Université de NANCY I

UER de Mathématiques

M2C : UNE APPROCHE METHODIQUE POUR LA CONCEPTION CERTIFIEE DES SYSTEMES DE COMMANDE DES AUTOMATISMES INDUSTRIELS REPARTIS

==========

THESE

Présentée à l'Université de Nancy I
pour l'obtention du grade de

DOCTEUR D'ETAT ES-SCIENCES MATHEMATIQUES
(Mention : Informatique)

Par

Joachim TANKOANO

==========

Thèse soutenue publiquement le 8 Mars 1988 devant
la commission d'examen composée de

Président	M.	**J.P. THOMESSE**, ENSEM NANCY
Rapporteurs	MM.	**M. COURVOISIER** Université P. Sabatier de TOULOUSE
		P. LADET, Ecole des Mines de ST ETIENNE
		J.M. PROTH INRIA NANCY
Examinateurs	MM.	**J.C. DERNIAME** Université de NANCY I
		C. IUNG, ENSEM NANCY
		G.R. PERRIN, Université de BESANCON
		M. RAYNAL, Université de RENNES

Université de NANCY I UER de Mathématiques

M2C : UNE APPROCHE METHODIQUE POUR LA CONCEPTION CERTIFIEE DES SYSTEMES DE COMMANDE DES AUTOMATISMES INDUSTRIELS REPARTIS

==========

THESE

Présentée à l'Université de Nancy I
pour l'obtention du grade de

DOCTEUR D'ETAT ES-SCIENCES MATHEMATIQUES
(Mention : Informatique)

Par

Joachim TANKOANO

==========

Thèse soutenue publiquement le 8 Mars 1988 devant
la commission d'examen composée de

Président	M.	**J.P. THOMESSE**, ENSEM NANCY
Rapporteurs	MM.	**M. COURVOISIER** Université P. Sabatier de TOULOUSE
		P. LADET, Ecole des Mines de ST ETIENNE
		J.M. PROTH INRIA NANCY
Examinateurs	MM.	**J.C. DERNIAME** Université de NANCY I
		C. IUNG, ENSEM NANCY
		G.R. PERRIN, Université de BESANCON
		M. RAYNAL, Université de RENNES

A la mémoire de ma mère

A Jacqueline

A Joël et Olivier

Plus qu'une tradition, je voudrais ici exprimer ma reconnaissance à tous ceux du laboratoire du CRIN ou de l'extérieur qui m'ont aidé par leurs conseils, par leurs discussions, et aussi par leur amitié.

Je tiens tout particulièrement à exprimer ma profonde gratitude à Mr. J.C. Derniame, Professeur à l'Université de Nancy I, pour la confiance qu'il m'a témoignée en m'acceptant dans l'équipe de recherche qu'il anime, pour l'orientation qu'il a donnée à mon travail, et aussi pour ses multiples encouragements.

Je remercie vivement :

Mr. J.P. Thomesse, Professeur à l'Ecole Nationale Supérieure d'Electricité et de Mécanique de Nancy, qui me fait l'honneur de présider ce jury.

Mr. M. Courvoisier, Professeur à l'Université Paul Sabatier de Toulouse, qui a accepté d'être rapporteur de ma thèse. Sa longue expérience sur l'utilisation des réseaux de Petri dans la conception des systèmes de commande le désignait tout naturellement pour évaluer mon travail, et sa lecture critique a contribué à son amélioration.

Mr. P. Ladet, Professeur à l'Ecole des Mines de St Etienne. Egalement spécialiste de l'utilisation des réseaux de Petri dans la conception des systèmes de commande, il a accepté d'être rapporteur de ma thèse. Les discussions que nous avons eues, et ses critiques avisées ont contribué à clarifier mes idées.

Mr. J.M. Proth, Directeur de l'INRIA à Nancy. Il a aussi accepté d'être rapporteur de ma thèse. Sa connaissance des problèmes d'optimisation dans les ateliers de production le désignait également pour cette évaluation.

Je remercie vivement les personnalités scientifiques qui se sont intéressées à mon travail, et qui me font l'honneur de participer à ce jury, malgré leurs lourdes tâches:

Mr. C. Iung Professeur à l'Ecole Nationale Supérieure d'Electricité et de Mécanique de Nancy

Mr. G.R. Perrin , Professeur à l'Université de Besançon

Mr. M. Raynal Professeur à l'Université de Rennes.

Je remercie également D. Boudebous avec qui j'ai fait équipe ces dernières années, pour sa collaboration qui a sans aucun doute contribuée à l'avancement de mon travail.

Je tiens également à associer à ces remerciements, tous les collègues de l'IUT de l'Université de Nancy II. La qualité de nos rapports m'ont permis de concilier mes activités d'enseignement et de recherche.

Je ne saurais terminer sans remercier Mme D. Marchand, qui a assuré avec le plus grand soin, la dactylographie d'une partie de cette thèse.

PLAN

CHAPITRE 1

INTRODUCTION

PLAN

1.0 LES IDEES CENTRALES

Un système de commande de procédé industriel discontinu est un système chargé de surveiller et de piloter des processus de production environnants assujéttis à des évolutions parallèles et à des contraintes temporelles strictes [Le Lann 83]. Les ateliers flexibles, caractérisés par un nombre important de machines de production (40 000 chez General Motors) [Minet 87], géographiquement distribués, et qui ne sont pas indépendants les uns des autres, conduit de plus en plus à concevoir ces systèmes comme des systèmes répartis, plus faciles à maintenir, à reconfigurer, à étendre, et offrant un plus haut degré de fiabilité et de disponibilité [Thomesse 80] [Ladet 82] [Kramer 83] [Calvez 84] [Pascal 87] [Le Lann 87] [Lonchamp 87].

La contrepartie de ces avantages est l'extrême complexité de conception de ces systèmes, caractérisée par quatre tâches principales :

- La **spécification** du service attendu
- L'**organisation** du système
- Sa **validation**
- Et l'**évaluation** de ses performances.

De ces quatres tâches, la validation est sans conteste la plus importante, mais aussi la plus complexe et la plus délicate [Wirth 77]. Ceci est dû d'une part à l'explosion combinatoire du nombre des états d'un système, et d'autre part à la difficulté, voire à l'impossibilité matérielle d'observer globalement ces états [Lamport 78] [Deschizeaux 82]. De ce fait, les méthodes de validation qui sont en général proposées dans la littérature sont soit trop lourdes dans leur mise en oeuvre, soit limitées dans le type de propriétés qu'elles permettent de vérifier.

Les réseaux de Petri (RdPs) [Peterson 81] [Brams 83] [TSI 85] qui constituent sans aucun doute l'un des modèles d'analyse du parallélisme pour

lequel il existe le plus important corpus de résultats applicables, n'échappent pas à cette règle. Bon nombre de propriétés importantes, comme celles liées à la vivacité, qui garantissent à des degrés divers la possibilité pour un système de prolonger indéfiniment des séquences d'opérations, ne sont pas toujours décidables. Les résultats intéressants qui ont pu être dégagés, ne sont en général valables que pour des classes particulières de réseaux, ou applicables que dans des contextes bien particuliers, **ce qui les rend très souvent difficilement exploitables**. C'est le cas par exemple du théorème de Commoner [Commoner 72], qui donne pour certaines classes une condition de vivacité fondée sur la structure du réseau. On peut aussi citer de nombreux résultats existants sur certains types de transformation qui permettent de préserver les bonnes propriétés d'un réseau, dont la vivacité. C'est le cas par exemple des règles de raffinement fondées sur la notion de bloc bien formé [Valette 76], des règles de fusion et de substitution fondées sur la notion d'abstraction [André 81], et des règles de réduction qui permettent de réduire ou d'augmenter la taille d'un réseau [Brams 83] [Berthelot 85 a].

Dans ce mémoire, nous exploitons ces résultats à des degrés divers, pour proposer une approche méthodique de conception rigoureuse des systèmes de commande répartis, qui s'appuie sur

- Une **démarche globale**
- Quatre **modèles de représentation**
- Et deux **outils** (PETRI-C et PETRI-S).

Les caractéristiques essentielles de cette approche résident dans deux points principaux qui justifient son intérêt et font son originalité.

Le premier point concerne les **règles de construction et de transformation** mécanisables que nous proposons pour à la fois **guider et discipliner le raisonnement** dans les tâches de spécification et d'organisation. Ces règles s'appuient sur des techniques de **raffinement** et de

décomposition de réseaux de Petri interprétés, qui permettent de **simplifier**, voire d'**éliminer les étapes délicates de validation**, et de rendre ainsi la construire de systèmes complexes et sûrs de fonctionnement abordable par un concepteur moyen .

Le second point concerne la définition d'une approche de **modélisation de l'environnement commandé et du flux d'activité**, qui a permis la réalisation d'un outil d'aide au **prototypage rapide** [Boudebous 88], basé sur une simulation conjointe de la partie commande et de la partie opérative. Ce simulateur peut être utilisé soit comme un **outil de mise au point**, soit pour **dialoguer avec l'utilisateur** afin de clarifier ses besoins, soit pour évaluer les performances et **dimmensionner l'atelier**. Il est caractérisé par sa très grande facilité de mise en oeuvre (ce qui est un point fondamental pour le prototypage), et aussi et surtout par le fait qu'il permet d'éviter les doubles définitions (avec tous les risques d'incohérence que cela comporte) en exploitant au mieux la complémentarité qui existe entre les différentes vues caractéristiques du fonctionnement d'un atelier.

Ces deux points sont essentiels pour aborder de façon cohérente et réaliste les tâches intimement liées de spécification, d'organisation, de validation, et d'évaluation des performances d'un système de commande.

Avant de présenter le plan de l'ouvrage, nous précisons le cadre général dans lequel ce travail s'est déroulé, les objectifs poursuivis, les concepts généraux qui ont inspiré la démarche globale de conception que nous avons retenue, et les grandes lignes de cette démarche.

1.1 GENESE

Ce travail de recherche qui s'est déroulé au sein de l'équipe "Génie Logiciel" du CRIN, s'inscrit dans le cadre d'un projet de recherche plus vaste portant à la fois sur les aspects

- de **conception**

- de **programmation**

- et d'**implémentation**

des applications industrielles réparties. Ce projet qui a débuté dans le contexte de l'ATP sur le parallélisme a conduit autour des années 83 à la définition du langage FLEXI [Derniame 83] [Zakari 84] autour duquel le prototype d'un **système de programmation** permettant de développer des applications réparties sur un réseau de processeurs communicant via un réseau local a été réalisé [Thiel 85]. Le but de notre travail a été de compléter ce système de programmation par un **système d'aide à la conception**, l'objectif à terme étant de converger vers un environnement de développement ciblé sur une classe particulière d'applications, les applications temps réel réparties.

De ce fait, le langage FLEXI a constitué le point de départ de toutes nos réflexions. Il a largement influencé notre démarche globale de conception, et fournit tous les mécanismes nécessaires pour décrire, tout au long de cette démarche , les aspects structurels d'un système de commande.

L'un des éléments déterminants dans la méthode de conception d'un système complexe, réside en effet dans le choix du modèle de représentation de sa structure. Dans la littérature, différents langages sont proposés pour décrire de façon statique ou dynamique la structure d'un système réparti, soit comme un ensemble d'entités communicantes mono ou multi-tâches qui constituent les unités de répartition [Kahn 77] [Hoare 78] [Hansen 78] [Lister 80] [Raynal 81] [Kramer 81] [Kramer 82] [Andrews 82] [Courtiat 87] [Herrmann 87], soit comme un ensemble d'entités circulantes [Betourné 85]. Plus proche des particularités liées à la commande des procédés industriels, on peut également citer les approches développées dans [Thomesse 80] et [Ladet 82] pour décrire les applications réparties en séparant structure de contrôle et traitement.

FLEXI fait partie de cette famille de langages. Les concepts qui le caractérisent sont proches de ceux développés également dans [Raynal 81] et plus particulièrement dans [Lister 80] [Kramer 81] et [Kramer 82]. Ils

constituent un apport important des notions d'**abstraction** et de **modularité** à la programmation d'applications réparties.

Les modules et tâches

Dans FLEXI, les modules constituent des unités de répartition, c'est à dire des unités locales qui ne peuvent être réparties sur plusieurs sites. Définis au sens des modules de Kramer [Lister 80] [Kramer 81] [Kramer 82], ces modules ne peuvent communiquer entre eux que par des échanges de messages, et peuvent être constitués de une ou plusieurs tâches parallèles. L'intérêt de ce concept réside dans le fait qu'il permet de faire abstraction de la répartition lors de la conception de la structure interne d'un système, et de réaliser ainsi une programmation indépendante de l'architecture du réseau de processeurs cible.

Les modules constituent également des unités de réalisation. Chaque module peut, en fonction de sa destination, être programmé et testé séparément dans un langage spécifique, en utilisant un noyau commun de langage proposé dans FLEXI. Ceci permet en particulier de développer des applications réparties sur des réseaux de processeurs hétérogènes.

Les ports et politiques de communication

Les points d'entrée et de sortie des messages dans un module sont représentés par des ports qui définissent l'interface de communication de ce module avec son environnement. Cet interface se définit comme suit:

Module < nom de module > interface is
in < nom de port > input = < type de message >
out < nom de port > output = < type de message >
inout < nom de port > input = < type de message > output = <type de message>
outin < nom de port > output = < type de message > input = <type de message>
.....
.....

Plusieurs "types de communication" autorisant la production et la consommation de messages sur les ports selon différentes stratégies, permettent de bien séparer, lors de la programmation, les problèmes de gestion de la communication, des problèmes algorithmiques.

Les liaisons entre modules

A l'intérieur d'un module, les partenaires sont désignés de façon implicite dans les primitives de communication par l'intermédiaire des ports. Ce sont les liaisons définies comme suit entre ces ports, à l'extérieur des modules, qui explicitent en dernier ressort les partenaires d'un échange:

Connect < nom de module >/< nom de port > on < nom de module >/< nom de port >
< nom de module >/<nom de port >
.....

On rend ainsi la programmation des modules indépendante de la topologie du réseau logique d'échange, ce qui permet de les assembler de différentes manières en fonction du service à assurer.

Répartition et reconfiguration du système

Enfin, FLEXI permet de définir la répartition d'un système, en décrivant l'affectation des modules sur des sites physiques, indépendamment du contenu des modules et des liaisons définies entre ces modules, ce qui a pour conséquence de permettre une reconfiguration statique rapide. Cette affectation se définit simplement de la façon suivante:

Machine < nom de processeur > : < nom de module >

1.2 LES GRANDES LIGNES DE LA DEMARCHE GLOBALE DE CONCEPTION

La démarche globale de conception qui nous servira de fil directeur tout au long de ce mémoire découle de ces différents concepts. Elle se situe en aval d'une **démarche productique** [AFNOR 85] ayant permis de concevoir le système de production selon une approche globale. Nous supposerons donc acquis un certain nombre de résultats :

- Spécification des produits
- Spécification des gammes (ou procédés) de fabrication
- Spécification et organisation globale des moyens de production (postes de travail, moyens de transport et de stockage, organisation générale du flux d'information).

Elle vise la conception de systèmes de commande dont les composants sont à répartir de façon statique sur un réseau de processeurs (ordinateurs, micro-ordinateurs ou automates programmables) dotés d'un exécutif temps réel et d'un système de communication par échanges de messages adéquats.

Elle est caractérisée essentiellement par deux étapes dans le cycle de décision et par trois niveaux dans le cycle d'abstraction (c.f. figure 2.1), ce qui nous rapproche des démarches généralement préconisées pour la conception des systèmes d'information [Tardieu 86].

1.2.1 LE CYCLE DE DECISION

Il s'articule autour des deux étapes suivantes :

(1) L'étape de conception globale

Les choix de solution opérés à ce niveau concernent l'architecture globale du système.

