avons dans un premier temps proposé différents algorithmes pour la résolution automatique des équations de la théorie des régions à l'aide d'un solveur comme CPLEX. La problématique de cette résolution est la détermination d'un nombre minimale de place de contrôle pour obtenir le comportement en boucle fermé lorsque le système est contrôlable. Cette résolution est rendue difficile par le fait que les équations de séparations d'événement sont des contraintes disjonctives. En raison du fait que CPLEX n'énumère pas toutes les solutions relatives à un système d'équation contenant une équation de séparation et que l'on ne peut pas orienter le mode de résolution de CPLEX, il est nécessaire de résoudre directement le système formé par la combinaison de l'ensemble des équations de séparation ayant une place de contrôle commune. Pour cela nous avons proposé un algorithme dit combinatoire en raison d'une recherche systématique des différentes combinaisons. Cet algorithme pose deux problèmes. D'une part il prend trop de temps dans le cas de la résolution de problème un peu compliqué comme l'exemple de Valentin [VAL94] utilisé au chapitre III pour poser le problème de la spécification de contrôleurs de pièces déterministes à partir d'un ordonnancement. D'autre part, il n'est pas optimal comme nous l'avons montré à la section 4. Pour trouver une solution à ce problème, il nous semble qu'il est nécessaire d'utiliser une approche permettant d'énumérer de manière efficace toutes les solutions relatives aux systèmes obtenus à l'aide de l'ensemble des contraintes disjonctives (équations de séparation). Une possibilité nous semble résider dans l'utilisation de la programmation par contrainte [BOU04].

Conclusion générale

Le problème du contrôle/commande des Système à Événements Discrets (SED) fait l'objet de nombreuses études actuellement. Pour ce type de système (SED), il s'avère nécessaire de disposer d'une commande assurant un bon comportement en fonction de la structure et de l'état du procédé commandé. Wonham et son équipe ont proposé la théorie du « Supervisory Control » (Contrôle par Supervision) pour modéliser la commande des SED en utilisant le formalisme des automates à états finis. Le contrôle par supervision propose de synthétiser la commande d'un procédé en distinguant d'une part le modèle des potentialités du système (fonctionnement en boucle ouverte) et d'autre part les contraintes de fonctionnement imposées au système (modèle de spécification). La synthèse du modèle en boucle ouverte et des spécifications de fonctionnement donne le modèle en boucle fermé du système.

Pour effectuer la synthèse d'une commande, il convient donc d'établir un modèle de spécifications du système considéré. Ce modèle doit être capable de prendre en compte de manière efficace, les différentes contraintes de fonctionnement du système de production. Aussi nous avons choisi les Réseaux de Petri (RdP) comme formalisme de modélisation pour leur efficacité et leur simplicité à coder des fonctionnements parallèles et des synchronisations. Dans ce travail, nous avons montré qu'aujourd'hui dans la littérature, la plupart des approches à base de RdP sont fondés sur la spécification de contraintes de fonctionnement de type états interdits. La limite de ces approches est en général l'utilisation du graphe d'accessibilité ou de ces corollaires comme les invariants de marquage généralisés. Outre les aspects combinatoires généralement liés à ces approches, au chapitre II, nous avons montré que tous les types de spécifications utilisateur ne peuvent pas être exprimés sous forme d'états interdits ou de transitions interdites. Nous aborderons les différents types des spécifications d'utilisateurs qui se prêtent bien à l'utilisation du formalisme des RdP.

Dans le premier chapitre de ce mémoire nous avons présenté le contexte de nos travaux et positionné ainsi notre problématique relative au problème de contrôle/commande et plus particulièrement de la reconfiguration des Systèmes à Événements Discrets. Nous y avons ainsi présenté les principaux modèles existant dans la littérature comme outil de

synthèse de la commande. Ces approches permettent a priori de mieux garantir le respect du cahier de charges que les approches de type ingénierie de la commande. Des équipes de recherche commencent à proposer des extensions de la synthèse de commande permettant de surveiller et superviser les SED [NIE94], [NIE96], [LAF02], [LAF05]. Mais dans ce travail nous nous sommes focalisés sur les extensions portant sur la conception de la commande de coordination des systèmes manufacturiers à l'aide du formalisme RdP [ALP02] [BOU04] [GHA02c] [HOL90]. Nous nous sommes ainsi intéressés à l'approche d'Alpan qui est une méthode de synthèse mixte, basée sur l'utilisation des RdP pour la modélisation du procédé et l'utilisation des automates pour formuler les spécifications [ALP02]. Nous avons également analysé l'apport de la notion de Contraintes Généralisées d'Exclusion Mutuelle (ou CGEM), développées par des auteurs comme A. Giua [GIU92] [GIU93] ou Yamalidou [YAM96]. Nous avons montré les limites de ces travaux qui concernent soit l'aspect combinatoire des approches soit la difficulté à modéliser tous les types de spécifications. Dans ce cadre, nous avons montré que ces approches permettent de modéliser des problèmes de type transitions d'état interdites (ou PTEI) [GHA02] ou de type d'état interdit (ou PEI). Au deuxième chapitre, nous avons proposé le problème de Séquences Interdites de Transitions d'Etat (ou PSITE) qui est un type de spécification complémentaire avec ceux proposés jusque là dans la littérature.

Ce chapitre, nous également permis d'introduire un nouveau concept émergeant au niveau de la communauté travaillant sur les systèmes manufacturiers. Il s'agit du concept de Système Manufacturier Reconfigurable (ou SMR) qui est proposé par le professeur Y. Koren de la NSF Engineering Research Center aux Etats-Unis ([KOR99] et [KOR05]). Nous avons montré que pour concevoir ce type de système, il est nécessaire de pouvoir construire des commandes reconfigurables en ligne. La synthèse de commande est une solution dans ce cadre.

Les deuxième et troisième chapitres sont la base de notre contribution. Dans le deuxième chapitre nous nous sommes intéressés à l'utilisation de la théorie de régions de A. Gaffari et N. Rezg [GHA02a], [GHA02c] pour résoudre les problèmes de PSITE que nous y avons introduit. L'approche proposée est entièrement basée sur la modélisation du comportement potentiel du procédé et des contraintes de fonctionnement entièrement sous forme de modèles RdP. Nous avons donc proposé dans ce chapitre le « Graphe d'Accessibilité Synchrone Contraint » (GASC) comme outil support pour réaliser la synthèse de ces deux

modèles synchronisés par des transitions communes. L'idée de l'algorithme proposé est d'extraire progressivement les équations de la théorie de région par un parcours une exploration des états autorisés et des états interdits, sans avoir à construire au préalable le graphe d'accessibilité.

Dans le troisième chapitre, nous nous sommes intéressés à l'exploitation des résultats du chapitre II pour la construction de la commande de coordination d'un système de production. Notre objectif est en fait de proposer une approche mixte synthèse et ingénierie de la commande. Nous nous sommes donc situer dans le cadre du projet CASPAIM développé au LAGIS depuis de nombreuses années. Cela nous a amené a proposé une démarche de construction des modèles de coordination du système de transport d'une part et des contrôleurs des pièces d'autre part. Pour la synthèse du graphe de coordination du système de transport (GCST) nous proposons une approche modulaire inspirée du concept de modularité locale de Queiros et Cruz [QUE00a], [QUE00b], [QUE02]. Concernant les contrôleurs des pièces, nous avons montré que cette approche pouvait sous certaines conditions (ordonnancement cyclique déterministe) être utilisée pour réaliser la synthèse d'allocateurs de ressources déterministes pour l'interprétation des gammes opératoires.

Le quatrième chapitre nous a permis d'introduire l'outil NetMDI qui montre la faisabilité de nos idées et d'illustrer l'ensemble de la démarche proposée sur un exemple didactique de système de cellule de production robotisée. Dans ce chapitre, nous avons mis l'accent sur l'efficacité de la résolution des équations de la théorie des régions en programmation linéaire à l'aide d'un solveur non énumératif des solutions comme CPLEX.