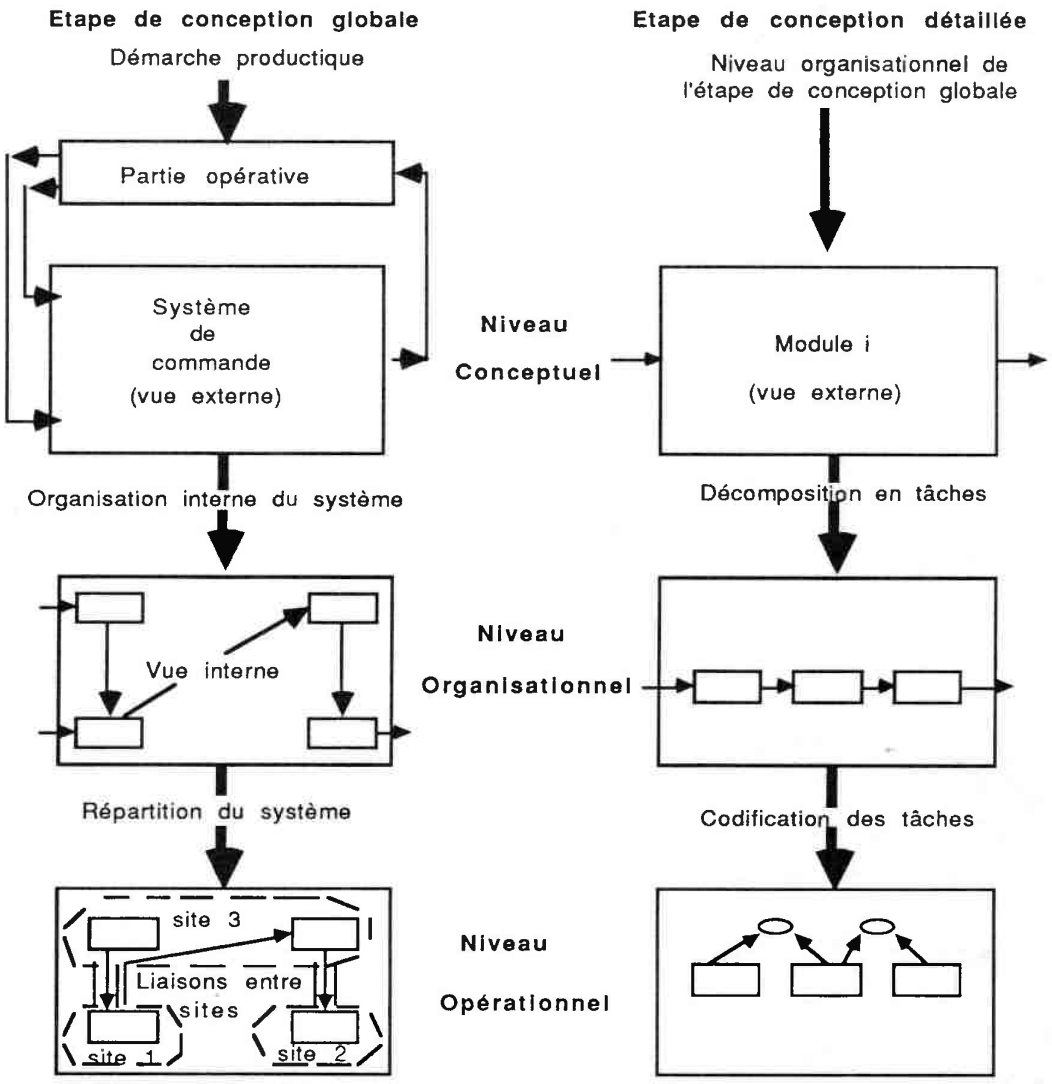


Figure 2.1

(2) L'étape de conception détaillée

A ce niveau on s'intéresse à la conception détaillée des sous-systèmes identifiés à l'étape précédente.

1.2.2 LE CYCLE D'ABSTRACTION

Pour chaque étape du cycle de décision, nous distinguons les trois niveaux d'abstraction suivants, qui permettent de construire progressivement un système de commande réparti en faisant le plus longtemps possible abstraction des choix d'organisation, des choix de répartition, et des contraintes de réalisation dans un langage de programmation donné.

(1) Le niveau conceptuel

Pour répondre à des contraintes de productivité, l'organisation de la production issue de la démarche productique impose très souvent un fonctionnement parallèle et parfois concurrent des outils de production à automatiser. Ce fonctionnement qui décrit un parallélisme de situation [Raynal 81] constitue dans la plupart des cas l'essentiel de l'énoncé du problème à résoudre. Au niveau conceptuel, il s'agit essentiellement pour le concepteur de dialoguer avec l'utilisateur, pour décrire le comportement externe du système de commande, en définissant la relation d'ordre qui doit être maintenue entre les entrées et les sorties de ce système, pour contraindre le procédé commandé à évoluer en respectant le fonctionnement voulu.

(2) Le niveau organisationnel

Il s'appuie sur les résultats acquis au niveau conceptuel. Au stade de la conception globale, il s'agit pour le concepteur de déterminer l'architecture interne qui facilitera le mieux ou qui rendra possible la répartition du système de commande sur un réseau de processeurs. Cette tâche conduit pour l'essentiel à regrouper dans des unités de répartition en fonction de différentes contraintes [

Le Lann 83]

- Contraintes de temps

- Contraintes de répartition géographique des outils de production

- Contraintes de flexibilité dans la production

- Contraintes de robustesse

- Contraintes de modularité [El Fazziki 85]

les entrées et les sorties qu'il est important de ne pas dissocier, et à définir le comportement interne du système de commande.

Au stade de la conception détaillée, il s'agit de décomposer chaque sous-système en tâches parallèles, soit pour des raisons d'efficacité, soit pour des raisons de clarté.

(3) Le niveau opérationnel

Le niveau opérationnel est consacré aux choix de répartition sur des sites physiques des unités de répartition identifiées au niveau organisationel, (à l'étape de conception globale), et aux choix de réalisation du comportement spécifié pour chaque tâche dans un langage de programmation donné, (à l'étape de conception détaillée).

1.2.3 LES AVANTAGES DE LA DEMARCHE

La hiérarchisation des préoccupations proposée dans cette démarche présente plusieurs avantages:

(1) Elle simplifie la complexité de conception, en séparant spécification du problème, organisation du système, et prise en compte des contraintes des

langages de programmation et de l'architecture physique du réseau de processeurs . Plus particulièrement, au niveau conceptuel, en retardant la prise en compte des contraintes d'organisation, elle facilite la construction de la spécification du problème par le concepteur, et l'analyse de cette spécification par l'utilisateur, tous les deux pouvant dans leur raisonnement présupposer l'existence d'une échelle temporelle universelle.

(2) Par ailleurs, les décisions sont prises dans un ordre qui facilitera la maintenance ultérieure du système. Les décisions qui évoluent le plus vite sont prises le plus tard possible, ce qui permet de prendre en compte leurs évolutions sans remettre en cause les décisions prises plus tôt [Dijkstra 76]. Ainsi, les choix de répartition du système peuvent être modifiés lorsque les caractéristiques du réseau physique des sites changent sans que ne soit remis en cause l'architecture interne du système. De la même façon, l'architecture interne du système peut être modifiée lorsque les contraintes d'organisation changent sans que ne soit remis en cause le comportement externe du système.

Tout en étant globalement descendante, cette démarche ne s'oppose en aucune façon à une approche ascendante, à des étapes particulières de la conception. Les modèles de représentation et les outils qui la supportent autorisent ces deux approches.

1.3 LE PLAN DE L'OUVRAGE

Le plan que nous adoptons comporte deux parties. La première partie est consacrée à la description des modèles de représentation que nous avons retenu pour supporter notre démarche de conception. Nous y abordons successivement la présentation du modèle de description du comportement d'un système de commande (chapitre 2), la modélisation de son environnement (chapitre 3), et la modélisation du flux d'activité (chapitre 4). La modélisation de l'environnement et du flux d'activité complètent le modèle de description du comportement dans une optique de simulation.

La deuxième partie est consacrée à la présentation des outils d'aide à la

conception. L'outil d'aide au prototypage rapide présenté de façon détaillée dans [Boudebous 88] ne sera que effleuré. Dans le chapitre 5 qui sert d'introduction à cette deuxième partie, nous effectuons tout d'abord un survol des différentes formes que prennent de façon générale les outils d'aide à la conception. Ensuite, nous abordons la présentation des techniques de raffinement (chapitre 6) et de décomposition (chapitre 7) que nous proposons comme techniques d'aide à la conception.

PREMIERE

PARTIE

Chapitre 2

LE MODELE DE DESCRIPTION DU COMPORTEMENT

PLAN

2.0 INTRODUCTION

Le choix d'un modèle de spécification du comportement que doit avoir un système pour assurer le service qu'on attend de lui, semble faire partie des problèmes sujets à polémiques, tant il est intimement lié à la culture scientifique des individus, au domaine d'application, et aussi à la destination des spécifications qu'on peut en dériver. Dans ce chapitre, après une étude critique des aspects fondamentaux qui différencient les différents modèles de spécification, nous aborderons une présentation détaillée d'un modèle dérivé des réseaux de Petri Interprétés (RdPI), sur lequel nous nous appuyons pour spécifier le comportement d'un système à répartir, aux différentes étapes de notre démarche de conception.

2.1 LES ASPECTS CARACTERISTIQUES DES MODELES DE SPECIFICATION

Définitions:

Soient

-S *un système représenté par un module ou par un réseau de modules communicants*

-et E = { c1, c2, ...cn} *l'ensemble des événements de communication possibles sur les ports de* S.

On appelle ***trace*** *de* S *tout mot sur le vocabulaire* E, *construit en concaténant dans leur ordre d'occurrence, les événements de communication observés pendant une exécution de* S. *L'ensemble de toutes les traces que l'on peut ainsi construire pour un système défini son* ***comportement***. *On parle de* ***comportement externe*** *lorsque ce système est perçu comme un module unique, et de* ***comportement interne*** *lorsqu'il s'agit plutôt d'un réseau de modules communicants.*

Spécifier le comportement d'un système, c'est définir d'une manière ou d'une autre l'ensemble de ses traces possibles. De façon plus intuitive, il s'agit essentiellement, pour chaque module d'un système, de préciser comment celui-ci doit interagir avec son environnement, en faisant abstraction de son implémentation.

L'un des aspects caractéristiques des modèles de spécification que nous abordons en tout premier lieu est le type de relation qu'ils permettent de définir entre les événements constitutifs d'une trace. Cette relation peut être perçue de deux points de vue.

Le premier point de vue est **fonctionnel**. Il conduit à définir les valeurs à produire comme des fonctions des valeurs consommées, soit de façon procédurale, soit de façon statique à l'aide de pré et de post conditions.

Le second point de vue concerne les **synchronisations**. Il permet de définir les contraintes d'entrelacement des différents événements de communication, et de préciser ainsi les valeurs à prendre parmi plusieurs valeurs consommées pour calculer une valeur à produire.

Tout en étant complémentaires, chacun de ces points de vue peut devenir primordial en fonction de la classe du problème traité. Ainsi, le point de vue fonctionnel est caractéristique des **systèmes dits transformationnels** (ex: la construction d'algorithmes paralléles). Dans ce cas, le service attendu est essentiellement défini par la relation fonctionnelle que le système doit maintenir entre ses entrées et ses sorties. Quant au point de vue des synchronisations, il est caractéristique des **système dits réactifs** (ex: la commande des procédés industriels) qui doivent plutôt maintenir une relation d'ordre entre leurs entrées et leurs sorties pour assurer le service attendu.

Un autre aspect caractéristique des modèles de spécification est le type de raisonnement qu'ils induisent sur le comportement d'un système.

La spécification d'un système peut être vue soit comme un **générateur**, soit comme un **accepteur** de traces. Elle induit alors un raisonnement sur le comportement futur, en supposant une connaissance de l'état courant du système.

On peut aussi voir la spécification d'un système comme un moyen pour **vérifier à posteriori** qu'une trace donnée est bien une trace du système considéré. Dans ce cas, la spécification induit un raisonnement sur le passé du système.

De par son caractère plus constructif, le premier point de vue permet une conception progressive du parallélisme dans le comportement d'un sysrème, et facilite le passage d'une spécification à une réalisation.

Par ailleurs, le concepteur peut raisonner sur l'état courant d'un système, soit à l'aide de **variables d'état**, soit de façon duale à l'aide de **séquences d'événements** observés sur les ports du système. Dans le premier cas, les variables d'état servent à résumer un passé, le calcul de ce passé devant être explicité par ailleurs. Dans le second cas, le passé d'un système est décrit de façon plus explicite à l'aide d'un calcul. Les deux approches étant duales, il est en général possible, pour le spécifieur, de passer assez simplement dans certains modèles, de l'une à l'autre, en fonction des spécificités du problème à résoudre. Toutefois, on peut noter que la première approche offre un meilleur moyen d'abstraction, la définition des calculs pouvant être différée dans le temps par rapport à la définition des variables d'état.

Un autre aspect caractéristique des modèles de spécification que nous abordons concerne la formalisation des spécifications.

Elle peut être faite soit de façon **statique** à l'aide d'assertions, soit de façon **dynamique** à l'aide de programmes abstraits. La première approche est plus proche du raisonnement humain. Les énoncés sont en général orthogonaux et formulés de façon incrémentale. Toutefois, face à un très grand nombre

d'énoncés, il devient très difficile de faire le lien entre les différentes contraintes exprimées. La deuxième approche a contre elle d'amener le concepteur à définir l'énoncé d'un problème de façon algorithmique, ce qui peut éventuellement conduire à des surspécifications. En contrepartie, la possibilité de représenter l'état d'un système, soit par des variables d'état, soit par la structure de contrôle d'un programme abstrait, peut faciliter dans une certaine mesure la compréhension des liens existants entre les différentes contraintes.

Enfin, le dernier mais non le moindre des aspects caractéristiques des modèles de spécification que nous abordons ici, concerne le fait qu'un modèle peut ou non autoriser une **conception certifiée,** en permettant d'une part de vérifier formellement que le comportement décrit possède de **bonnes propriétés**, et d'autre part de garantir la **cohérence** entre une spécification et la manière dont elle est implémentée. Cet aspect est fondamental pour la conception de certaines applications, comme les applications de commande de procédés industriels, qui nécessitent une sureté de fonctionnement très élevée. Bien souvent, cet aspect justifie à lui tout seul l'intérêt d'un modèle.

2.2 QUELQUES EXEMPLES BIBLIOGRAPHIQUES

2.2.1. LE MODELE DE J. MISRA ET K. M. CHANDY

Dans [Misra 81], Misra et Chandy proposent de décrire le comportement externe d'un système H par une formule générale du type

R | H | S

où R et S sont des assertions sur les traces de H. Cette formule signifie que, pour toute trace de H, si S est initialement vrai, et si de plus R est vrai jusqu'au rang k de la trace, alors S sera vrai jusqu'au rang k+1 (pour tout $k \geq 0$).

Lorsque H est composé d'un réseau de modules communicants, on spécifie

le comportement de chaque module h_i par une formule de même type

$$r_i \mid h_i \mid s_i.$$

L'ensemble de ces formules définit le comportement interne de H.

Les conventions suivantes sont utilisées dans les assertions:

- si C désigne un canal, ZC désigne la trace sur C
- si Z_1 et Z_2 sont des traces, $Z_1 \ll Z_2$ exprime que Z_1 est préfixe de Z_2
- si Z_1 et Z_2 sont des traces, $Z_1 \equiv Z_2$ exprime que Z_1 et Z_2 sont identiques
- *empty* désigne la trace vide

On peut illustrer ce modèle de spécification sur l'exemple simple suivant: Soit "merge" un système recevant sur ses canaux d'entrée "in_1" et "in_2" deux listes de valeurs triées en ordre croissant, et produisant sur son canal de sortie "out" la fusion de ces deux listes. Le comportement de ce système peut être spécifié de la façon suivante:

$$mi(Zin_1),\ mi(Zin_2) \mid merge \mid mi(Zout),\ Zout \subseteq Zin_1 \cup Zin_2$$

Dans cette formule, mi(Z) sert à désigner que la trace Z est monotone croissante, et $Zout \subseteq Zin_1 \cup Zin_2$ que les valeurs apparaîssant dans la trace Zout apparaîssent aussi, soit dans Zin_1, soit dans Zin_2. On exprime ainsi deux choses: d'une part les contraintes de classement que doivent respecter les deux listes en entrée et la liste en sortie, et d'autre part la relation qui doit exister entre les valeurs consommées et les valeurs produites.

Des règles d'inférence permettent de vérifier formellement la cohérence entre la spécification externe et la spécification interne d'un système.

Commentaires: Ce modèle permet de décrire le comportement passé d'un système de façon statique. Toutefois, le fait de raisonner en termes de traces conduit à recourir à des fonctions spécifiques du type de "mi", ou à introduire une très grande sérialisation dans le comportement d'un système. En fait, ce modèle est trés lourd pour décrire un système réactif caractérisé par des contraintes d'entrelacement très complexes.

2.2.2. LE MODELE DE B.S. CHEN ET R.T. YEH

Dans ce modèle [Chen 83], on décrit le comportement d'un système en définissant des contraintes sur l'entrelacement des événements, dans un formalisme utilisant la logique du premier ordre étendue à l'opérateur de précédence temporelle (noté $\mapsto$) et à l'opérateur de causalité (noté $\Rightarrow$). Si dans un système l'événement e_1 se produit avant l'événement e_2, on note $e_1 \mapsto e_2$ (ce qui se lit e_1 précède e_2). Si la cause de l'occurrence d'un événement e_2 est l'occurrence d'un autre événement e_1, on note $e_1 \Rightarrow e_2$ (ce qui se lit que l'événement e_2 est causé par l'événement e_1). Deux événements liés par la relation de causalité sont, bien entendu, également lié par la relation de précédence temporelle. Ces deux relations sont transitives, mais ne sont ni reflexives, ni symétriques.

Ce qui suit est un exemple de spécification d'un protocole de communication dans le formalisme de Chen et Yeh. Dans cet exemple, "A" désigne la trace sur le canal d'entrée du système de transmission, et "B" la trace sur son canal de sortie. Le symbole $\nRightarrow$ est utilisé pour désigner l'implication, et le symbole $\equiv$ l'équivalence. Si "a" est un événement, "a.msg" désigne le message associé à l'événement "a".