Les perspectives de ce travail sont multiples. Elles concernent notamment l'outil NetMDI qui doit être rendu plus efficace pour permettre de synthétiser la commande de véritables systèmes industriels. Cet outil peut être rendu plus efficace au niveau des deux étapes de base rappelé ci-dessous : l'extraction des équations de la théorie des régions et la recherche d'un nombre minimal de places de contrôle. Pour ceux qui est de l'extraction des équations de la théorie des régions, nous nous intéressons à l'exploitation de techniques d'abstraction pour l'analyse de l'accessibilité dans l'espace d'états d'un modèle RdP. L'intérêt de cette abstraction à l'aide de la notion de steps est qu'elle permet d'énumérer rapidement des séquences admissibles sans qu'il y ait nécessité de construire le graphe d'accessibilité [BEN00], [BOU04]. Il faudra toutefois identifier comment formaliser dans ce cadre les notions de transition d'état interdit et de séquences de transitions d'états d'interdits.

Concernant la recherche du nombre minimal de places de contrôles relatives à un ensemble de contrainte disjonctives, c'est un problème ouvert. Si l'on reste dans le cadre de la programmation linéaire et d'outils tels que CPLEX, il va falloir trouver une formulation mathématique du problème permettant de contourner l'énumération combinatoire. Dans ce cadre, nous intéressons à l'extension des formulations comme celles proposées par A. Giua sur la recherche d'une solution pour ensemble de contraintes parmi lesquelles un certain nombre peuvent être disjonctives [GIU05], [GIU06]. Mais dans ce cadre, nous pensons également qu'une approche basée sur la programmation par contraintes peut être plus efficace car cette technique permet d'explorer rapidement un grand nombre de solutions afin de retenir celle qui correspond à un critère d'optimalité.

Parmi les autres perspectives qu'il nous semble intéressant d'explorer, nous pouvons également citer la nécessiter de pouvoir prendre en compte dans nos modèles la formulation de spécifications liées au temps et également de pouvoir distinguer des comportements du système en fonction des objets qu'il manipule. Cela pourrait se traduire par le passage de modèles RdP généralisés que nous avons utilisé dans cette étude à des modèles basés sur les RdP colorés et temporels. Le choix de ces types de RdP nécessitera sans doute une adaptation voire une remise en cause d'approche basée sur la théorie des régions.

Enfin, la dernière perspective envisageable est l'aide voire l'automatisation de l'écriture des modèles de spécifications. Nous avons vu au chapitre III que la transformation des ordonnancements cycliques en modèles RdP des contrôleurs de machines est assez systématique. On peut envisager d'utiliser des techniques de transformation de modèles pour réécrire en formalisme RdP des spécifications utilisateurs comme celles relatives aux ordonnancements ou aux contraintes de sécurité.

Bibliographie

[ACH04] Z. Achour, N. Rezg et X. Xie, *Supervisory Control of Partially Observable Marked Graphs*, IEEE Transactions on Automatic control, Vol. 49, No. 11, Novembre 2004.

[AFN91] AFNOR, *Organisation et gestion de la production industrielle – Concepts fondamentaux de la gestion de production*, Norme NF X 50-310, Décembre 1991.

[ALA86] P. Alanche, P. Salvi, G. Morel, M. Roesch, P. Lhoste et M. Salim, *Modélisation de la Partie Opérative et Application à la structuration de la commande des systèmes automatisée*, Journées AFCET - Méthode et Outils Modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, Mars 1986.

[ALP97] Gülgün Alpan, *Design and Analysis of Supervisory Controllers for Discrete Event Systems,* Ph.D. Dissertation, Rutgers, the Sate University of New Jersey, 1997.

[ALP02] Gülgün Alpan et Mohsen A. Hafari, *Synthesis of a Closed-Loop Combined Plant and Controller Model,* Systems, Man and Cybernetics, Part B, IEEE Transactions on Volume 32, Issue 2, pp.163–175, Avril 2002.

[AMA92] S. Amar, E. Craye et J. C. Gentina, *Une méthode hiérarchique de specification et de prototypage des systèmes de production flexible,* APII 1992 Vol. 6, No. 5, 1992.

[BAD95] E. Badouel, L. Bernardinello et P. Darondeau, *Polinominal algorithms for the synthesis of bounded nets,* Proceedings of CAAP'95, Springer Verlag LNCS 915, pp. 364-378, 1995.

[BEN00] A. Bennasser, *L'accessibilité dans les Réseaux de Petri : une approche basée sur la programmation par contraintes*, Thèse de doctorat, Université de Sciences et Technologies de Lille, 2000.

[BER98] P. Berruet, *Contribution au recouvrement des Systèmes Flexibles de Production Manufacturière : Analyse de la Tolérance et Reconfiguration*, Thèse de doctorat, Ecole Centrale de Lille, 1998.

[BER05] Berruet P., Lallican J.L, Rossi A., Philippe J-L, *A component based approach for the design of FMS control and supervision* , IEEE SMC 2005, pp. 3005-3011, Hawaii, October 2005.

[BON93] A. de Bonneval, *Mécanismes de reprise dans les systèmes de commande à événements discrets*, Thèse de doctorat, Université Paul Sabatier, Toulouse, Septembre 1993.

[BOU88] Jean-Pierre Bourey, *Structuration de la Partie procédurale du Système de Conmmande de Cellules de Production Flexibles dans l'Industrie Manufacturière*, Thèse, L'université des Sciences et Techniques de Lille Flandres Artois, 1988.

[BOU04] Thomas Bourdequd'huy, *Techniques d'abstraction pour l'analyse et la synthèse de réseaux de Petri,* Thèse de Doctorat, Ecole Centrale de Lille, 2004.

[CAB06] M.P. Cabasino, A. Giua, C. Seatzu, *Identification of Deterministic Petri Nets*, 8th International Workshop on Discrete Event Systems (WODES'2006), n° WA1.2, , Ann Arbor (Michigan), USA, July 10-12, 2006

[CHA95a] A. Chaillet-subias, *Approche multi-modèles pour la commande et la surveillance en temps réel des systèmes à événement discrets*, Thèse de doctorat, Université Paul Sabatier de Toulouse, Décembre 1995.

[COM91] M. Combacau, *Commande et surveillance des systèmes a événements discrets complexes : application aux ateliers flexibles*, Thèse de doctorat, Université Paul Sabatier, Décembre 1991.

[DAN00] N. Dangoumau, A.K.A. Toguyeni, M. Dupas et E. Craye, *Reconfiguration Process for automated Production Systems*, dans les proceedings de MCPL'2000, Grenoble, France, 3 - 8 Juillet 2000.

[GHA02a] A. Ghaffari, N. Rezg et X. Xie,.*Algebraic and Geometric Characterization of Petri Net Controllers Using the Theory of Regions,* Proceeding of the 2002 Sixth International Workshop on Discrete Event Systems - WODES'02, Reims, France, 2002

[GHA02b] A. Ghaffari, *Les réseaux de Petri pour le contrôle des systèmes à événements discrets*, Thèse de doctorat, LGPIM, Metz, 2002.

[GHA02c] A. Ghaffari, N. Rezg et X. Xie, *Live and Maximally Permissive Controller Synthesis Using Theory of Regions,* in Synthesis and Control of Discrete Event Systems, B. Caillaud et al. (eds). Kluwer Academic Publishers, pp.155-166, 2002.

[GIU92] A. Giua, F. DiCesare et M. Silva, *Generalized Mutual Exclusion Constraints on Nets with Uncontrollable Transitions*, Proc. IEEE Int. Conf. on Systems, Man, & Cybernetics (Chicago, USA), pp. 934-9, Octobre 1992.

[GIU93] A. Giua, F. DiCesare et M. Silva, *Petri Net Supervisors for Generalized Mutual Exclusion Constraints,* Proc. 12th IFAC World Congress, Sidney, Australia, Vol. 1, pp. 267-270, Juillet 1993.

[GIU05] A. Giua, C. Seatzu, *Identification of free-labeled Petri nets via integer programming*, 44th IEEE Conference on Decision and Control and 2005 European Control Conference CDC-ECC '05, Seville, Spain, pp. 7639- 7644, 12-15 December 2005

[GOU04] D. Gouyon, *Contrôle par le produit des systèmes d'exécution de la production : apport des techniques de synthèse*, Thèse de doctorat, Nancy-I, 2004.

[HOL90] Holloway L. E. et B. H. Krogh, *Synthesis of feedback Control Logic for a Class of Controlled Petri Nets*, IEEE Transactions on Automatic control, Vol. 35, No. 5, Mai 1990.

[HUV94] B. Huvenoit, *De la conception à l'implantation de la commande modulaire et hierarchisee de systémes flexibles de production manufacturière*, Thèse de doctorat, LAIL, Ecole Centrale de Lille, 1994.

[KOR99] Y. Koren, U. Heisel, F. Jovane, T. Moriwoki, G. Pritshow, A. G. Ulosy et H. Van Bruseel, *Reconfigurable Manufacturing Systems*, Annals of the CIRP, Vol.48, pp.527-540, 1999.

[KOR05] Y. Koren, *Reconfigurable Manufacturing and Beyond*, In the summary of keynote Speech of CIRP05 3rd International Conference on Reconfigurable Manufacturing, Ann Arbor, Michigan, Etats-Unis, 10-12 Mai 2005

[KUM96] R. Kumar et L. E. Holloway, *Supervisory Control of Deterministic Petri Nets with Regular Specification Languages*, IEEE Transactions on Automatic Control, Vol. 41, Issue 2, pp.245-249, Février 1996.

[LAF02] O. Contant, S. Lafortune, et D. Teneketzis, *Diagnosis of Distributed Discrete Event Systems: Architecture and Modular Verification Approach*, Control Group Report #CGR 04-04, Department of EECS, University of Michigan, 2002.

[LAF05] S. Lafortune, Y. Wang et T. S. Yoo, *Diagnostic Décentralisé des Systèmes à Événements Discrets*, In MSR'05 "Modélisation des Systèmes Réactifs" RS - JESA, V. 39, Grenoble, France, 5-7 Octobre 2005.

[LAM05a] Lamotte F., Berruet P., Philippe J.L, *A model for the reconfiguration of Manufacturing Systems*,16th Triennial World Congress on Automatic Control, Czech Republic, July 2005.

[LAM05b]Lamotte F., Berruet P., Philippe J.L., *Using model transformation for the analysis of the architecture of a reconfigurable system*, IMACS 2005 world congress, Paris, July 2005.

[LEE04] E. J. Lee, N. Dangoumau et A. Toguyeni, *Supervisor Control and Petri nets for Scheduling Problem in FMS*, Joint Workshop on software Engineering Technology-KSEJW-2004, pp. 80 – 88, 26-27 Août 2004.

[LEE05a] E. J. Lee, N. Dangoumau et A. Toguyeni, *A Petri net based approach to design controllers for Flexible Manufacturing Systems*, In the Proceedings of 15th World IMACS, Paris, France, 11-15 Juillet 2005.

[LEE05b] E. J. Lee, A. Toguyeni et N. Dangoumau, *Petri net based Controllers for Forbidden Sequences of State-Transitions in the Control of Flexible Manufacturing Systems*, Conceptual Modeling and Simulation Conference - CMS 2005, Marseilles, France, 20-22 Octobre 2005.

[LEE06a] E. J. Lee, A. Toguyeni et N. Dangoumau, *A Petri Net based Decentralized Synthesis Approach for the Control of Flexible Manufacturing Systems*, CESA 2006 world congress (The International Association for Mathematics and Computers in Simulations), Beijing, China, 4-6 Octobre, 2006

[LEE06b] E. J. Lee, A. Toguyeni et N. Dangoumau, *A Petri net based approach for the Synthesis of Parts' Controllers for Reconfigurable Manufacturing Systems*, SICE-ICASE International Joint Conference, Busan, Korea, 18-21 Octobre, 2006

[LHO91] P. Lhoste, *Surveillance des M.S.A.P. : les Atouts de la Modélisation de Comportement*, Journée Surveillance de Pôle SED (GT2) du GR Automatique, Paris,

France, Février 1991.

[MAB96] M. Mabrouk, *Proposition d'une méthode et d'un outil d'aide à la Reconfiguration des Systèmes Automatisés de Production*, Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, Mai 1996.

[NIE94] E. Niel, *De la Sécurité Opérationnelle des Systèmes de Production*, Habilitation à Diriger des Recherches, Université de Lyon I, décembre 1994

[NIE96] E. Niel, N. Rezg, M. Nourelfath, S. Boukhobza, "Supervisory Control in the context of Operational Safety Reactivity", IMACS-IEEE SMC CESA'96, Lille, july 1996, pp. 746-751

[PAR92] T. Parayre, *Le MESAP : vers une Méthodologie d'Exploitation des Systèmes Automatisés de Production*, Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 1992.

[QUE00a] M. H. de Queiroz, J. E. R. Cury, *Modular Control of Composed Systems*, Proceedings of the American Control Conference, Chicago, Etats-Unis, 2000

[QUE00b] M. H. de Queiroz, J.E.R. Cury, *Modular Supervisory Control of Large Scale Discrete Event Systems*, Proceedings of the 5th International Workshop on Discrete Event Systems (WODES 2000), Gand, Belgique, 21-23 Août 2000

[QUE02] M. H. de Queiroz et J.E.R. Cury, *Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell*, Proceedings of the 6th International Workshop on Discrete Event Systems (WODES' 2002), pp. 377-382, Zargoza, Spain, Octobre 2002

[RAM87a] P. J. Ramadge et W. M. Wonham, *Supervisory control of a class of Discrete Event Processes,* SIAM J. Control and Optimization, Vol. 25, No. 1, Janvier 1987.

[RAM87b] P. J. Ramadge et W. M. Wonham, *Modular feedback logic for discrete-event systems,* SIAM J. Control and Optimization, Vol.25, No. 5, pp. 1202-1218, Septembre 1987.

[RAM87c] P. J. Ramadge et W. M. Wonham, *On the Supremal Controllable Sublanguage of a Given Language*, SIAM J. Control and Optimization, Vol. 25, no. 3, 1987.

[REV97] S. A. Reveliotis, M. A. Lawley, P. M. Ferreira, *Polynomial complexity deadlock avoidance policies for Sequential Resource Allocation systems*, IEEE Transaction on Automatic Control, Vol. 42, pp. 1344-1357, 1997

[REV01] S. A. Reveliotis, M. A. Lawley, P. M. Ferreira, *Structural Control of large-scale flexibility automated manufacturing systems*, dans The Design of Manufacturing Systems, édité par C.T. Leondes, CRC press, pp 4.1-4.34, 2001

[REV02] S.A. Reveliotis, *Liveness Enforcing Supervision for Sequential Resource Allocation Systems: state of the Art and Open Issues*, dans Synthesis and Control of Discrete Event Systems, Edité par B. Caillaud, P. Darondeau, L. Lavagno et X. Xie, Kluwer Academic Publishers, pp. 203-212, 2002

[SAH87] A. E. K. Sahraoui, *Contribution à la surveillance et à la commande d'ateliers*, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1987.

[SAH92] A. E. K. Sahraoui, *An approach for monitoring discrete event systems*, AFCET APII, Vol. 26, N°2, pp.91-106, 1992.

[SFA89] A. Sfalcin, M. Roesch, P. Lhoste et G. *Morel, Fiabilité, Disponibilité, et Sécurité des installations intégrant des filtres de comportement*, 2éme colloque annuel du club FIABEX, INT, Evry, Novembre, 1989.