(1) $\forall \quad a \in A, \exists\ b \in B$ tel que $a \Rightarrow b$

(2) $\forall$ $b \in B$, $\exists$ $a \in A$ tel que $a \Rightarrow b$

(3) $\forall$ $a \in A$ $b_1, b_2 \in B$

$a \Rightarrow b_1 \wedge a \Rightarrow b_2 \nRightarrow b_1 \equiv b_2$

(4) $\forall$ $a_1, a_2 \in A$ $b_1, b_2 \in B$

$a_1 \Rightarrow b_1 \wedge a_2 \Rightarrow b_2 \nRightarrow (a_1 \mapsto a_2 \wedge b_1 \mapsto b_2) \vee (a_1 \equiv a_2 \wedge b_1 \equiv b_2) \vee (a_2 \mapsto a_1 \wedge b_2 \mapsto b_1)$

(5) $\forall$ $a \in A$ $b \in B$

$a \Rightarrow b \nRightarrow b.msg = a.msg.$

L'énoncé (1) exprime le fait qu'aucun message ne doit se perdre, l'énoncé (2) le fait qu'aucun message ne doit être rajouté par le système de transmission, l'énoncé (3) le fait qu'aucun message ne doit être dupliqué, l'énoncé (4) le fait que l'ordre d'émission doit être préservé à la reception, et l'énoncé (5) le fait que le contenu des messages doit être transmis sans erreur.

Tout comme dans le modèle de Misra et Chandy, il est possible ici également de vérifier la cohérence entre spécification externe et spécification interne. En plus, il est possible de démontrer comme un théorème que le comportement d'un système sous les contraintes spécifiées, vérifie bien certaines propriétés.

Commentaires : Le modèle de Chen et Yeh permet de décrire le comportement d'un système de façon statique en raisonnant sur le passé en termes d'événements. Par rapport au modèle précédent, il présente plusieurs avantages:

1) Les spécifications des contraintes d'entrelacement peuvent se faire au niveau le plus fin, parce que les assertions portent sur des événements et non sur des traces.

2) En plus, comme le montre très bien l'exemple présenté, ce que le système est autorisé à faire ou à ne pas faire peut être véritablement énoncé de façon orthogonale .

Toutefois, le fait que la notion d'état du système ne peut être décrit qu'en caractérisant un ordre d'événements, et le fait qu'il n'existe pas de moyen direct d'exprimer un ordre total entre événements, peuvent conduire à des spécifications très lourdes. Le modèle de Misra et Chandy et le modèle de Chen et Yeh sont caractéristiques d'un ensemble de propositions qui se fondent sur la logique moderne (logique du premier ordre, logique temporelle etc...) [Ramamritham 83] [Chandy 86].

2.2.3. LE MODELE DE J. R. ABRIAL ET S. A. SCHUMAN

Dans ce modèle [Abrial 79], on définit un système par

- S l'ensemble de ses états possibles
- S_b le sous ensemble de S représentant l'ensemble des états de départ
- S_f le sous ensemble de S représentant l'ensemble des états d'arrivée
- I l' ensemble de ses événements tel que à chaque événement i est associé une fonction e_i dont le domaine est S_i , sous-ensemble de S, et le codomaine S. La fonction e_i définit les changements d'états possibles, chaque fois que l'événement i se produit.

Un système de communication pour le transfert des entiers positifs $\mathbb{N}$ selon le modèle producteur-consommateur via un tampon borné peut être défini de la façon suivante:

$S = \mathbb{N} \times \{0,1,.....,n\} \times \mathbb{N}$ (n désigne la taille du tampon. L'état du système est représenté par un triplet : le nombre de valeurs restant à transmettre, l'état du tampon, et le nombre de valeurs reçues)

$I = \{p,c\}$ (p désigne l'événement de production et c l'événement de consommation)

$S_b = \mathbb{N} \times \{0\} \times \{0\}$

$S_f = \{0\} \times \{0\} \times \mathbb{N}$

$S_p = \mathbb{N}+ \times \{0,.....,n-1\} \times \mathbb{N}$

$S_c = \mathbb{N} \times \{1,.....,n\} \times \mathbb{N}$

$e_p(in,buf,out) = (in - 1, buf + 1, out)$

$e_c(in,buf,out) = (in, buf - 1, out + 1).$

La figure 2.1 représente par un graphe l'ensemble des évolutions possibles du système pour trois valeurs à transmettre. Dans ce graphe, chaque noeud correspond à un état du système, caractérisé par le nombre de valeurs non encore transmises, le nombre de valeurs dans le tampon, et le nombre de valeurs déjà reçues. Les arcs sont étiquetés par les événements à l'origine des changements d'état.

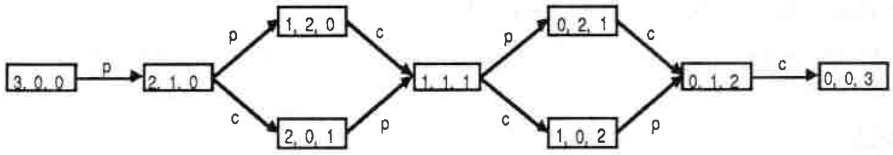


Figure 2.1

Un langage de type ensembliste permet de décrire de façon très lisible un système. En plus, dans ce modèle, il est possible de vérifier certaines propriétés du comportement décrit, et la cohérence d'une spécification avec son implémentation.

Commentaires : Le modèle d'Abrial et Schuman permet de décrire le comportement d'un système de façon statique comme un accepteur de traces, en raisonnant sur le futur en termes d'événements et de changements d'états. Alors que le modèle de Chen et Yeh encourage le spécifieur à aborder de façon

directe le comportement d'un système tel qu'il doit être, ce modèle favorise plutôt une approche par des raffinements successifs introduisant de plus en plus de parallélisme dans le comportement. Cette différence d'approche trouve sa justification essentiellement dans le fait que le raisonnement sur le futur est plus constructif que le raisonnement sur le passé. Le modèle d'Abrial et Schuman est assez caractéristique d'une famille de propositions ([Raynal 81] [Perrin 85]) proches du modèle de système de transitions de KELLER [Keller 77].

2.2.4. LE MODELE DE Z.C. CHEN ET C.A.R. HOARE

Dans [Chen 81], Z.C. Chen et C.A.R. Hoare proposent un formalisme d'expression du comportement d'un système, et un formalisme pour énoncer les propriétés de ce comportement.

Le langage d'expression du comportement est très proche du formalisme de Backus utilisé pour les grammaires. Il n'autorise ni variables locales, ni instructions d'affectation. Les itérations sont exprimées à l'aide de la récursivité droite, et les choix non déterministes à l'aide de l'opérateur '|'. La notation pour les opérations de communication est la même que celle proposée dans CSP [Hoare 78]. Dans une opération de communication, le partenaire de l'échange est désigné par l'intermédiaire d'un canal bidirectionnel. Dans ce formalisme, on note le comportement d'un processus sous la forme générale

p $\triangleq$ P où p désigne le nom du processus et P l'expression de son comportement.

Par exemple, les énoncés suivants spécifient un protocole de communication:

(1) sender $\triangleq$ (input?y:M $\mapsto$ q[y])
(2) q[x:M] $\triangleq$ (wire!x $\mapsto$ (wire?y:{ACK} $\mapsto$ sender | wire?y:{NACK} $\mapsto$ q[x]))
(3) receiver $\triangleq$ (wire?z:M $\mapsto$ (wire!ACK $\mapsto$ output!z $\mapsto$ receiver
| wire!NACK $\mapsto$ receiver))

Dans l'énoncé (1), sender désigne le processus émetteur. Il lit sur le canal input un message de type M. La valeur lue est placée dans la variable y. Le processus se comporte ensuite comme le processus q[y], q désignant un tableau de processus.

L'énoncé (2) définit le tableau de processus q, et le comportement de chaque processus référencé dans ce tableau. La dimension du tableau est déterminée par le type M. Un processus donné du tableau commence par émettre son rang via le canal wire. Ensuite, il reçoit via le même canal, soit le message ACK, soit le message NACK. Dans le premier cas, il poursuit son comportement comme le processus sender. Dans le second cas, il recommence l'émission.

Dans l'énoncé (3), receiver désigne le processus récepteur. Il commence par lire un message de type M sur le canal wire. Ensuite, il émet sur le même canal, soit le message ACK, soit le message NACK. Dans le premier cas, il poursuit son comportement en émettant sur le canal output le message reçu, puis en se remettant en attente pour un nouveau message. Dans le second cas, il se remet tout de suite en attente pour un nouveau message.

Le système global est défini par l'énoncé suivant:

protocol Δ (chan wire : (sender // receiver))

On exprime ainsi qu'il est constitué des deux processus sender et receiver. wire est désigné à l'aide du mot clé chan comme étant le seul canal d'interconnexion de ces deux processus. Les autres canaux, input et output référencés dans les processus du système, définissent ses points d'interaction avec l'environnement. La figure 2.2 présente graphiquement l'architecture du système spécifié.

Les énoncés de propriétés s'expriment sous la forme générale

P sat R

où P désigne un processus ou un réseau de processus, et R une assertion sur le comportement de P. Cette notation signifie que R doit être vrai avant et après chaque opération de communication de P. Elle permet ainsi d'exprimer des propriétés de correction partielle. Les assertions sont formulées à l'aide de la logique du premier ordre et de quelques opérateurs sur les traces:

$s \leq t$	indique que la trace sur le canal s est préfixe de la trace sur le canal t
$x \wedge s$	indique que le message x est préfixe de la trace sur le canal s
$\# s$	désigne la longueur de la trace sur le canal s
s_i	désigne le message de rang i sur le canal s

L'énoncé qui suit est un exemple d'énoncé de propriété sur le comportement décrit plus haut:

protocole sat input ≤ output

Il exprime le fait que la séquence des messages délivrés doit être préfixe de la séquence des messages émis. Des règles d'inférence permettent de prouver que le comportement d'un système sous les contraintes spécifiées respecte bien les propriétés exprimées.

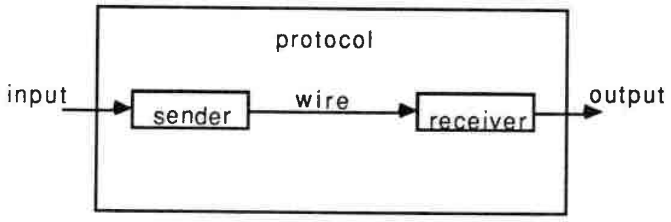


Figure 2.2

Commentaires: Le modèle de Chen et Hoare conduit le concepteur à spécifier le comportement d'un système à l'aide d'un programme abstrait agissant comme un générateur de traces. Il induit un raisonnement sur le comportement futur

en fonction de l'état courant du système. Cet état courant est entièrement défini de façon implicite dans la structure de contrôle du programme abstrait, ce qui encourage une très grande sérialisation. Cette sérialisation peut devenir nuisible dans la spécification initiale des systèmes dits réactifs. En contrepartie, les relations fonctionnelles pouvant s'exprimer assez aisément à l'aide de la logique du premier ordre, on peut dire que ce modèle est bien adapté à la spécification du comportement des systèmes dits transformationnels. Il est caractéristique d'une famille de propositions autour des expressions de chemins et des grammaires [Campbell 74] [Milner 80] [Hughes 83] [Zave 85].

2.3. LE CHOIX DU MODELE

Comme le montre bien l'étude bibliographique que nous venons de présenter, il n'existe pas de solution universelle aux problèmes que posent la spécification du comportement d'un système. Tout au plus, une proposition peut cibler au mieux une classe de problèmes à résoudre. De ce fait, dans une démarche de conception, la définition d'un modèle de spécification est plus que fondamentale. Elle circonscrit le type de problèmes que l'on peut espérer résoudre, et le type d'aide que l'on peut apporter au concepteur.

Dans ce travail, notre objectif n'est toutefois pas orienté vers une telle préoccupation. Introduit depuis 1962 par le professeur Petri, les résultats accumulés autour des réseaux de Petri (RdP) au fil du temps (c.f. [BRAMS 83] [TSI 85] [Peterson 81] ...), en font aujourd'hui l'un des modèles le mieux connu et le plus adapté à la spécification du comportement des systèmes réactifs. **Il permet d'exprimer les contraintes de temps, et est adapté à la fois à la validation et à la simulation**. Notre objectif est plus de nous appuyer sur cet acquis considérable pour aborder d'autres problèmes non moins fondamentaux liés à la conception des applications de commande réparties, tel celui de la définition d'une démarche globale de conception, et celui de la définition des outils qui supportent cette démarche.

L'annexe A est consacré à une présentation synthétique des RdP. Cet

annexe est destiné au lecteur non familier du formalisme. Nous y abordons successivement la présentation du modèle de base, les types de propriétés qu'il permet de vérifier, les abréviations préconisées, et les extensions apportées pour accroître son pouvoir d'expression. **Les rappels traités sont indispensables pour aborder la suite de l'exposé.**

Dans ce qui suit, nous présentons de façon plus détaillée le modèle que nous avons retenu. Ce modèle s'appuie sur les RdP dits interprétés. Dans cette présentation, nous étudions trois points principaux. Le premier point concerne une généralisation des règles permettant de synchroniser l'évolution d'un réseau aux interactions qu'un système doit avoir avec son environnement. Nous nous appuyons pour cela sur la notion de type de communication. Le second point concerne une redéfinition des règles d'arbitrage pour le tir des transitions franchissables. Comme nous le verrons, cette redéfinition a été rendue nécessaire par le fait que nous nous situons dans un contexte réparti. Enfin, le troisième point concerne l'extension du modèle à l'expression des communications inter-modulaires

2.4 SPECIFICATION DU COMPORTEMENT EXTERNE A L'AIDE DES RdP INTERPRETES

On décrit le comportement externe d'un système de commande à l'aide de RdP interprétés (RdPI) [Moalla 78] [Moalla 85] en synchronisant l'évolution d'un RdP autonome (RdP généralisé, ou RdP coloré) aux interactions que le système doit avoir avec son environnement. Dans cette spécification, le RdP autonome sert à mémoriser le passé du système et de son environnement.

2.4.1 DEFINITION

Plus précisément, un RdPI R_I spécifiant le comportement externe d'un système de commande est défini par un RdP autonome R , un triplet <V; OP; PR> caractérisant les interactions du système avec son environnement, et deux

fonctions Φ et Ψ servant à établir un lien entre ces interactions et l'évolution du réseau. Dans cette définition

-V = E + S + C + H désigne l'union de quatre ensembles disjoints de variables où

- . E représente l'ensemble des **variables d'entrée** du système dont les valeurs successives sont engendrées par l'environnement
- . S l'ensemble des **variables de sortie** du système dont les valeurs successives déterminent la commande appliquée à l'environnement
- . C l'ensemble des **variables internes** au système
- . et H un ensemble de **variables de temporisation** servant à initialiser la génération de top d'horloge au bout d'intervalles de temps prédéfinis

-OP = $\{op_1, op_2, \ldots, op_n\}$ désigne un ensemble d'opérations où chaque opération op_i est une application

$$op_i : E + C + H \longmapsto S + C$$

-PR=$\{pr_1, pr_2, \ldots, pr_m\}$ désigne un ensemble de prédicats définis sur E + C + H

-Φ : P $\longmapsto$ OP associe à chaque place p une **opération**

-Ψ : T $\longmapsto$ PR associe à chaque transition t un prédicat ou **réceptivité**

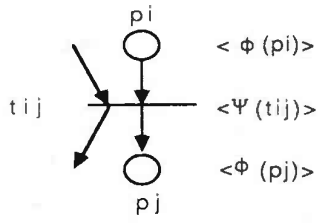


Figure 2.3 - Représentation graphique d'un RdPI

2.4.2 QUELQUES NOTIONS ET CONVENTIONS SUPPLEMENTAIRES

Nous précisons la définition d'un RdPI par quelques notions et conventions supplémentaires dont la plupart sont empruntées au GRAFCET [Blanchard 79] [GREPA 85].

a) La notion d'événement et de changement d'état

Dans la formulation des réceptivités associées aux transitions, on s'intéresse soit à un état particulier de l'environnement (caractérisé par une condition logique sur les variables d'entrée), soit à l'instant correspondant au changement de cet état. Pour des raisons de simplification dans la spécification, on a coutume d'associer à tout état d'un environnement caractérisé par une condition logique "cond" deux types d'événements:

- l'événement noté

  ↑ cond (c'est le front montant de "cond")

qui exprime le passage de faux à vrai de la condition "cond" et correspond à l'instant où l'environnement entre dans l'état considéré,

- et l'événement noté

  ↓ cond (c'est le front descendant de "cond")

qui exprime le passage de vrai à faux de la condition "cond" et correspond à l'instant où l'environnement sort de l'état considéré.

Une réceptivité est alors notée sous la forme générale

<e.c>

où e désigne un événement de changement d'état, et c une condition caractéristique d'un état de l'environnement. Ainsi, dans la figure 2.4 les réseaux (a) et (b) ont la même signification.