[TOG95] A.K.A. Toguyéni, E.Craye et J.C. Gentina, *Knowledge-based supervision of flexible manufacturing systems*, Journal of Artificial Intelligence in Industrial Decision Making, Control and Automaton, pp. 631-662, 1995.

[TOG03] A.K.A. Toguyéni, P. Berruet et E.Craye, *Models and algorithms for failure diagnosis and recovery in FMS*, International Journal of Flexible Manufacturing Systems, Vol. 15, N°1, pp. 57-85, Janvier 2003.

[TOG05] A.K.A. Toguyéni et O. Korbaa, *Indirect monitoring of the failures of a flexible manufacturing system under cyclic scheduling*, ROBOTICS AND Computer6Integrated Manufacturing, Vol. 21, pp. 1-10, 2005.

[VAL94] C. Valentin, *Modeling and analysis methods for a class of hybrid dynamic systems*, ADPM'94, Brussels, Belgium, pp.221-226, 1994.

[WON88] W. M. Wonham et P. J. Ramadge, *Modular supervisory control of discrete event systems*, Mathematics of control of discrete event systems, 1(1):13-30, 1988.

[WON96a] K.C. Wong et W.M. Wonham, *Hierarchical Control of Discrete Event Systems, Discrete Event Dynamic Systems: Theory and Applications*, Vol. 6, N°3, pp. 241-273, Juillet 1996

[WON96b] K.C. Wong et W.M. Wonham, *Hierarchical Control of Discrete Event Systems, Discrete Event Dynamic Systems: Theory and Applications*, Vol. 6, N°3, pp. 275-306, Juillet 1996

[WON98] K. C. Wong et W. M. Wonham, *Modular Control and Coordination of Discrete-Event Systems*, In Discrete Event Dynamic Systems, 8(3):241-273, 1998.

[WON00] W.M. Wonham, *Supervisory Control of Discrete-Event Systems: An Introduction*, Proceeding IEEE International - Conference on Industrial Technology 2000 (ICIT2000), GOA, India, pp. 474-479, Janvier 2000.

[WON04] W.M. Wonham, *Notes on Control of Discrete-Event Systems: ECE 1636F/1637S*, 2004-2005.

[YAM96] K. Yamlidou, J. Moody, M. Lemmon et P. Antsaklis, *Feedback control of Petri nets based on place invariants*, vol.32, No.1, pp. 15-28, 1996

[ZAM97] E. Zamai, *Architecture de Surveillance-Commande pour les Systèmes à Evénements Discret Complexes*, Thèse de doctorat, Université Paul Sabatier de Toulouse, Septembre 1997.

Annexe A

1. Conception de l'outil

Dans le chapitre précédent, nous avons proposé le GASC par fusion synchrone de leur graphe d'accessibilité en précisant basant les deux principales règles à respecter pour la construction du modèle de synthèse. Cette fois-ci, toujours en se basant sur les règles précédentes, nous établissons un algorithme permettant la construction du GASC de RdP synchronisés par les transitions communes évoquées dans le chapitre précédent. Nous réalisons cet algorithme avec Microsoft Visual C^{++} 6.0. Il collecte à l'entrée le comportement souhaité et l'ensemble de séquences interdites de transitions d'état calculés par l'algorithme 1. Pour résoudre un système linéaire, il faut faire appel aux fonctions de la bibliothèque du logiciel de programmation mathématique ILOG CPLEX 9.0. Nous utilisons *PM Editeur* pour modéliser le procédé et la spécification en un modèle RdP et pour obtenir les informations permettant la construction GASC. La conception de notre travail est illustrée dans la Figure 73.

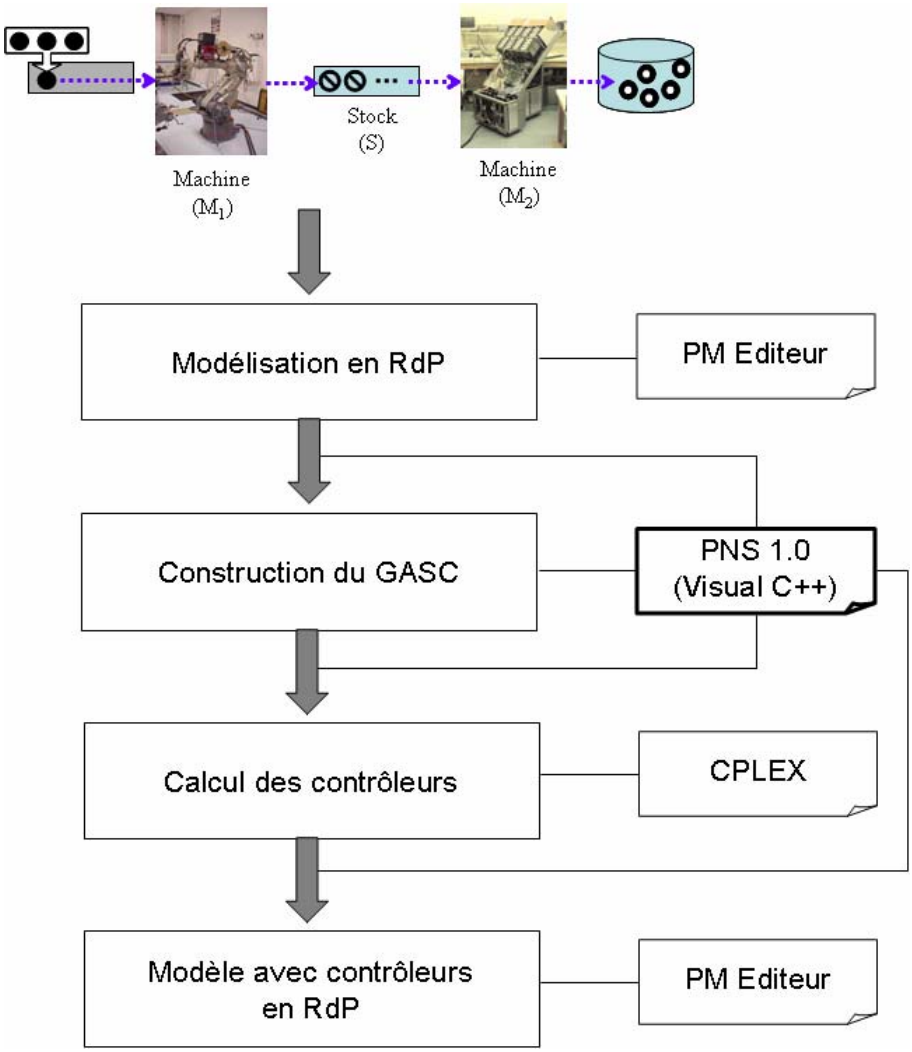
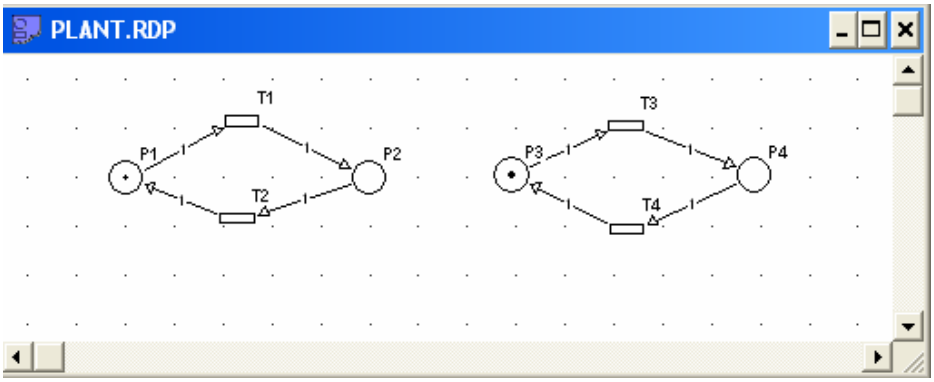


Figure 73. Explication d'un outil pour la synthèse

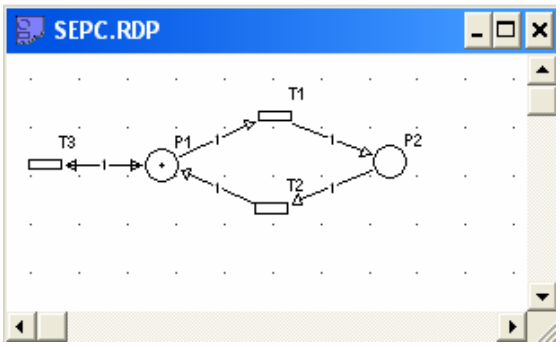
1.1 Modélisation en RdP en utilisant PM Éditeur 3.1

Nous pouvons dessiner le modèle du procédé et le modèle de la spécification par le *PM Editeur 3.1*. Ils sont enregistrés sous la forme de fichiers RDP (*.rdp). En général, quand on modélise en Réseaux de Petri, on nomme les places par 'P1, P2, …, Pn' et les transitions par 't1, t2, …, tm'. C'est pourquoi PM Editeur commence à numéroter les objet à partir de 1. De plus, le '*Signal*', information qui permet de distinguer les différentes places et transitions dans '*Attributs*', porte automatiquement le même chiffre que le numéro de la place ou de la transition considérée. C'est-à-dire, si une place est nommée 'P1', son '*Signal*' est noté '1'. Donc, nous devons bien indiquer et vérifier le numéro du '*Signal*' de la place et celui de la

transition quand nous dessinons les modèles. La Figure 74 est le modèle du procédé et le modèle de la spécification de l'exemple utilisé dans le chapitres II.



A. Modèle du procédé



B. Modèle de la spécification

Figure 74. Exemple du modèle du procédé et du modèle de la spécification utilisant *PM Editeur 3.1*

Dans la Figure 74, qui représente le modèle du procédé, nous pouvons utiliser sans rien changer les notations. Mais, dans le cas du modèle de la spécification, nous devons modifier les '*Signal*' des places et des transitions.

D'abord, nous changeons le '*Signal*' de la place. Un clic avec le bouton droit de la souris sur la place 'P1' permet d'afficher la boîte de dialogue intitulé '*Attributs Place*'. Comme le montre la Figure 75, nous changeons les '*Signal*' de la place 1 à '5' et nous écrivons 'P5' sur '*Desc.*' pour la renommer en P5. Donc, la place 'P1' est désormais considérée comme la place 'P5'.

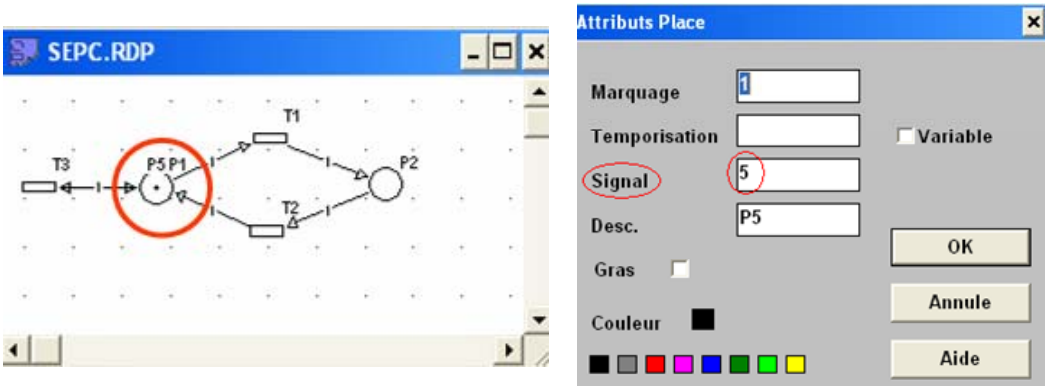


Figure 75. Changement de '*Signal*' de la place dans *PM Editeur*

Ensuite, nous changeons les '*Signal*' des transitions de la même manière que précédemment. La Figure 76 illustre le changement de la transitions 1 't3' en la transition 't1'. Finalement, nous avons le modèle de la spécification comme indiqué dans la Figure 77.

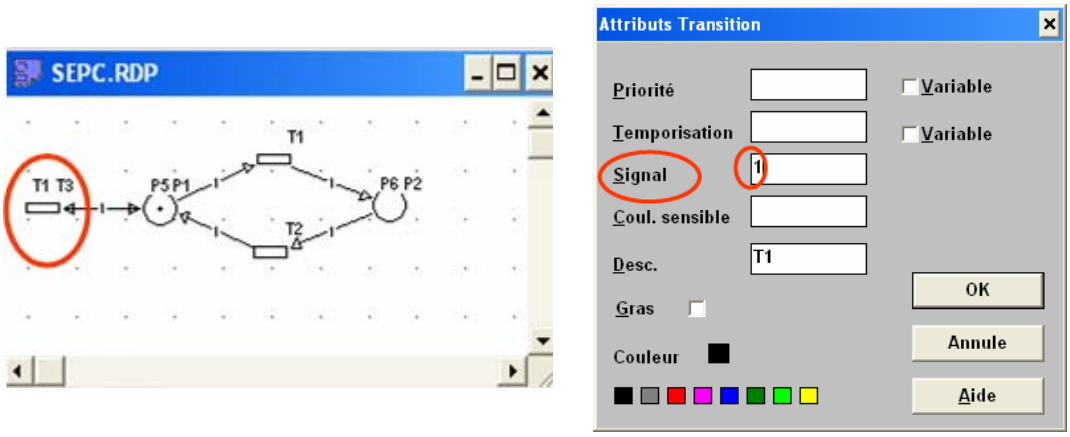


Figure 76. Changement de '*Signal*' de la transition dans *PM Editeur*

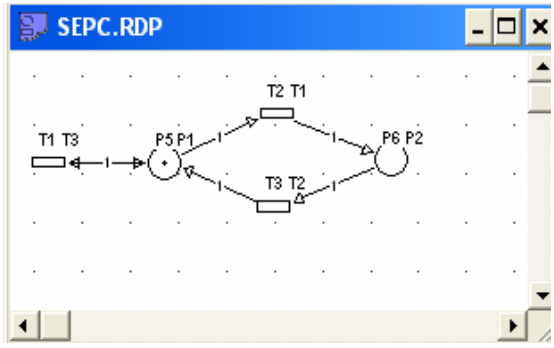


Figure 77. Modèle de la spécification réinscrit

1.2 Réalisation de l'algorithme de GASC

Nous construisons un programme nommé '*NetMDI*' en utilisant Visual C^{++} 6.0 pour réaliser l'algorithme du GASC. Celui-ci utilise le modèle du procédé et le modèle de la spécification construits par le *PM Editeur* par la méthode évoquée précédemment. Le '*NetMDI*' construit les séquences autorisées, les SITE et les cycles à partir des informations contenues dans ces deux modèles. Les résultats de cette opération sont fournis sous formes d'équations autorisées, d'équations interdites et d'équations cycliques permettant de calculer ultérieurement les places de contrôle à partir de '*CPLEX* '.

La Figure 78 représente l'écran d'ouverture de notre programme. On peut ouvrir le modèle du procédé et le modèle de la spécification en choisissant '*Open plant model*' et '*Open spec model*'.