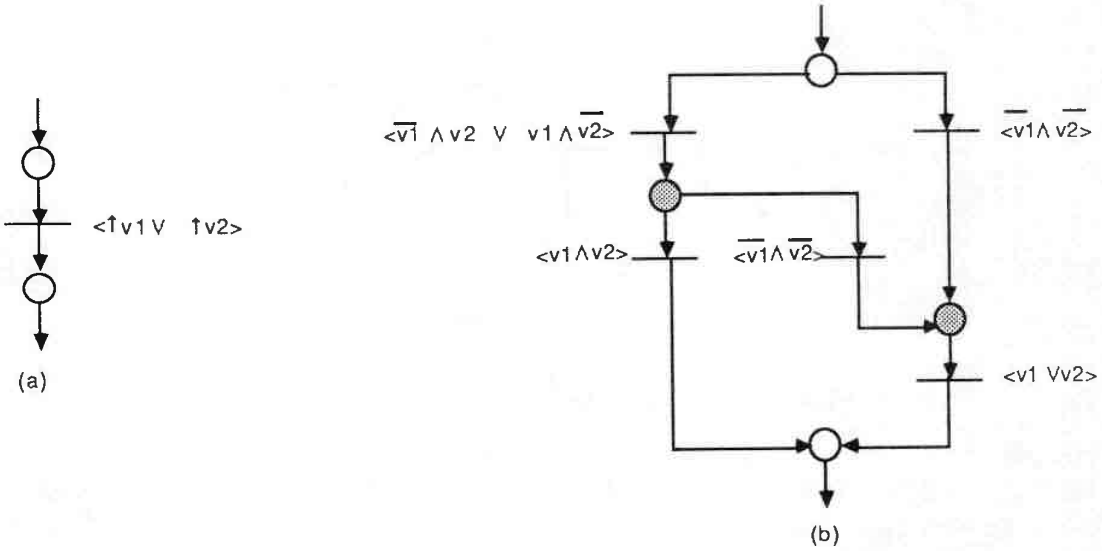


Figure 2.4

b) La prise en compte du temps

Dans la spécification d'un système de commande, certaines contraintes s'expriment en fonction d'un temps écoulé. Deux opérations prédéfinies permettent l'expression de ces contraintes à l'aide des variables de temporisation:

Armer (vtemp,n)	Force la variable de temporisation vtemp à vrai au bout de n unités de temps
Desarmer (vtemp)	Désactive l'effet de Armer (vtemp,n)

Dans la figure 2.5, on exprime ainsi qu'après l'exécution de l'opération a, la transition t_1 sera franchie si l'événement ↑ v survient, sinon la transition t_2 sera franchie au bout de 10 unités de temps.

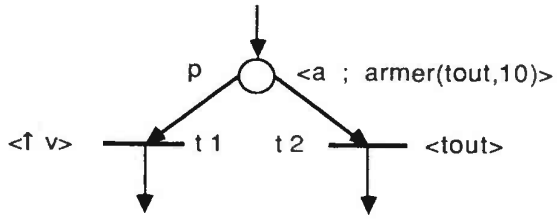


Figure 2.5

c) Les tableaux de variables

Nous retenons la possibilité de définir des tableaux de variables avec les conventions suivantes:

tabv [i] est une référence à la ième variable dans le tableau de variables tabv (NB.: i peut être une constante ou une variable interne. Si tabv[i] est associé à une place ou à une transition, i peut aussi être la composante d'une couleur associée à cette place ou à cette transition)

tabv[i:Typeind] est une référence à une variable quelconque du tableau de variables tabv. Typeind désigne le domaine de définition de l'indice du tableau. Le rang de la variable sélectionnée est affecté à la variable d'indiçage i.

La figure 2.6 illustre un exemple d'utilisation de ces conventions. La réceptivité associée à la transition t exprime la détection d'un événement indiquant le passage de faux à vrai d'une variable d'entrée quelconque du tableau Tve. La variable d'indiçage i prend la valeur du rang de la variable

concernée, et est ensuite utilisée pour sélectionner la variable de sortie de Tvs qui reçoit la valeur vrai dans l'action associée à la place p_2.

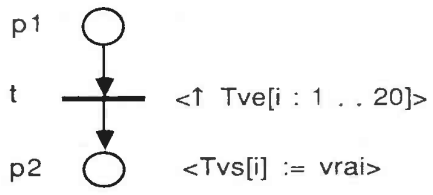


Figure 2.6

d) Les notions d'action ponctuelle et d'action continue

Notons enfin que les actions associées aux places peuvent être impulsionnelles ou continues. Les actions impulsionnelles ne sont exécutées que lorsque, après un tir de transition, le marquage des places auxquelles elles sont associées augmente. Les actions continues par contre sont exécutées après chaque cycle, tant que les places auxquelles elles sont associées sont marquées. Sur un réseau, on distingue les actions impulsionnelles des actions continues en suffixant les actions continues d'une étoile.

2.4.3 LES REGLES DE FONCTIONNEMENT D'UN RdPI

L'un des éléments clés du formalisme des RdPI réside dans la définition de règles de synchronisation de l'évolution des marquages aux occurrences d'événements engendrées par l'environnement. Pour ce qui nous concerne, ces règles doivent offrir des possibilités de description directe, bien adaptées au contexte de la spécification du comportement d'un système réactif à répartir, et éviter toute possibilité d'ambiguïté dans l'interprétation des intentions du spécifieur.

Nous abordons une discussion autour de ces aspects selon deux points de

vue. Le premier concerne la prise en compte des événements pour le tir des transitions, et le second les règles d'arbitrage pour choisir les transitions à franchir quand plusieurs sont franchissables. Nous mettrons tout particulièrement en exergue quelques insuffisances des règles habituelles de fonctionnement des RdPI par rapport aux objectifs indiqués, ce qui nous permettra de mieux justifier les aménagements que nous préconisons.

2.4.3.1 LA PRISE EN COMPTE DES OCCURRENCES D'EVENEMENTS

Regle 1: *Dans la sémantique des RdPI, on considère habituellement une transition* t *(fig. 2.7) à laquelle est associé un évenement* e *et une condition* c *comme étant validée , ssi toutes les places en entrée de cette transition sont suffisamment marquées. Elle devient franchissable dès l'occurrence d'un événement* e, *à condition que* c *soit également vérifiée. Le tir de la transition provoque alors non seulement l'évolution du marquage du réseau, mais aussi l'exécution des opérations associées aux places apparaîssant en sortie de la transition.*

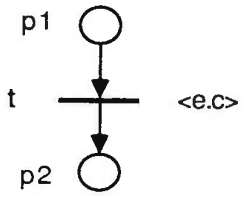


Figure 2.7

Le mécanisme de synchronisation de base fourni par cette règle conduit à ignorer les occurrences de e engendrées par l'environnement aux instants où on n'a pas à la fois t validée et c vérifiée. De ce fait, il ne permet pas d'exprimer que le tir d'une transition peut être synchronisé à une évolution passée de l'environnement. Il en résulte que pour spécifier certaines contraintes nécessitant une connaissance sur le passé de l'environnement, le spécifieur doit

utiliser ce mécanisme de base pour décrire dans le réseau les principes de mémorisation de ce passé. Il peut ainsi être amené à **surcharger** le réseau et à le rendre illisible, ou à **introduire des places non bornées et des transitions non vivantes** nuisibles à l'analyse des séquences de commandes qui peuvent être générées par le système décrit.

Si par exemple, dans la figure 2.7, les intentions du spécifieur étaient que pour chaque occurrence de l'événement e la transition t soit tirée une et une seule fois, le réseau doit être modifié comme indiqué dans la figure 2.8.

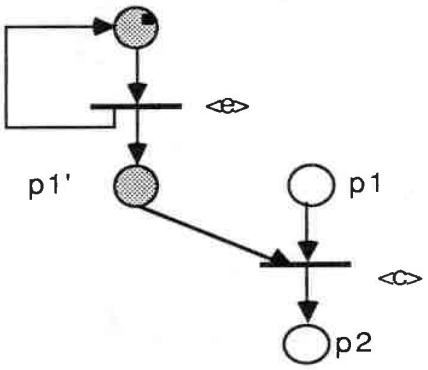
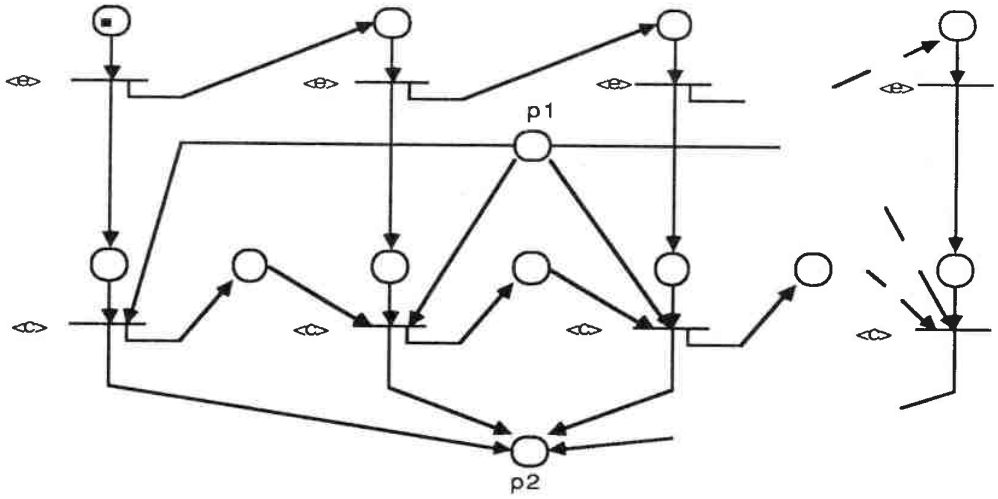


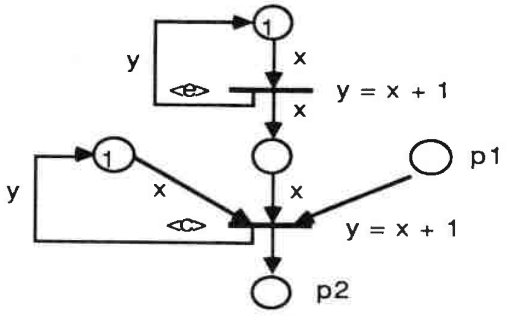
Figure 2.8

Cette modification peut devenir encore plus complexe s'il est important d'expliciter le fait que les événements doivent être pris en compte dans leur ordre de détection (cf. fig. 2.9)

Dans cet exemple, on peut constater en plus que le fait qu'un nombre illimité d'occurrences de l'événement e puissent être engendrées par l'environnement avant d'être prises en compte par le système, a conduit à introduire dans la figure 2.8 une place p_1' non bornée, et dans la figure 2.9 des transitions qui sont toutes non vivantes.



(a) Solution exprimée à l'aide d'un RdP ordinaire



(b) Solution exprimée à l'aide d'un RdP à prédicats

Figure 2.9

Dans un contexte de spécification, il nous paraît plus simple de considérer que les relations de synchronisation entre occurrences d'événements et tirs de transitions découlent de politiques de communication entre le système de

détection des occurrences d'événements (le producteur), et le système chargé de faire évoluer le réseau (le consommateur). Ceci nous rapproche du point de vue développé dans [Thomesse 80] [Ladet 82] [Benmaiza 84]. Plutôt que de modifier le réseau, le spécifieur peut ainsi, de façon plus simple et plus lisible, associer à chaque transition le type de communication qui caractérise la politique de prise en compte des occurrences de l'événement qui lui est associé.

Règle 2 : *Nous considérons alors une transition* t *(fig. 2.10) à laquelle est associé un type d'événement* e, *un type de communication* τ , *et une condition* c, *comme franchissable, ssi on a simultanément* t *validée, la condition* c *vérifiée, et une occurrence de* e *consommable selon* τ . *Si* M *désigne le marquage de départ et* M' *le marquage d'arrivée, on notera alors* $M(t \not\# e(\tau)> M'$.

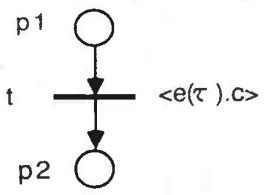


Figure 2.10

Nous présentons de façon informelle, à titre d'illustration, quelques exemples de types de communication . Dans [Perrin 85], on retrouve une formalisation des trois premiers à l'aide de types abstraits algébriques:

- Le type de communication "**égalité**" : Ce type implique que toutes les occurrences engendrées par l'environnement doivent être consommées une et une seule fois.

- Le type de communication "**aléatoire**" : L'occurrence consommée doit toujours correspondre à la dernière occurrence engendrée par l'environnement à cet instant. On peut donc ne jamais consommer certaines occurrences, ou les

consommer plusieurs fois. Ce type de communication est équivalent à la consultation.

- Le type de communication "**rafraichi**" : L'occurrence consommée correspond toujours à la dernière occurrence engendrée par l'environnement à cet instant, mais aucune occurrence ne peut être consommée plus d'une fois.

- Le type de communication "**strictement-rafraichi**" : L'occurrence consommée est toujours la première occurrence engendrée par l'environnement après la demande de consommation. Ce type de communication permet d'exprimer la notion d'événement futur. Nous considérerons ce type comme le type par défaut. Il conduit aux mêmes règles de synchronisation que la règle 1.

- Le type de communication "**par bascule**" : L'occurence consommée correspond toujours à la première occurrence engendrée par l'environnement après la dernière consommation.

D'autres types de communication peuvent être définis pour caractériser des relations de synchronisation avec l'environnement, aussi complexes qu'on le souhaite, prédéfinis ou spécifiques à une application, sans avoir à dire à un stade de spécification, comment ils sont réalisés.

2.4.3.2. LES REGLES D'ARBITRAGE POUR LE CHOIX DES TRANSITIONS A FRANCHIR

Habituellement, on définit les règles de fonctionnement d'un RdPI marqué $< R_I; M_0 >$ par rapport à un ensemble de séquences de parties de l'ensemble E des événements pouvant être engendrés par l'environnement du système modélisé, chaque élément X de cette séquence correspondant à une occurrence simultanée des événements qui le compose.

Pour un marquage accessible M de $< R_I; M_0 >$, l'occurrence d'un ou de

plusieurs événements peut rendre franchissables plusieurs transitions. On dira que ces transitions sont **réceptives** aux événements concernés. L'ordre de franchissement de transitions réceptives à un ensemble d'événements n'est pas toujours indifférent : le franchissement de certaines transitions peut en invalider d'autres, et certaines transitions peuvent être franchies plusieurs fois. Par exemple, dans le réseau de la figure 2.11 utilisé dans [Moalla 85] pour illustrer le fonctionnement des RdPI, si l'on suppose l'occurrence simultanée des événements b et h, il existe des séquences où la transition t_4 peut être franchie deux fois (ex.: (t_4 t_3 t_4 t_2)) et où le franchissement de la transition t_2 invalide celui de t4 (ex.: (t2 t3)).

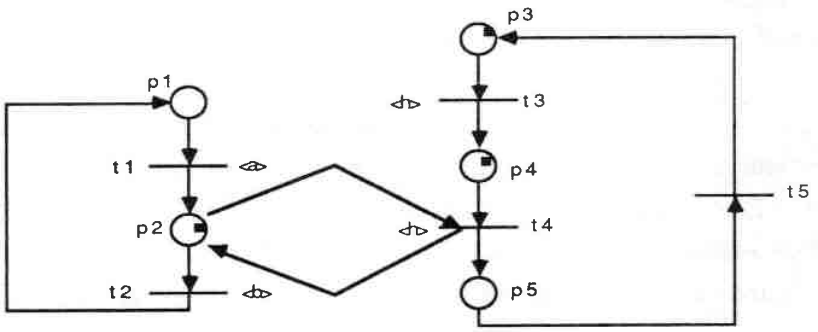


Figure 2.11

Pour éviter toute ambiguïté d'interprétation des intentions du spécifieur, il est donc primordial que soient définis des règles précises pour le choix des transitions réceptives à franchir.

Dans [Moalla 85], si pour un marquage accessible M de $< R_I; M_0 >$, $T_{X,M}$ = {t1,t2,.....,tn} désigne l'ensemble des transitions réceptives à un ensemble d'événements X, on appelle **séquence de simulation complète** à partir M, toute séquence de franchissements s_c telle que :

- si une transition t apparaît dans s_c alors elle est nécessairement incluse dans $T_{X,M}$
- toute transition t de $T_{X,M}$ n'apparaît au plus qu'une seule fois dans s_c
- toute séquence s_c' obtenue de s_c par permutation de transitions est franchissable à partir de M
- il n'existe pas d'autres séquences plus longues qui contiennent toutes les transitions de s_c et qui vérifient les trois premières conditions .

On dit qu'un RdPI marqué $< R_I; M_0 >$ est dans un **état stable**, si le marquage atteint est tel qu'aucune transition ne peut être franchie sans prendre en compte de nouveaux événements.

Lorsque dans un état stable, plusieurs transitions sont réceptives à un ensemble d'événements, on choisit d'abord de franchir une séquence de simulation complète, puis ensuite, on effectue des tirs répétés des transitions franchissables, sans la prise en compte d'un nouvel événement, jusqu'à ce que l'on atteigne un nouvel état stable. Lorsque pour un marquage accessible et un ensemble d'événements on peut de cette façon atteindre des états stables différents, on dit que le réseau n'est pas **persistant**. Lorsque pour un marquage accessible et un ensemble d'événements on peut ne jamais atteindre un état stable, on dit que le réseau n'est pas **prompt**.

Ces règles d'arbitrage permettent essentiellement au spécifieur

- d'introduire de façon controlée du non déterminisme et des sérialisations dans le comportement d'un système, en imposant dans certaines situations un choix aléatoire entre plusieurs tirs groupés de transitions

- ou au contraire de vérifier que le réseau construit n'est pas par erreur non persistant ou non prompt.

Ainsi, par exemple, si nous revenons au réseau de la figure 2.11, on peut vérifier que le comportement décrit n'est pas déterministe: pour le marquage initial, l'occurrence simultanée des événements b et h rend franchissable les séquences de simulation complète (t2 t3) et (t3 t4). On notera également que, selon que l'environnement engendre b et h simultanément, ou b suivi de h, ou h suivi de b, la séquence de transitions franchies peut être différente.

Dans un contexte centralisé, il est aisé de contraindre un système à se comporter conformément aux intentions d'un spécifieur :

- en définissant des intervalles d'observation pendant lesquels des événements détectés séquentiellement sont considérés comme simultanés

- et en faisant évoluer le système selon l'algorithme suivant :

cycle
Prise en compte des nouveaux événements
Choix d'une séquence de simulation complète
tir de la séquence de simulation complète choisie
recherche d'un état stable
fcycle

Par contre, dans un contexte réparti, il est en pratique impossible de faire évoluer un système en respectant les intentions du spécifieur : en l'absence d'horloge commune , il est en effet difficile de définir des critères rigoureux pour comparer des événements engendrés par l'environnement sur des sites répartis. Par exemple, dans le réseau de la figure 2.11, nous avons déjà vu que le spécifieur imposait un comportement différent selon que l'environnement engendrait b et h simultanément, ou b suivi de h, ou h suivi de b. Si en pratique l'on ne peut détecter la simultanéité de b et de h, ou l'ordre dans lequel ils sont engendrés, il est bien évident qu'on ne pourra jamais contraindre un système à se conformer à cette spécification.

En plus, le choix d'une séquence de simulation complète à un instant donné nécessite une connaissance exacte de l'état global du système dont on ne peut disposer dans un contexte réparti.

Pour avoir à notre disposition un modèle qui permette de construire une spécification qui ait du sens indépendamment du fait que le système spécifié est potentiellement réparti ou non, nous retiendrons plutôt les règles d'arbitrage suivantes:

1) D'abord nous considérerons que tous **les événements sont disjoints dans le temps**. On vérifiera alors que l'occurrence simultanée de deux événements conduit à un certain résultat en vérifiant que, quel que soit l'ordre d'occurrence de ces deux événements, le résultat atteint reste le même.

2) Si pour un marquage accessible M $T_{e,M}$ = {t1, t2,....,tn} désigne l'ensemble des transitions réceptives à l'événement e, nous appellerons séquence de simulation complète à partir de M, toute séquence de franchissements s_c telle que :

   - si une transition t apparaît dans s_c alors elle est nécessairement incluse dans $T_{e,M}$
   - toute transition t de $T_{e,M}$ n'apparaît au plus qu'une seule fois dans s_c
   - il n'existe pas d'autres séquences plus longues qui contiennent toutes les transitions de s_c et qui vérifient les deux premières conditions .

Pour un marquage accessible M et un événement e , nous noterons alors par

$T^*_{e,M} = \{s_{c1}, s_{c2},.....,s_{ck}\}$ l'ensemble des séquences de simulation

complètes

et $T_M = \{t_1, t_2, ..., t_l\}$ l'ensemble des autres transitions franchissables mais non synchronisées à e

3) Nous considérerons que, lorsque pour un marquage accessible M et un événement e plusieurs transitions sont franchissables, on choisit de façon aléatoire, soit de franchir une séquence de simulation complète $s_{ci} \in T^*_{e,M}$, soit de franchir une transition quelconque de T_M. Nous dirons dans ces conditions qu'un réseau marqué $< R_I; M_0 >$ est persistant, ssi pour tout marquage accessible M et pour tout e, tout couple de transitions t_1, t_2 franchissables peuvent être franchies dans n'importe quel ordre. ⊗

Par rapport aux règles habituelles de fonctionnement des RdPI, lorsque pour un marquage accessible et pour un événement donné plusieurs transitions sont franchissables, ces règles introduisent moins de contraintes dans l'ordre de franchissement des transitions. Les contraintes d'ordonnancement impossibles à respecter dans un contexte réparti ont particulièrement été évitées. Toutefois, nous avons gardé la possibilité d'imposer dans certaines situations un tir groupé de transitions en associant un événement à plusieurs transitions qui peuvent être simultanément validées. Dans ce cas, le concepteur devra admettre que le tir des transitions concernées modélise des changements d'état sur le même site et qu'elles ne seront jamais réparties.

L'une des conséquences immédiates de ces nouvelles règles de fonctionnement réside dans le fait qu'elles permettent d'énoncer assez simplement une condition suffisante pour que l'interprétation associée à un RdPI $R_I = < R; V; OP; \Phi, \Psi >$ ne modifie pas les propriétés du réseau sous-jacent R:

1) Pour tout couple de transitions (t1, t2) ∈ T dont les tirs doivent être synchronisés aux occurrences d'un même événement, il ne doit pas exister

de marquage accessible dans le réseau R qui valide simultanément t1 et t2. En d'autres termes, la synchronisation avec l'environnement ne doit jamais imposer de tir groupé de transitions.

2) Pour tout marquage accessible dans R, les transitions validées ne doivent être synchronisées qu'à des événements indépendants, qui peuvent se produire dans n'importe quel ordre. Ce qui revient à dire que pour tout marquage accessible dans R_I, le choix de la transition à franchir peut être considéré comme aléatoire, donc que l'évolution de R_I a les mêmes contraintes que l'évolution de R.

Cette propriété nous sera utile au chapitre 6. On remarquera qu'avec la règle de Moalla, sous les mêmes hypothèses, un RdPI n'a pas nécessairement les mêmes propriétés que le réseau autonome sous-jacent. A partir d'un marquage stable, on franchit tout d'abord une des transitions réceptives à un événement, puis ensuite on franchit les transitions non synchronisées jusqu'à ce l'on atteigne un nouvel état stable.

2.5. SPECIFICATION DU COMPORTEMENT INTERNE A L'AIDE DES RdP INTERPRETES

2.5.1. REMARQUES PRELIMINAIRES

Deux approches sont en général retenues pour traduire dans une spécification à l'aide de RdP les interactions entre entités communicantes.

Dans la première approche, ces interactions sont exprimées à l'aide de **places partagées** [Valette 82] [Courvoisier 83] [Ramamoorthy 85] [Berthelot 85 b] . Pour émettre, l'entité émettrice franchit une transition qui produit une marque dans la place partagée. Pour recevoir le message ainsi émis, l'entité receptrice franchit une transition qui consomme la marque produite(fig. 2.12).

Ce mécanisme de base exprime une communication asynchrone via un tampon non borné, mais peut être utilisé pour construire d'autres types de communication (fig. 2.13). Au besoin, cela peut nécessiter comme dans la figure 2.13 (c) de spécifier explicitement un canal de communication assurant les interactions des entités communicantes selon la politique souhaitée. Dans ce cas, les places de communication de chaque entité communicante définissent son interface avec ce canal.

Notons toutefois que dans les exemples présentés, les marques des places partagées ne servent à représenter que des événements de communication, sans référence aux messages véhiculés. Le modèle des RdP colorés ou à prédicats permet, de façon simple, d'associer en plus aux marques des paramètres caractérisant ces messages.

L'intérêt de cette approche réside essentiellement dans le fait qu'il est possible de valider directement le comportement d'un réseau d'entités communicantes en s'appuyant sur la théorie des RdP, puisque séquencement des opérations et séquencement des communications sont entièrement exprimés dans ce formalisme.

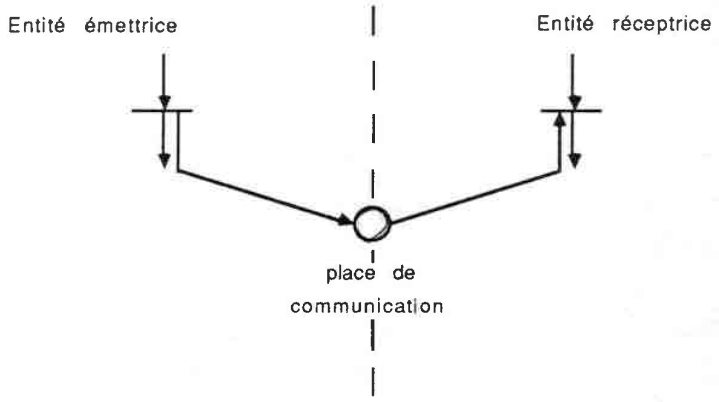
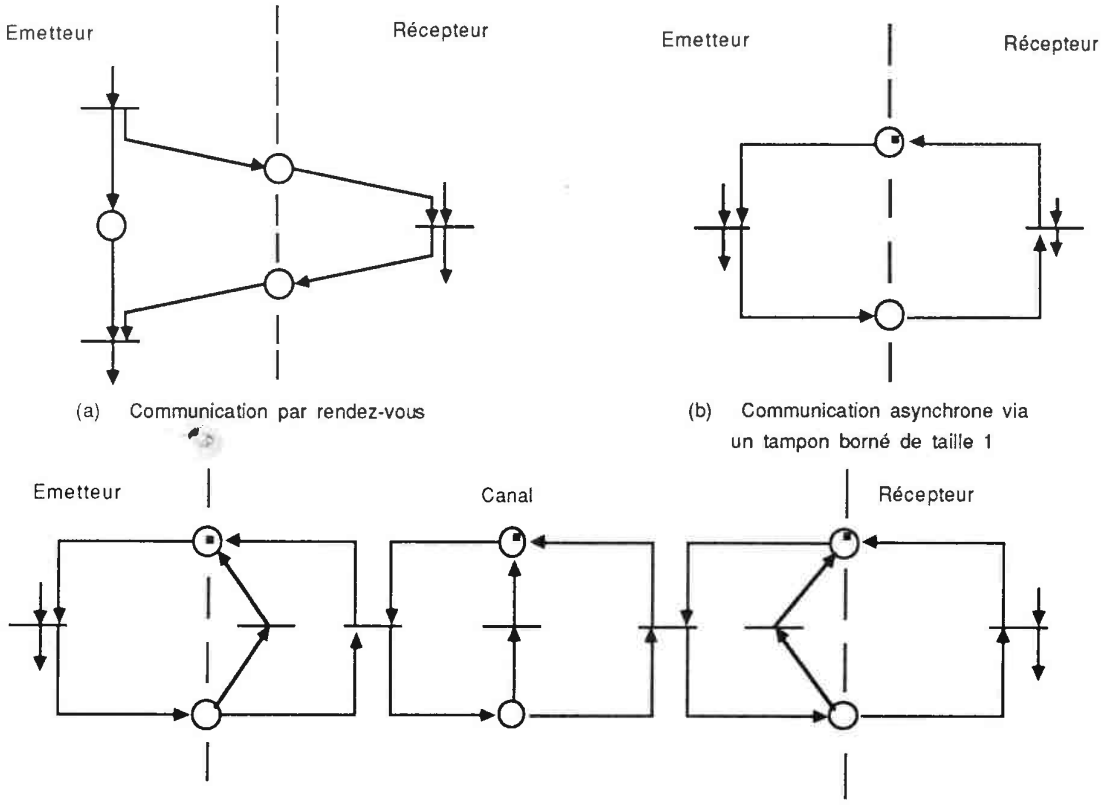


Figure 2.12



(a) Communication par rendez-vous

(b) Communication asynchrone via un tampon borné de taille 1

(c) Communication asynchrone via un tampon borné de taille 3 avec possibilité de perte de messages

Figure 2.13

Dans la deuxième approche, les interactions entre entités communicantes sont exprimées en synchronisant le tir des transitions à des opérations de communication.

Dans [Bochmann 77], on procède de cette façon sans imposer une sémantique unique aux opérations de communication de base qui peuvent être ainsi utilisées. Pour analyser le comportement d'un réseau d'entités communicantes, on ne peut plus alors utiliser que partiellement les techniques de preuves propres aux RdP, en les combinant à des techniques de vérification d'invariants par induction [Keller 76].

Dans [Ayache 85], on procède également de la même manière, mais en utilisant un seul type de communication de base qui est celui du **rendez-vous** [Hoare 78]. En étiquetant dans une entité une opération d'émission d'un message m (notée E ! m) à une transition, et dans une autre entité une opération de réception du même message m (notée R ? m)à une autre transition, on exprime ainsi que ces deux transitions ne doivent être franchies que simultanément (fig. 2.14).

Comme pour la communication à l'aide de places partagées, ce mécanisme de base peut être utilisé pour exprimer des interactions via un canal modélisant d'autres types de communication (fig. 2.15).

Par rapport à la communication à l'aide de places partagées, les communications sont exprimées dans cette approche dans un formalisme autre que celui des RdP. Toutefois, l'interconnexion des transitions liées par un même rendez-vous permet d'obtenir un RdP global ayant les mêmes propriétés que le réseau de RdP initial (fig. 2.16). Le comportement d'un système réparti peut ainsi être étudié à partir de ce réseau global en utilisant les techniques de preuves propres aux RdP.

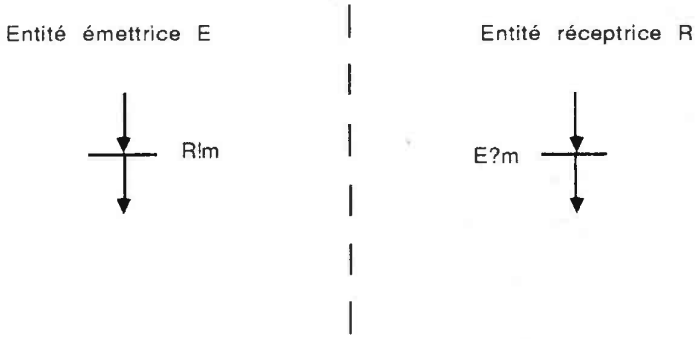
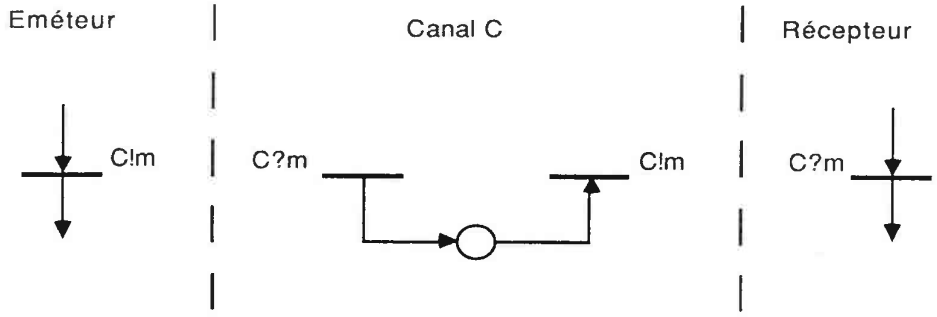
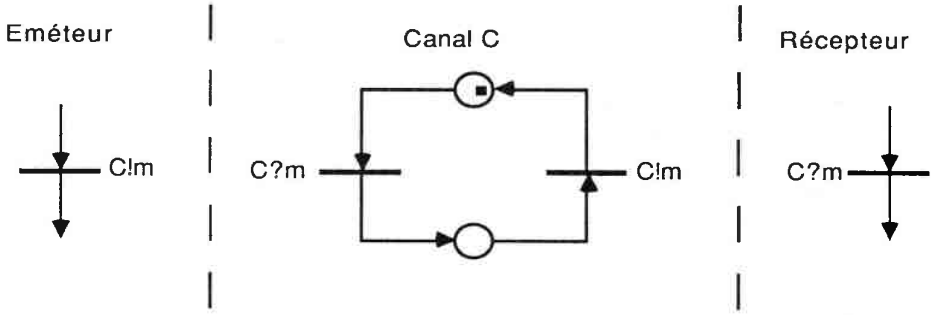


Figure 2.14



(a) Communication asynchrone via un tampon non borné



(b) Communication asynchrone via un tampon borné de taille 1

Figure 2.15

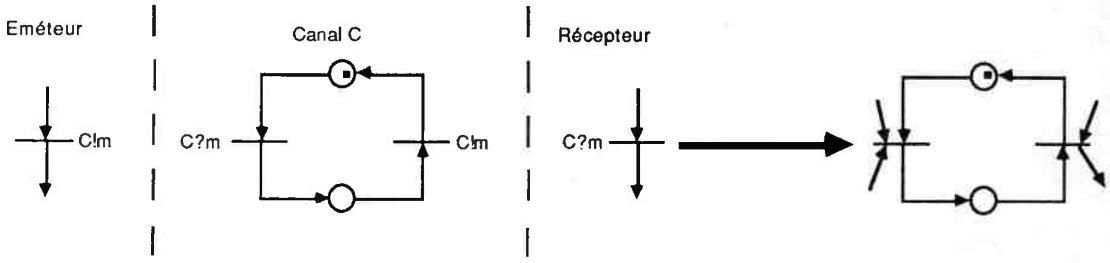


Figure 2.16

Par ailleurs, on peut noter que dans un certain sens, ces deux approches sont duales : On peut décrire la communication à l'aide d'une place partagée en utilisant des rendez-vous, et vise-versa (cf. fig. 2.15 (a) et 2.16 (a)). Cependant, d'un point de vue comportemental, cette dualité n'est pas parfaite: En utilisant la communication à l'aide de places partagées, on ne peut exprimer un choix non déterministe (du type des commandes gardées de Dijkstra) entre une opération de communication par rendez-vous et toute autre opération. Par contre, l'autre approche laisse cette possibilité ouverte. Par exemple, dans la figure 2.17 (a), tant que le rendez-vous n'a pas été accepté, la transition t1 peut être franchie en utilisant la marque qui valide la transition t2. Dans la figure 2.17 (b), nous avons traduit le rendez-vous en utilisant une communication à l'aide de places partagées. Le franchissement de la transition t2 ne veut pas dire que le rendez-vous a été accepté, mais invalide le tir de t1. Nous verrons au chapitre 7 que cette différence a son importance.