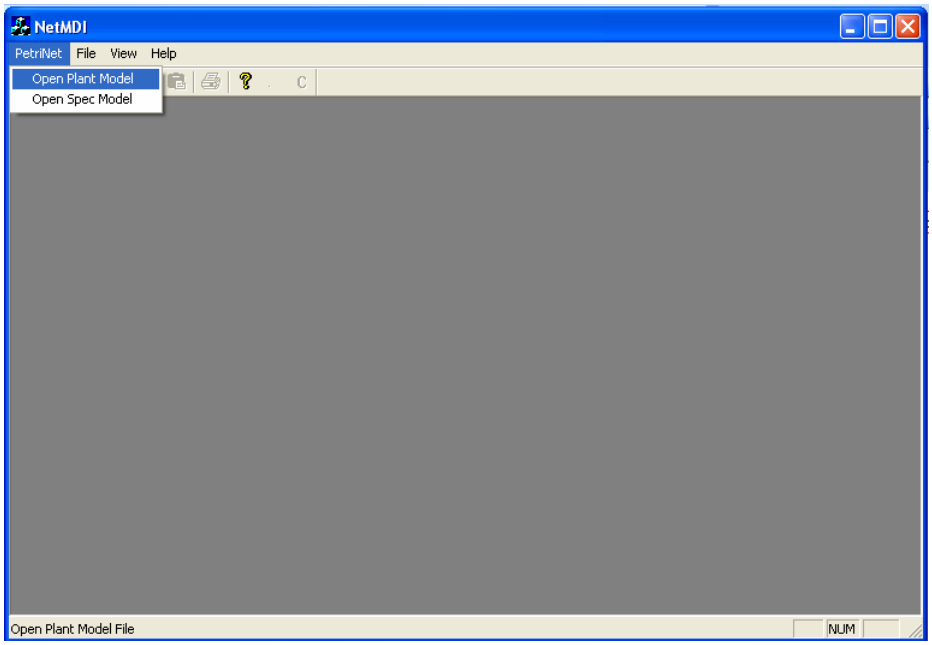


Figure 78. Écran d'ouverture de '*NetMDI*'

Lorsqu'on ouvre un fichier du modèle RdP, toutes les informations concernant les places, les transitions et les arcs du modèle du procédé viennent du fichier RDP (*.rdp). Les informations sont présentées comme le montre la Figure 79.

Après avoir ouvert un ou deux fichiers (Figure 79), on choisit dans le menu PetriNet '*Get Reachability Graph*' pour obtenir les informations sur les 'marquages pour obtenir les séquences autorisées' ('*Able Marking*') et les 'marquages pour définir les séquences interdites' ('*Forbidden Marking*').

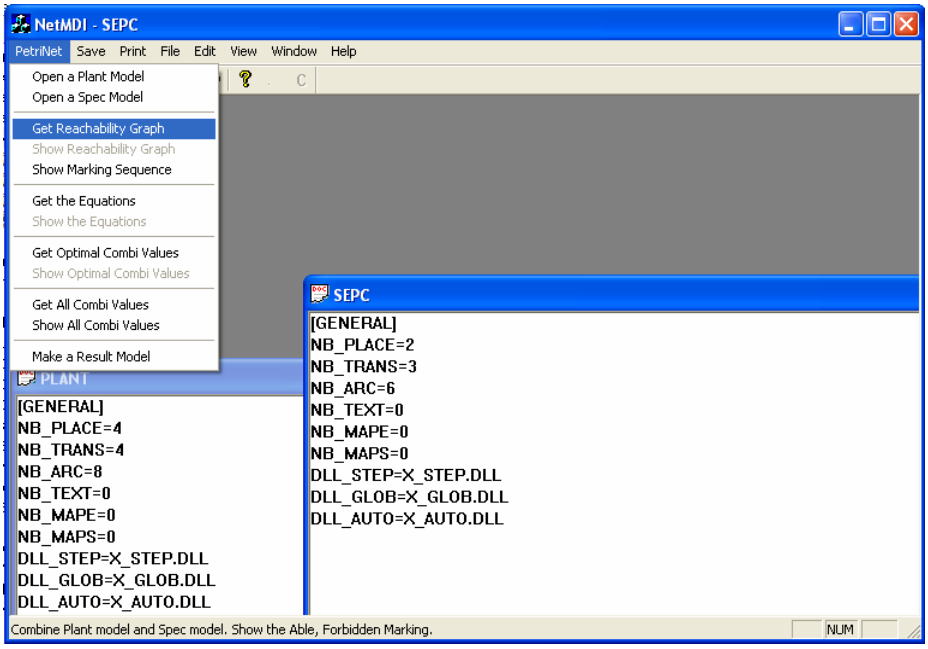


Figure 79. Ouverture du modèle RdP pour générer GASC

Illustrons maintenant de manière plus détaillée le processus de la construction du Graphe d'accessibilité synchrone contraint. Si nous cliquons sur '*Get Reachability Graph*' (Figure 79), notre programme exécute l'algorithme GASC (Figure 80) : à partir du marquage initial, il construit le marquage suivant en cherchant les transitions franchissables (c'est-à-dire autorisées). A ce moment-ci, les niveaux à chaque marquage sont générés. Le nivaux initial est numéroté par '0' et les suivants en ajoutant '1'. Nous désignons par '*AMi* (i = 0, …, n, n entier positif)' les marquages générés par les transitions autorisées, et par '*FMi* (i = 0, …, n, n entier positif)' les marquages générés par les transitions interdites en utilisant notre algorithme.

'*Upper T <= AM*' représente un marquage créé par franchissement dans le niveau antérieur. Par exemple, la transition franchissable du marquage 'AM4' est la transition 't2'. A cet instant-ci, notre programme construit le marquage 'AM5' et désigne par 't2<=AM4' le

'*Upper T <= AM*'. Dans le cas où un marquage généré par la transition franchissable est le même que celui déjà construit dans le niveau précédent, nous utilisons l'indicatif '*Same T <=AM*'. Par exemple, pour le marquage 'AM5' du niveau 5, le marquage généré par franchissement de la transition 't4'est le marquage 'AM2' du niveau 2. Nous dénommons par 't4<=AM5' le '*Same T <=AM*' du 'AM2'. Cette équation-ci sera utilisée ultérieurement pour calculer les équations de cycle.

Ensuite, '*Same FM*' signifie un marquage qui peut être franchissable mais qui est déjà indiqué comme un marquage interdit dans un autre niveau. Par exemple, le marquage 'AM4' généré par franchissement de la transition 't1' du marquage 'AM3' est le même que 'FM1' qui est classifié en tant que marquage interdit dans le niveau 2. C'est le problème des séquences interdites de transitions d'état défini dans le chapitre III. Ce qui signifie que, pour aller du marquage 'AM0' jusqu'au marquage '$[1\ 0\ 0\ 1\ 1\ 0\ 0]$ (FM1 = AM4)', la séquence '$t1*t3$' est interdite mais la séquence '$t1*t2*t3*t1$' est autorisée. Nous désignons par 'FM1' le '*Same FM*' du marquage 'AM4'.