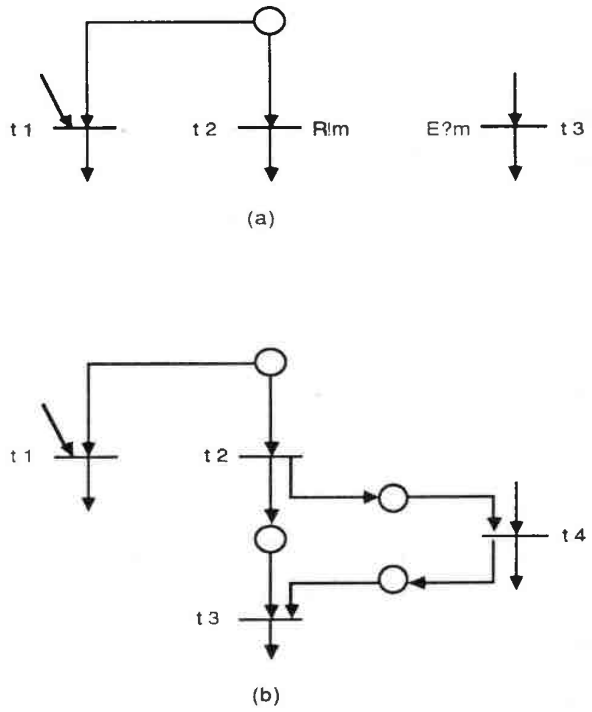


Figure 2.17

La solution que nous retenons pour décrire les interactions entre entités communicantes relève de la deuxième approche. Toutefois, nous avons choisi de nous appuyer sur les deux types de communication synchrone et asynchrone. Ainsi, nous gardons dans notre modèle la possibilité d'exprimer de façon directe, dans un contexte réparti, deux formes de synchronisation de base que le modèle des RdP autonomes permet. Pour le type d'applications qui nous intéresse (la commande des procédés industriels), ceci nous paraît suffisant pour exprimer de la façon la plus concise possible la coopération entre des tâches parallèles, l'essentiel des contraintes d'évolution de ces tâches étant plutôt dictées par l'environnement. Enfin, pour permettre une expression concise de la synchronisation des interactions inter-modulaires à l'occurrence d'événements externes, nous retiendrons en plus des formes syntaxiques particulières. Les sections qui suivent décrivent dans les détails ces différentes extensions de notre modèle de RdPI de base.

2.5.2. LES COMMUNICATIONS SYNCHRONES (CSP [Hoare 78])

Nous désignons par PE?m une opération de réception du message m sur le port d'entrée PE, et par PS!m une opération d'émission du message m vers le port de sortie PS. Contrairement à CSP, on peut ainsi définir des rendez-vous entre plusieurs partenaires

- via des liaisons convergentes entre n émetteurs et un récepteur (fig 2.18),
- ou via des liaisons divergentes entre un émetteur et n récepteurs (fig 2.19).

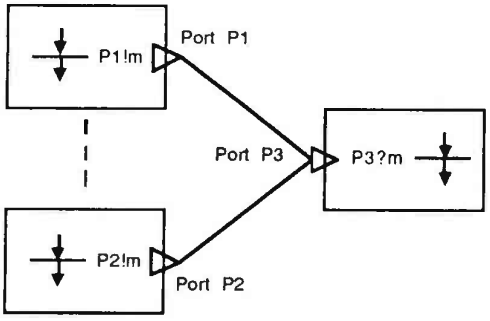


Figure 2.18

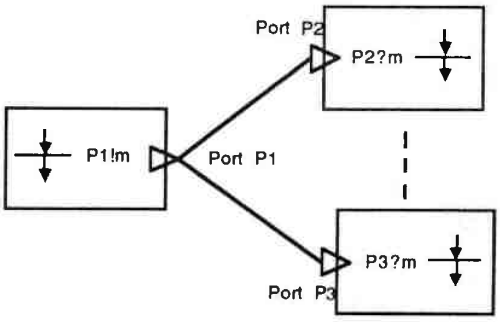


Figure 2.19

Dans le cas d'une liaison convergente, le récepteur acquitte un rendez-vous ssi les n émetteurs ont tous émis le même message, tandis que dans le cas d'une liaison divergente, l'émetteur n'acquitte un rendez-vous que ssi les n récepteurs attendent le même message. Chaque fois que cela ne crée pas d'ambiguité, nous omettrons la désignation des ports.

REGLE : *Les opérations de communication synchrone modifient les règles d'évolution des* RdPI *de la façon suivante:*

Dans la figure 2.20, dans (1) et (2), si la transition t *est validée, elle devient franchissable dès qu'une occurrence de* e *devient consommable selon* τ *, à condition que* c *soit vérifiée. Le tir de la transition est alors différé jusqu'à ce que le rendez-vous exprimé par* ?M *ou* ! M *devienne exécutable. Pendant ce temps, les marques dans les places en entrée qui rendent la transition valide sont consommées et ne sont donc plus disponibles pour le tir des autres transitions.*

Par contre, les notations (3) et (4) imposent les mêmes règles d'évolution qu'un événement externe: si à un instant donné la transition t est validée, elle devient franchissable dès que le rendez-vous peut avoir lieu. ⊗

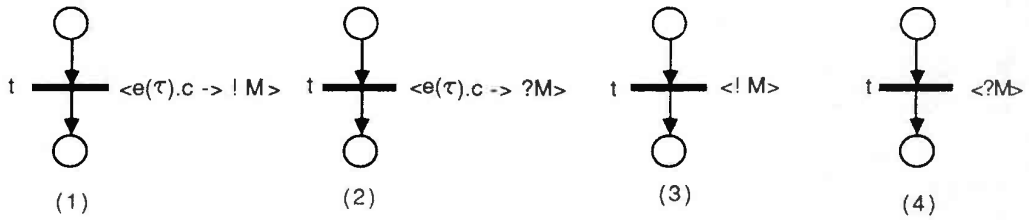


Figure 2.20

Dans la figure 2.21, nous donnons quelques notations équivalentes significatives portant sur des interactions entre deux entités communicantes. On remarquera tout particulièrement que la notion de tir différé conduit à traduire toute transition t étiquetée par < e(τ).c->!M > ou < e(τ).c ->?M > par deux

transitions (a, $\underline{a}$) telles que : si t est franchissable et si le rendez-vous qui lui est associé peut avoir lieu, (a, $\underline{a}$) forment une séquence de franchissements ininterruptible, sinon, si t est franchissable et si le rendez-vous qui lui est associé n'est pas immédiatement réalisable, (a, $\underline{a}$) seront franchies dans cet ordre, mais peuvent être séparées par le franchissement d'autres transitions. En se référant au schéma d'interconnexion des transitions décrit précédemment (c.f. fig. 2.16) on peut voir comment dans notre modèle, des transitions étiquetées par des opérations de communication synchrone peuvent être interconnectées pour obtenir un réseau global.

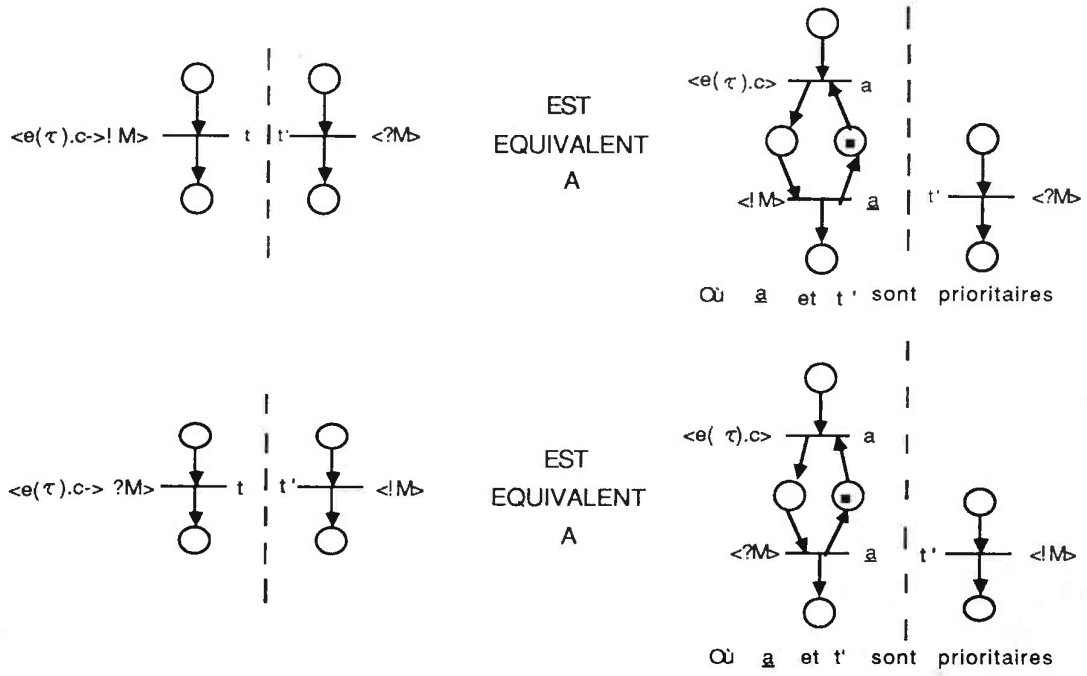


Figure 2.21

Le franchissement de transitions $t_1, t_2,....,t_n$ appartenant à des modules différents et liées par un même rendez-vous s'effectue donc toujours de façon synchrone. Il peut être synchronisé à l'occurrence d'un événement e consommable selon une politique τ. On notera alors $M_0(\{t_1,t_2,...,t_n\} // e(\tau) > M$

pour dire que ce franchissement permet de passer du marquage M_0 au marquage M.

2.5.3 LA COMMUNICATIONS ASYNCHRONE

Les conventions de notation que nous retenons pour les opérations de communication asynchrones via des tampons non bornés sont les suivantes: PE?/m pour la réception et PS!/m pour l'émission. Ici également, nous admettrons des communications entre plusieurs partenaires via des liaisons convergentes et divergentes. Dans le cas d'une liaison convergente, chaque message émis par chacun des n émetteurs est reçu par le récepteur. Dans le cas d'une liaison divergente, chaque message émis par l'émetteur ne sera reçu que par un seul des n récepteurs choisi au hasard parmi les récepteurs prêts à recevoir le message. Ce choix de politique de communication est imposé par les règles de production et de consommation des marques dans un RdP.

REGLE : *Les opérations de communication asynchrone modifient les règles d'évolution des* RdPI *de la façon suivante:*

Dans la figure 2.22, dans (1) et (3), la réception du message M *est traitée comme une condition pouvant rendre la transition* t *franchissable. Dans (1), si la transition* t *est validée, elle devient franchissable dès qu'une occurrence de* e *devient consommable selon* τ *à condition que* c *soit vérifiée et que le message* M *soit recevable . Par ailleurs , dans (3), si la transition* t *est validée, elle devient franchissable si le message* M *est recevable. Par contre, dans (2) et (4), les opérations de communication ne modifient en rien les règles d'évolution des marquages.*

Nous donnons là également dans la figure 2.23 quelques notations équivalentes significatives portant sur des interactions entre deux entités communicantes. Ces notations permettent de rapprocher primitives de communication asynchrone et communication à l'aide de places partagées. Par ailleurs, elles montrent comment dans notre modèle, des transitions étiquetées

par des opérations de communication asynchrone peuvent être interconnectées pour obtenir un réseau global.

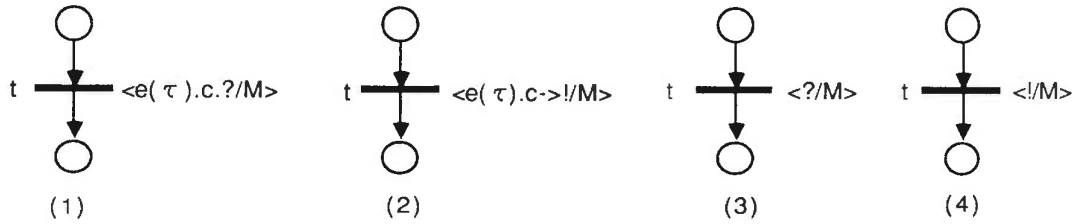


Figure 2.22

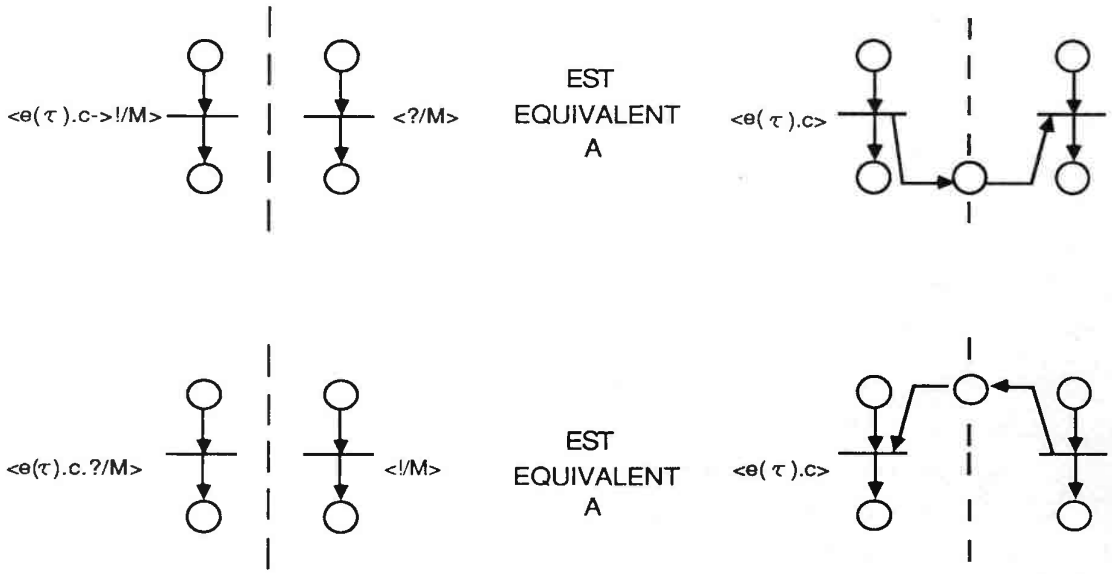


Figure 2.23

Des transitions t1, t2, ..., tn appartenant à des modules différents et liés par une opération de communication asynchrone se franchissent donc séparément, le franchissement des transitions étiquetées par des opérations de

réception devant être précédé de celui des transitions étiquetées par des opérations d'émission.

2.5.4. L'UTILISATION COMBINEE DES COMMUNICATIONS SYNCHRONES ET ASYNCHRONES

A partir des définitions précédentes, nous en déduisons des formes syntaxiques plus générales, qui permettent d'associer au tir d'une transition, à la fois des interactions synchrones et asynchrones (fig 2.24). Ces interactions s'effectuent toujours dans un ordre bien déterminé : réception asynchrone, rendez-vous, et émission asynchrone.