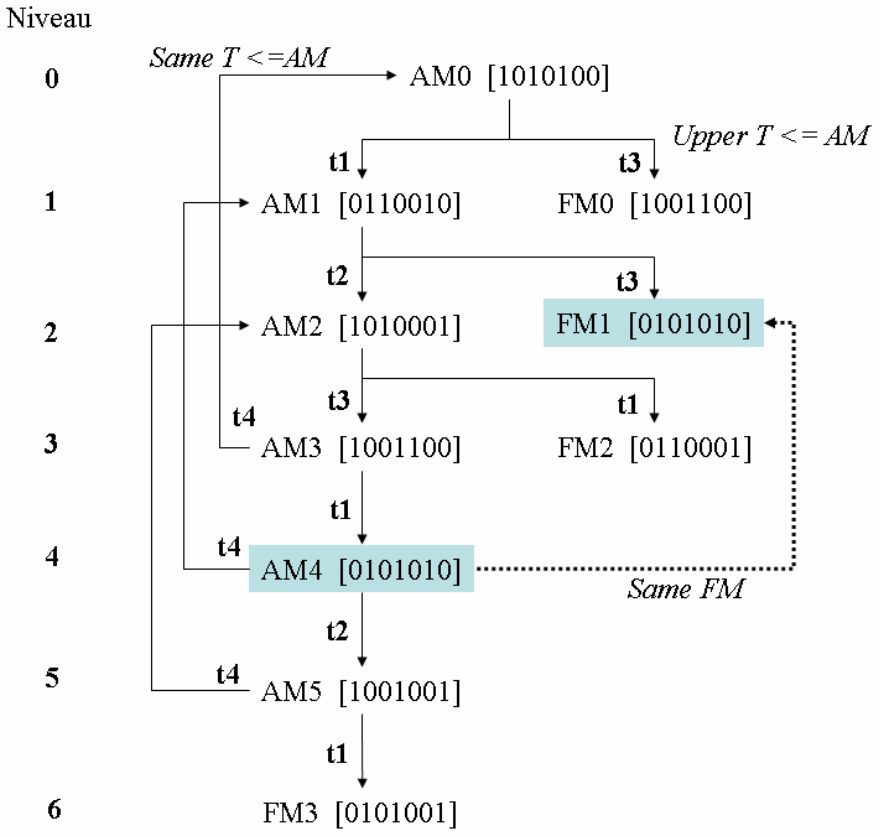


Figure 80. Graphe d'accessibilité synchrone contraint visuel

Notre programme nous propose le résultat par GASC d'algorithme comme l'indique la Figure 81. Celle-ci montre les marquages ('*Marking*'), le niveau du graphe d'accessibilité ('*Level*'), les transitions franchies ('*Upper T <= AM*' et '*Same T <=AM*', le premier est du niveau bas au niveau haut et le deuxième est du niveau haut au niveau bas.) et les marquages pour définir les séquences souhaitées et même interdites ('*Same FM*').

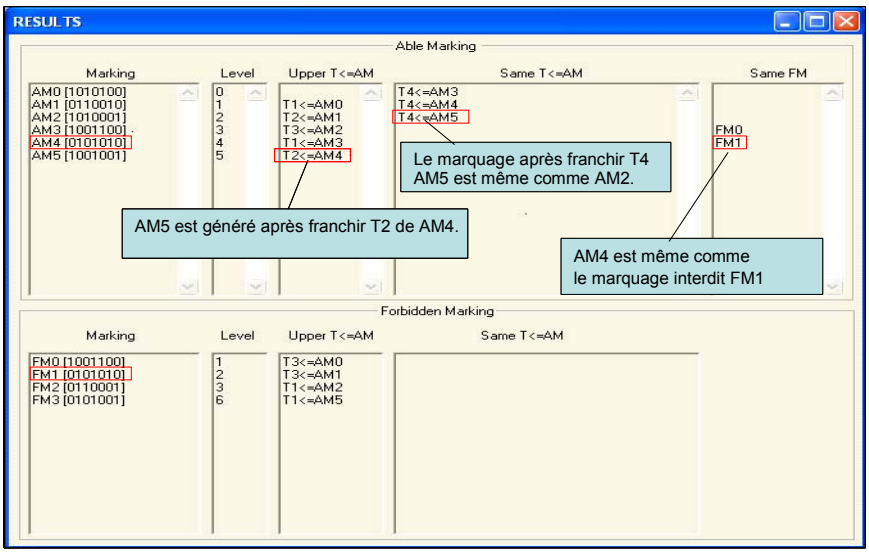


Figure 81. Informations sur le graphe d'accessibilité synchrone contraint

Si nous choisissons '*Show Marking Sequance*' dans le Menu de la Figure 79, nous obtenons le tableau de la Figure 82. Celui-ci permet de visualiser les résultats répartis en deux catégories : les séquences autorisées ('Able Marking Sequence') et les séquences interdites ('*Forbidden Marking Sequence*'). Dans cet exemple, notre programme nous offre cinq séquences autorisées : $t1$, $t1*t2$, $t1*t2*t3$, $t1*t2*t3*t1$, $t1*t2*t3*t1*t2$, et quatre séquences interdites : $t3$, $t1*t3$, $t1*t2*t1$, $t1*t2*t3*t1*t2*t1$.

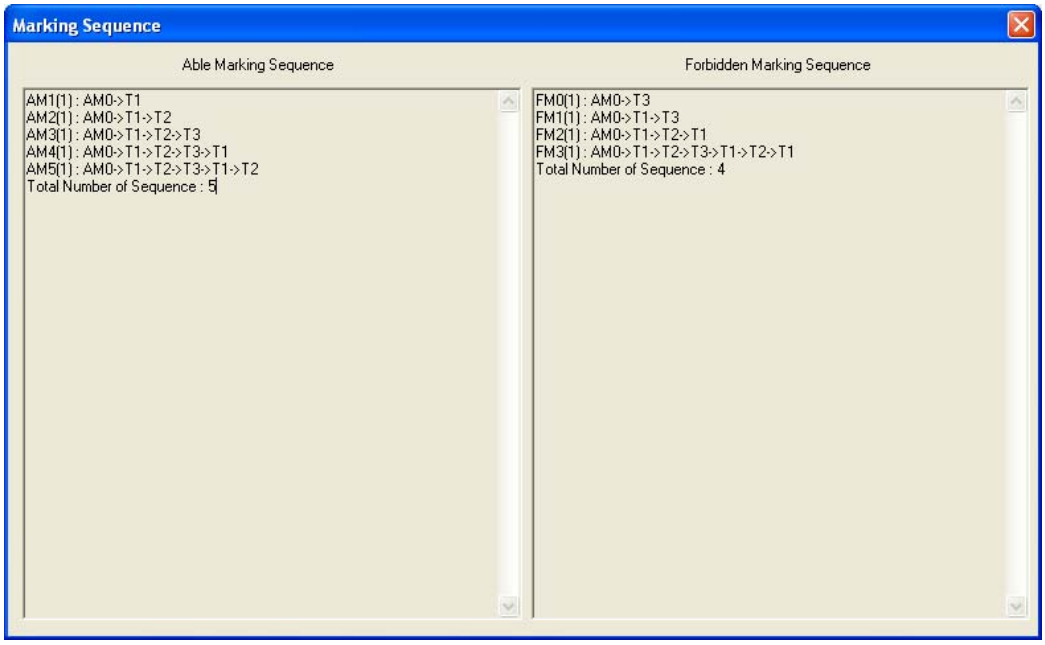


Figure 82. Séquences autorisée et séquences interdits

Ensuite, si on sélectionne '*Get the equation*' dans le menu du système *NetMDI*, le logiciel nous fournit les équations souhaitées, les équations interdites et les équations de cycle (Figure 83). Les premières ('*Able Equation*') sont à base de séquences autorisées et les secondes ('*Forbidden Equation*') à base de séquences interdites.