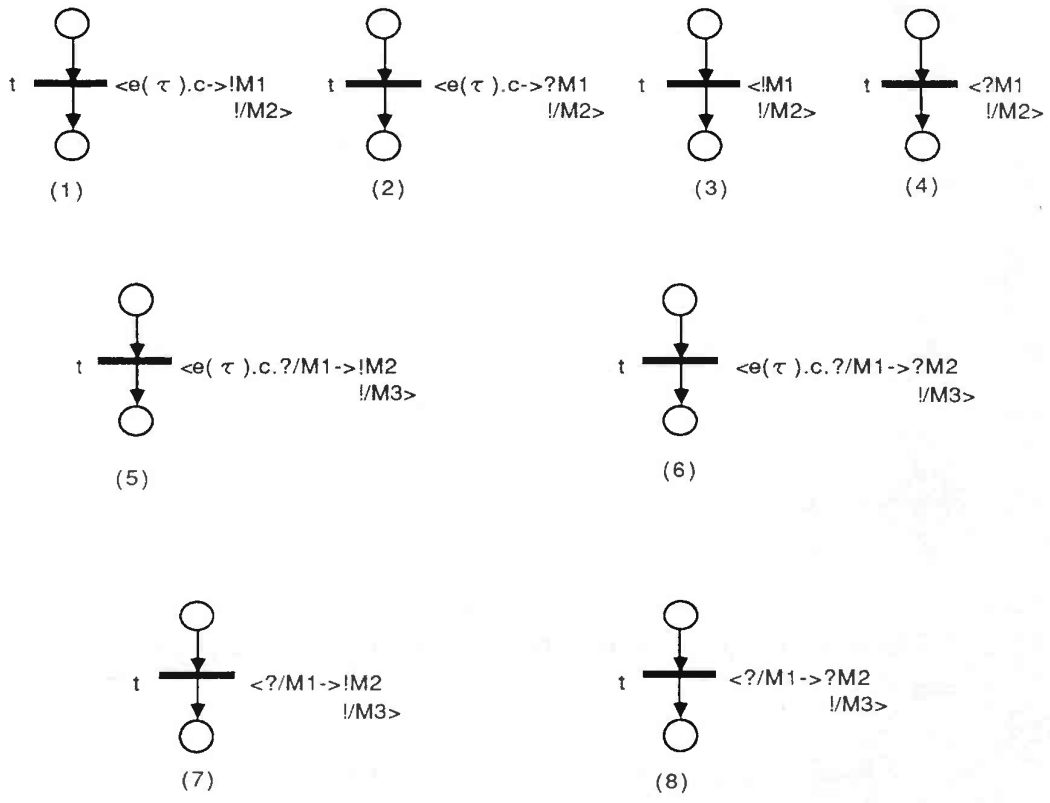


Figure 2.24

La figure 2.25 donne un exemple de notations équivalentes, qui montre bien comment à l'aide des deux primitives de communication synchrone et asynchrone, on peut exprimer de façon concise dans un contexte réparti, deux formes de synchronisation de base des RdP autonomes.

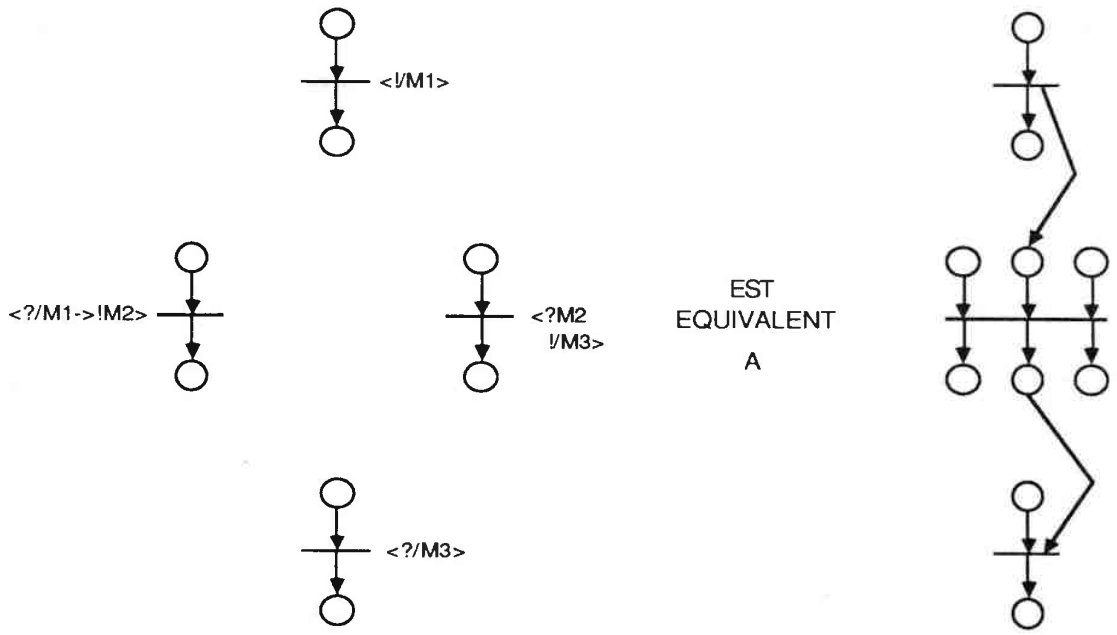


Figure 2.25

2.6 EXEMPLES

Clarté et concision sont deux critères de qualité d'une spécification indispensables pour faciliter le dialogue entre concepteurs et utilisateurs. Dans bon nombre de formalismes, ces deux aspects sont difficilement conciliables.

Nous terminons la présentation du formalisme que nous avons retenu, en illustrant sur deux exemples de procédés industriels réels, la clarté et la concision avec laquelle le comportement d'un système de commande peut être

décrit. Ces exemples nous servirons de support de réflexion dans les chapitres qui suivront.

2.6.1 LA POMPE DE KRAMER [Kramer 80]

Enoncé informel du problème :

Il s'agit de maintenir en dessous d'un certain seuil le niveau de l'eau, et d'assurer la surveillance de l'environnement atmosphérique, dans une galerie sousterraine d'une mine de charbon. On dispose pour cela

- d'une pompe de drainage pouvant recevoir les commandes de fonctionnement "dem" pour sa mise en route et "stop" pour son arrêt,

- d'un capteur de niveau servant à détecter les seuils "haut" et "bas" de l'eau dans la galerie,

- et de capteurs de méthane, de monoxyde de carbonne (CO_2), et d'aération qui permettent à tout instant de connaître le taux de ces gaz.

La conduite du dispositif est assuré en surface par un opérateur qui peut le mettre en service (commande "marche") ou hors service (commande "arrêt"). Cet opérateur doit par ailleurs avoir la possibilité de connaître l'état de la pompe et des gaz sur simple requête, et doit être informé sous forme d'alarmes de tout dépassement de seuil des gaz.

La pompe doit fonctionner automatiquement une fois qu'elle a été mise en service par l'opérateur, en fonction du niveau de l'eau dans la galerie. Pour des raisons de sécurité, elle ne doit pas démarrer ou continuer à fonctionner, dès que le pourcentage de méthane atteint un certain seuil.

Description formelle de l'énoncé :

Nous définissons les variables d'E/S du système de commande de ce dispositif comme suit :

Les variables d'entrée

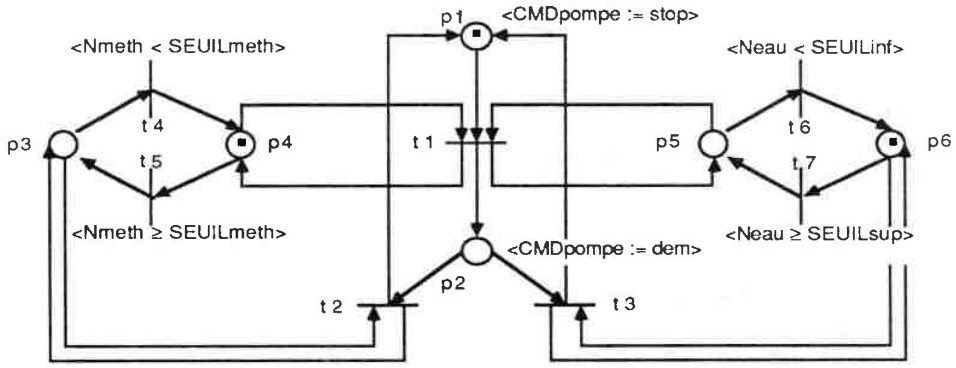
Neau : réel	- Désigne le niveau de l'eau dans la galerie
Nmeth : réel	- Désigne le taux de méthane dans l'atmosphère
Nco_2 : réel	- Désigne le taux de CO_2 dans l'atmosphère
Nair : réel	- Désigne le niveau d'aération dans l'atmosphère
MARCHEop : booléen	- Sert à détecter l'ordre de mise en service du dispositif par l'opérateur
ARRETop : booléen	- Sert à détecter l'ordre de mise hors service du dispositif par l'opérateur
REQpompe : booléen	- Est vrai si l'opérateur demande l'état de la pompe
REQmeth : booléen	- Sert à détecter les requêtes de l'opérateur sur le taux de méthane
$REQco_2$: booléen	- Sert à détecter les requêtes de l'opérateur sur le taux de CO_2
REQair : booléen	- Sert à détecter les requêtes de l'opérateur sur le niveau d'aération

Les variables de sortie

CMDpompe:(dem,stop)	- Désigne la commande à appliquer à la pompe
REPetatp : (dem,stop)	- Fournit à l'opérateur l'état de la pompe
REPmeth : réel	- Fournit à l'opérateur le taux de méthane

REPco_2 : réel	- Fournit à l'opérateur le taux de CO_2
ALMmeth : booléen	- Est mis à vrai pour déclencher l'alarme du méthane
ALMco_2 : booléen	- Est mis à vrai pour déclencher l'alarme de CO_2
ALMair : booléen	- Est mis à vrai pour déclencher l'alarme d'air

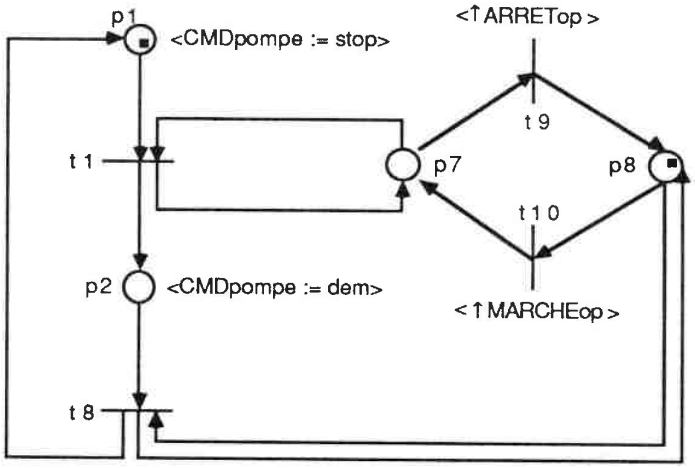
Le comportement du système de commande peut être décrit en quatre parties. La première partie (fig. 2.27) décrit comment la mise en route et l'arrêt de la pompe alternent en fonction du niveau de l'eau et du niveau de méthane dans la galerie.



N.B.: "pompe en état d'arrêt" = M(p1) = 1
"pompe en état de marche" = M(p2) = 1
"taux du méthane en dessous du seuil" = M(p4) = 1
"taux du méthane au dessus du seuil" = M(p3) = 1
"le niveau de l'eau a atteint le seuil supérieur et n'est pas redescendu jusqu'au seuil inférieur" = M(p5) = 1
"le niveau de l'eau est descendu jusqu'au seuil inférieur et n'est pas remonté jusqu'au seuil supérieur" = M(p6) = 1

Figure 2.27

La deuxième partie de la spécification (fig. 2.28) décrit comment la mise en route et l'arrêt de la pompe alternent avec les ordres de mise en service et de mise hors service du dispositif par l'opérateur. Cette spécification est construite autour de la transition t_1 et des places p_1 et p_2 définies dans la spécification précédente, en faisant abstraction des autres transitions et places.



NB.: "dispositif en service" = M(p7) = 1
"dispositif hors service" = M(p8) = 1

Figure 2.28

La troisième partie de la spécification concerne les demandes d'informations de l'opérateur sur l'état de la pompe (fig. 2.29), le taux de méthane (fig. 2.30), le taux de CO_2 (fig. 2.31), et le niveau d'aération (fig. 2.32). Comme dans la deuxième partie, certaines des places (p_1, p_2) apparaîssant dans cette spécification sont celles déjà définies dans la première partie.

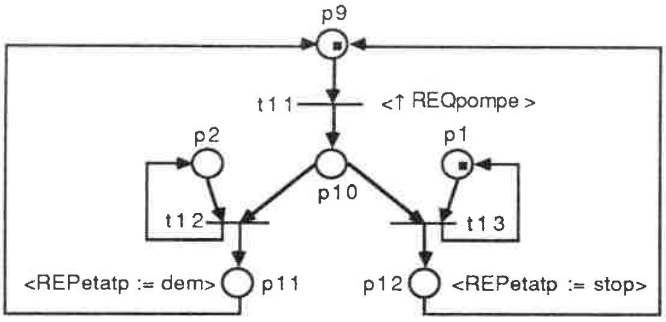


Figure 2.29

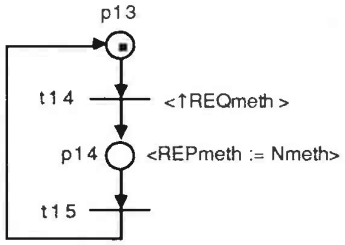


Figure 2.30

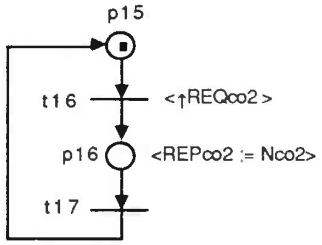


Figure 2.31

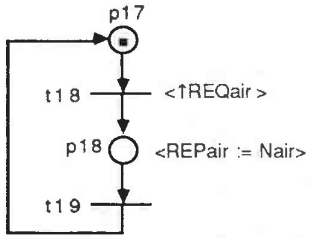


Figure 2.32

Enfin, la quatrième partie de la spécification décrit les conditions de déclenchement des alarmes pour le dépassement du seuil de méthane(fig. 2.33), du seuil de CO_2 (fig. 2.34), et du seuil d'aération (fig. 2.35). Elle s'appuie également sur les places p_3, p_4 et les transitions t_4, t_5 définies dans la première partie.

Chaque réseau dans chacune des parties de cette spécification est incomplet mais précise de façon complète et formelle une contrainte particulière de l'énoncé initial en faisant abstraction du reste. Tout en étant logique, le découpage proposé n'est en fait ni un découpage en procédures qui s'appellent, ni un découpage en modules communicants comme on a l'habitude de le faire en programmation. Ce découpage relève plutôt d'une formulation de

problème par des énoncés orthogonaux de contraintes comme nous l'avons déjà rencontré dans le modèle de Chen et Yeh. La spécification globale du problème s'obtient par simple fusion des différents réseaux. Tout comme les appels de procédures et les modules communicants, cette approche est aussi une façon de parcéliser un problème pour le rendre maîtrisable par l'esprit humain, et facilement modifiable.

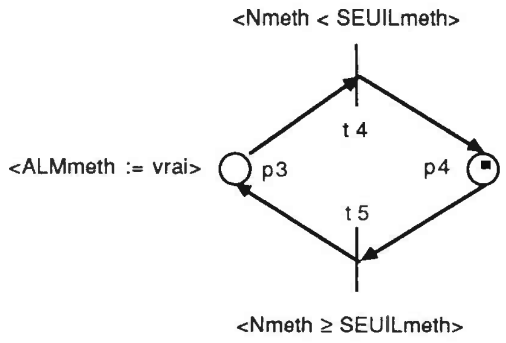


Figure 2.33

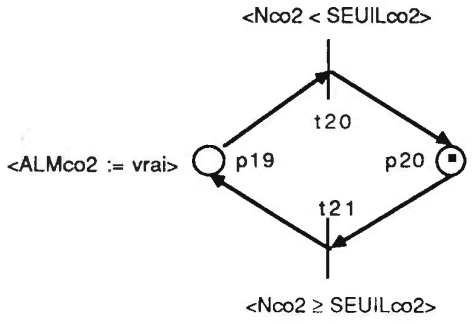


Figure 2.34

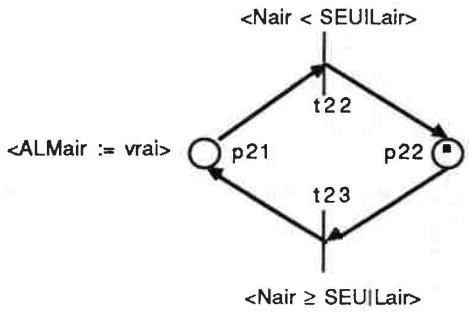


Figure 2.35

2.6.2 L'ATELIER DE BOBINAGE [Hollinger 85]

Enoncé informel du problème :

Cet exemple concerne un atelier de production de bobines de fil textile. L'atelier est composé de bobinoirs groupés en métiers. Nous ne nous intéresserons qu'au fonctionnement d'un métier. Une bobine est obtenue à l'aide d'un bobinoir entrainant en rotation un axe équipé d'un tube en carton, sur lequel est accroché l'amorce d'un fil en défilement continu. Lorsque la bobine est achevée, le fil est coupé en amont du bobinoir puis dévié vers un collecteur de déchets. La bobine achevée est retirée, et remplacée par un tube vide.