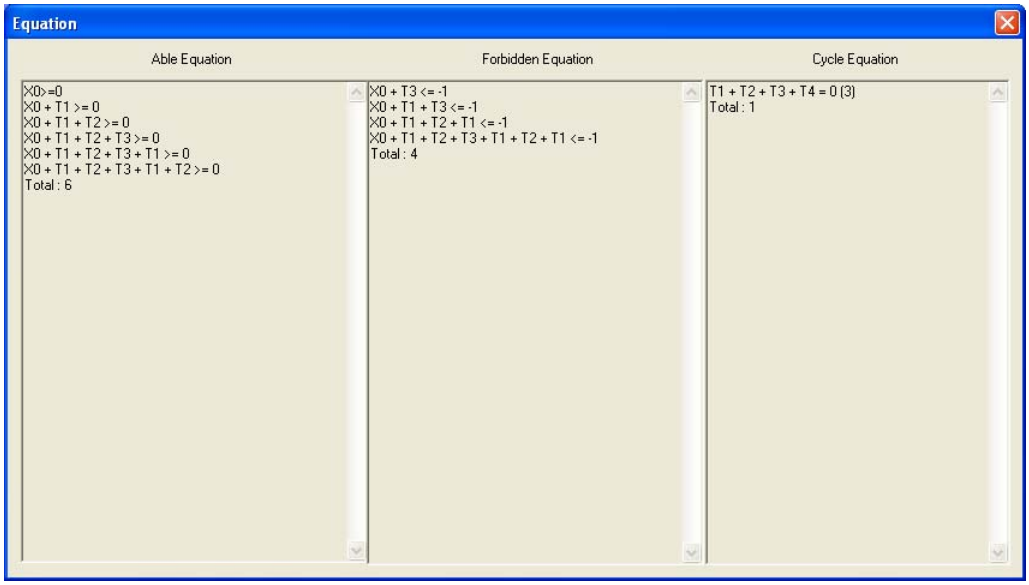


Figure 83. Obtention des équations : souhaitées, interdites et de cycle

Les séquences de transitions de tous les marquages pour définissant les séquences souhaitées doivent être vérifiées pour s'assurer que les séquences forment bien un cycle dans le processus. Pour construire les équations de cycle, nous utilisons les informations de '*Same T <=AM*' mentionnées précédemment. Dans notre exemple, nous avons trois '*Same T <=AM*' : AM0 (t4 <= AM3), AM1 (t4<=AM4) et AM2 (t4<=AM5). AM3 a une séquence ' AM3 <= t3 <= AM2 <= t2 <= AM1 <= t1 <= AM0' (Figure 82). D'après *Same T <=AM*', nous pouvons la représenter par 'AM0 <= t4 <= AM3 <= t3 <= AM2 <= t2 <= AM1 <= t1 <= AM0'. L'équation de cycle '$t1 + t2 + t3 + t4 = 0$' se construit par cette séquence.

En appliquant la même méthode aux AM1 (t4<=AM4) et AM2 (t4<=AM5), notre programme calcule les équations de cycle '$t2 + t3 + t1 + t4 = 0$' et '$t3 + t1 + t2 + t4 = 0$'. L'équation qui est de cycle est non orienté donc les trois équations de cycle sont représentées par une seule '$t1 + t2 + t3 + t4 = 0$'.

Les résultats montrés dans la Figure 83 sont les mêmes que les résultats des équations évoquées dans le Chapitre II.

Annexe B

2. Méthode d'Alpan

Algorithme I. Réduction du Modèle du Superviseur

Etape 1. *$\forall S_i$ et S_j (S_i, $S_j \in X$ et $i \neq j$) si $\exists (S_i, S_j)$ s.t.* $(\Psi_i^1 \cap \Psi_j^0 \neq \varnothing) \vee (\Psi_i^0 \cap \Psi_j^1 \neq \varnothing)$, *les états S_i et S_j sont distincts (i.e. non équivalents) et ne peuvent pas être combinés en un seul état (par la proposition 2)*

Etape 2. *Pour le couple (S_i, S_j) dont l'équivalence est prouvée par le pas 1, chercher l'ensemble $\Psi_{ij} = \Psi_i^1 \cap \Psi_j^1$. Si $\Psi_{ij} \neq \varnothing$, $\forall$ l'événement $x \in \Psi_{ij}$ cherche les états (S_r, S_s) sachant que [$\delta(S_i, x)$, $\delta(S_j, x)$] = (S_r, S_s) alors;*

> ***Etape 2.1***. *Si le couple (S_r, S_s) est noté distinct par le pas 1 de l'algorithme, les états S_i et S_j ne peuvent pas être équivalents ou combinés en un seul état.*
>
> ***Etape 2.2***. *Si le couple (S_r, S_s) est non noté distincte par le pas 1 de l'algorithme, alors ajouter le paire (S_i, S_j) à la liste des paires (S_r, S_s). Dans ce cas le paire (S_r, S_s) est trouvé enfin distincte, pour tous les paires de la liste des (S_r, S_s) qui sont notés distinctes.*

Définition 3: Les états S_i et S_j du contrôleur superviseur sont contrôlés équivalents si leurs modèles de contrôle (i.e. le contrôle de la sortie) ne sont pas contradictoires.

Définition 4: Un générateur S est « trim » si chaque état de ce générateur peut atteint à partir de tout état accessible à partir de l'état initial. Le génératrice « trim » possède des propriétés importantes telles que: vivacité, conservabilité, bornitude et la réversibilité pour tout marquage initial μ_0.

Titre - RECONFIGURATION DYNAMIQUE DE LA COMMANDE D'UN SYSTEME MANUFACTURIER : APPROCHE PAR LA SYNTHESE DE LA COMMANDE

Résumé – Ce travail est une contribution à la commande des systèmes manufacturiers et plus particulièrement à leur reconfiguration dynamique. En cas de reconfiguration, l'idée de base est de synthétiser les contrôleurs de commande en tenant compte d'une part des besoins de l'utilisateur et d'autre part de l'état du système. L'approche proposée est inspirée de la théorie du supervisory control, mais est adaptée à l'exploitation du formalisme des réseaux de Petri.

Dans ce travail, nous proposons un nouveau problème pour la modélisation des spécifications utilisateurs : le « Problème des Séquences Interdites de Transitions d'États» (PSITE). Nous proposons alors le « Graphe d'Accessibilité Synchrone Contraint » (GASC) pour synthétiser le modèle du procédé et le modèle des spécifications tous les deux basés sur les RdPs. Pour réaliser la synthèse, nous utilisons la théorie des régions revisitée par les travaux de thèse de A. Gaffari.

Dans la deuxième partie du travail nous montrons dans un premier temps que cette approche peut être utilisée pour la synthèse d'allocateurs de ressources, pour rendre déterministe l'interprétation de gammes opératoires. L'idée à ce niveau est de pouvoir synthétiser des contrôleurs déterministes qui mettent en œuvre un ordonnancement cyclique donné, établi en tenant compte de l'état du procédé et des objectifs de production. Ce travail est complété par la proposition d'un outil logiciel dont nous présentons le fonctionnement, l'intérêt et également les limites actuelles.

Mots Clés – Systèmes Flexibles de Production Manufacturière, Réseaux de Petri, Système à Événements Discrets, Graphe d'accessibilité synchrone contraint, Théorie des Régions, Problème des Séquences Interdites de Transitions d'Etats, Synthèse de la commande

2006 – Laboratoire d'Automatique, Génie Informatique et Signal

Title – THE DYNAMIC RECONFIGURATION CONTROL OF MANUFACTURING SYSTEM: APPROACH BY THE SYNTHESIS OF CONTROL

Summary – This work is a contribution to manufacturing systems control and more particularly to their dynamic reconfiguration. In the case of reconfiguration, the background idea is to synthesize controllers according with the user's request and the state of the system. In this case, we have developed a synthesis approach inspired of the supervisory control theory but adapted to the formalism of Petri Nets.

The first part of this work situates our contribution with respect to control synthesis based on PN formalism. The selection of PN is justified by their expression power with regard to the modeling of mechanisms such as parallelism and synchronization. Thereby, in our approach we address the different types of user specifications. We present a new problem: the "Forbidden Sequences of State-Transitions Problem" (FSSTP). We propose the "Constrained Synchronous Reachability Graph" (CSRG) to synthesize the plant model and the user specifications model based on PN. The theory is also based on the theory of regions as revisited in the thesis of A. Gaffari.

The second part of the work deals with the synthesis of resources allocators' to make determinist the interpretation of operating sequences. The idea is to synthesize deterministic controllers from a given cyclic scheduling established by taking into account the plant state and the production objective. This work is completed by the proposition of a software tool by which we present the main features and also the current limits.

Key words – Flexible Manufacturing System, Patri Net, Discrete Event System, Constrained Synchronous Reachability Graph, Theory of Region, Forbidden sequences of State-Transitions Problem, Supervisory Control Theory

2006 - Laboratoire d'Automatique, Génie Informatique et Signal