La coupe du fil, le remplacement d'une bobine et l'accrochage de l'amorce du fil sont assurés par un système embarqué; une navette assure le transport des tubes vides et des bobines achevées. Le système embarqué et la navette se déplacent chacun sur une voie parallèle au métier, à partir de leurs positions de dégagement. Lorsqu'une bobine est en voie d'achèvement, une requête est transmise au système embarqué et à la navette qui déclenchent leur déplacements vers le bobinoir concerné. Le remplacement de la bobine par un tube vide débute alors dès que le système embarqué termine sa course, et dès que la fin effective du bobinage intervient. Si un incident de casse de fil se produit sur le bobinoir, le système embarqué et la navette retournent à leur position de dégagement.

Description formelle de l'énoncé :

Nous définisons comme suit les variables d'E/S du système de commande d'un métier:

Les variables d'entrée

TABbool = **tableau** [1 .. n] **de** boolèen
Ereq : TABbool Ereq[i] = vrai indique qu'une bobine est en voie

	est détectée.
Efdepn : TABbool	Efdepn[i] = vrai indique que la fin de course de la navette vers le bobinoir no. i est détectée.
Efretn : booléen	Efretn = vrai indique que la fin de course de la navette vers sa position de dégagement est détectée.

Les variables de sortie

Adepse : booléen	Est mis à vrai pour déclencher le déplacement du système embarqué vers un bobinoir
Aretse : booléen	Est mis à vrai pour déclencher le déplacement du système embarqué vers sa position de dégagement
Aarrse : booléen	Est mis à vrai pour arrêter la course du système embarqué
Acf : booléen	Est mis à vrai pour déclencher la coupe du fil par le système embarqué.
Aech : booléen	Est mis à vrai pour déclencher l'échange de la bobine avec un tube vide par le système embarqué.
Adepn : booléen	Est mis à vrai pour déclencher le déplacement de la navette vers un bobinoir.
Aretn : booléen	Est mis à vrai pour déclencher le déplacement de la navette vers sa position de dégagement.
Aarrn : booléen	Est mis à vrai pour arrêter la course de la navette

La figure 2.37 décrit le comportement externe du système de commande d'un métier.

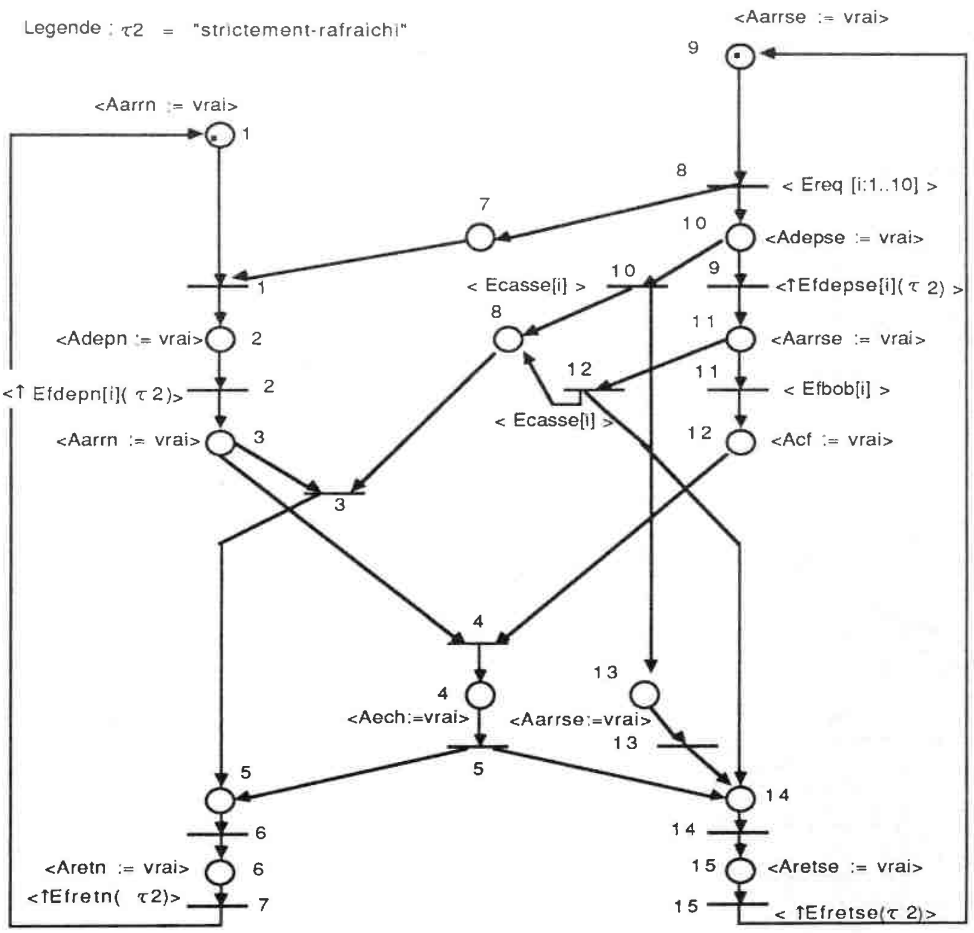
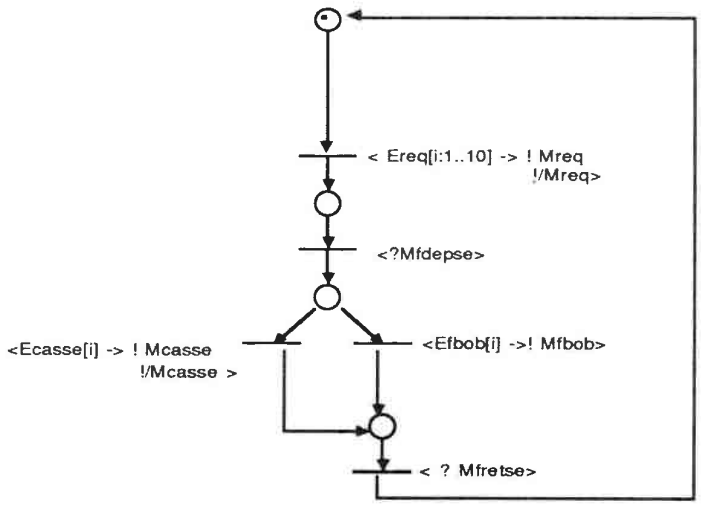


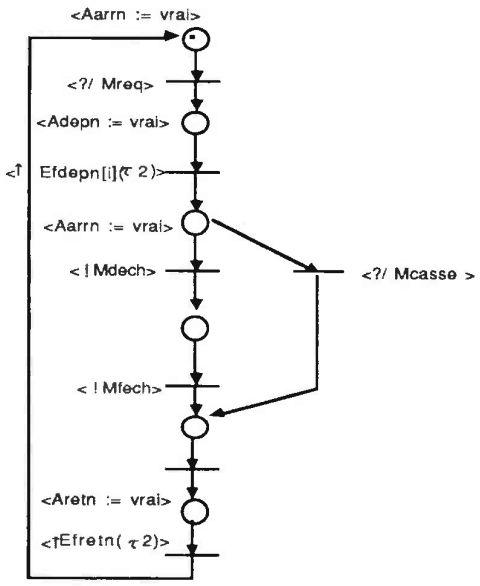
Figure 2.37

Description d'une solution répartie :

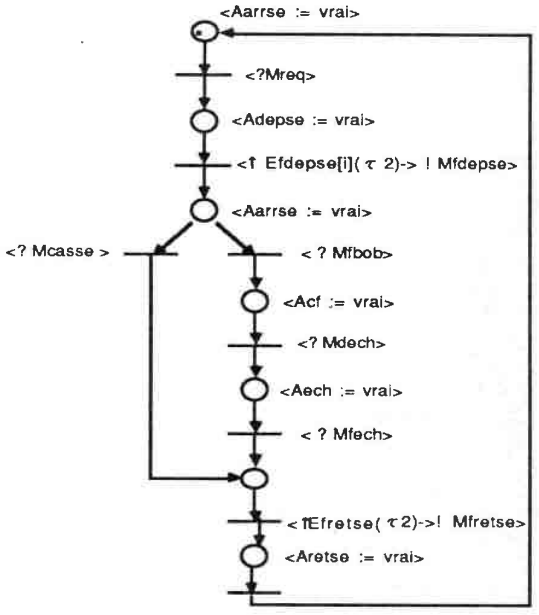
La figure 2.38 décrit le comportement interne d'un réseau de modules communicants assurant la commande d'un métier. Le réseau (a) est associé au métier, le réseau (b) au système embarqué, et le réseau (c) à la navette.



(a) Module de controle du métier



(b) Module de controle de la navette



(a) Module de controle du système embarqué

Figure 2.38

Chapitre 3

LA MODELISATION DE L'ENVIRONNEMENT

PLAN

3.0 INTRODUCTION

Nous avons présenté au chapitre 2 un modèle de description du comportement des systèmes de commande répartis. Le but de ce chapitre est de proposer une approche pour modéliser le comportement de l'environnement avec lequel ces systèmes interagissent.

La modélisation du comportement de l'environnement peut être intéressante à plusieurs points de vue. Par exemple, dans [Alanche 86], elle est utilisée pour **positionner des "filtres"** entre la partie commande et les cartes d'entrée/sortie, afin de s'assurer que les ordres engendrés par la partie commande et les événements détectés par les capteurs sont cohérents avec l'état des constituants de l'environnement (fig. 3.1).

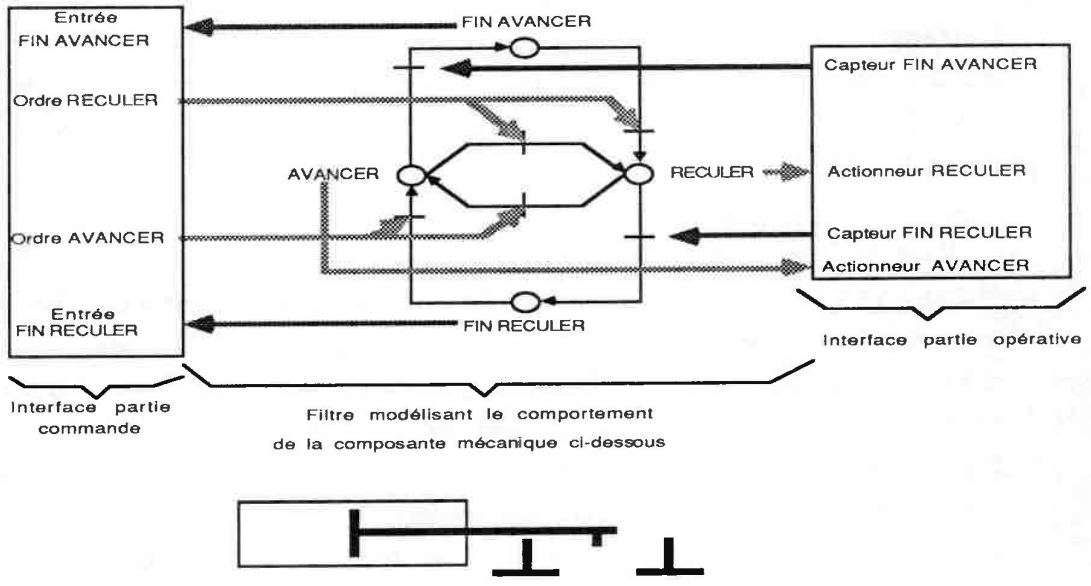


Figure 3.1

Ainsi, on peut, dans une mise en oeuvre réelle de la partie commande,

apporter de façon dynamique une réponse au problème de la sécurité des biens et des personnes, puis aussi à celui de la détection des pannes qui peuvent survenir dans la partie opérative.

Dans la même optique, on peut citer également l'approche préconisée dans [Raynal 81] pour décrire la **légalité des traces d'entrée** dans la spécification de l'interface d'accès d'un système.

Dans [Benzakour 86] par contre, la modélisation du comportement de l'environnement est utilisée pour faire hors site, une mise en oeuvre simulée de la spécification du comportement d'un système de commande, en permettant la **génération automatique des événements qui caractérisent l'évolution des grandeurs physiques de l'environnement** avec lequel ce système interagit. Dans un contexte de conception, cette mise en oeuvre simulée d'une spécification est un outil précieux pour les tests et pour le dialogue entre concepteurs et utilisateurs. Elle permet une mise au point fine des spécifications, en se comportant comme un véritable révélateur des conséquences des intentions que ces spécifications sont sensées exprimer. C'est à ce type d'utilisation de la modélisation du comportement de l'environnement que nous nous intéressons.

Dans ce qui suit, nous présentons tout d'abord les notions d'**événements endogènes** et **exogènes**, puis celles de **variables binaires fugitives** et **non fugitives** et de **variables continues**. Ensuite, en nous appuyant sur ces notions, nous proposons sept types génériques qui permettent au spécifieur d'instancier des **générateurs d'événements** de modification des variables d'entrée d'un système de commande. Les événements générés par ces générateurs permettent, au cours d'une simulation de la partie commande, de rendre compte, le plus fidèlement possible, de l'évolution des grandeurs physiques de l'environnement. Nous illustrons ensuite cette approche de modélisation du comportement de l'environnement par quelques exemples.

3.1 EVENEMENTS ENDOGENES ET EVENEMENTS EXOGENES

Les notions d'événements endogènes et d'événements exogènes que nous présentons ici permettent de différencier les événements de modification des variables d'entrée d'un système de commande par le fait que leurs occurrences peuvent être considérées comme dépendantes ou non des ordres engendrés par ce système.

Soit e un événement de modification d'une variable d'entrée d'un système de commande, rendant compte de l'évolution d'une grandeur physique G. Nous dirons que e est un événement endogène ssi l'évolution de la grandeur physique G peut être considérée comme dépendante des ordres engendrés par le système de commande. Si au contraire l'évolution de la grandeur physique G peut être considérée comme indépendante des ordres engendrés par le système de commande, nous dirons que e est un événement exogène.

Par exemple, dans l'exemple de la pompe de Kramer (c.f. section 2.6.1), le niveau de l'eau dans la galerie souterraine est une grandeur physique dont l'évolution dépend des ordres de commande que reçoit la pompe. Selon que la pompe est en état d'arrêt ou de marche, le niveau de l'eau dans la galerie souterraine augmente ou diminue. Les événements de modification de la variable d'entrée "Neau" qui rendent compte de cette évolution constituent de ce fait des exemples d'événements endogènes. Par contre, l'évolution du niveau de méthane dans l'atmosphère est une grandeur physique dont l'évolution n'est pas directement lié aux ordres engendrés par le système de commande. Les événements de modification de la variable d'entrée "Nmeth" qui rendent compte de cette évolution sont donc des exemples d'événements exogènes.

3.2 VARIABLES BINAIRES FUGITIVES, VARIABLES BINAIRES NON FUGITIVES, ET VARIABLES CONTINUES

Dans un procédé industriel, on distingue deux modes d'observation de

l'évolution d'une grandeur physique. Lors d'une observation, dans le premier mode, on s'intéresse à la détection du passage de la grandeur physique par une position nominale, tandis que dans le second mode, on s'intéresse plutôt à la position courante de la grandeur physique. Ces deux modes d'observation impliquent des capteurs de natures différentes et des variables d'entrée de types différents. Dans le premier cas, l'observation s'effectue à l'aide de capteurs "tout ou rien" dans des variables binaires servant à indiquer que la grandeur physique à atteint la position nominale considérée, et dans le second cas à l'aide de capteurs "numériques" dans des variables réelles servant à indiquer la position courante de la grandeur physique.

Nous utilisons les notions de variables binaires fugitives et non fugitives et celle de variables continues, pour différencier les variables d'entrée par le profil de variation des valeurs qu'elles prennent dans le temps.

3.3.1 VARIABLES BINAIRES FUGITIVES ET NON FUGITIVES

Soit v une variable d'entrée binaire servant à indiquer qu'une grandeur physique G a atteint une position nominale r. Nous dirons que v est une variable binaire fugitive, ssi la grandeur physique G ne peut occuper la position nominale r que pendant un temps court. Si au contraire la grandeur physique G peut occuper la position nominale r pendant un temps long, nous dirons plutôt que v est une variable binaire non fugitive. Au sens de l'automaticien, les variables binaires fugitives sont liées à la notion d'événement, et les variables binaires non fugitives à la notion d'état.

Le chronogramme de la figure 3.2 donne le profil type des variations d'une variable binaire fugitive, et le chronogramme de la figure 3.3 celui des variations d'une variable binaire non fugitive. La variable v prend la valeur vrai si la grandeur physique est à sa position nominale, et faux sinon.

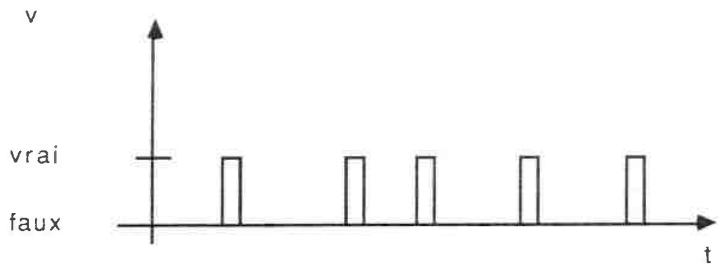


Figure 3.2

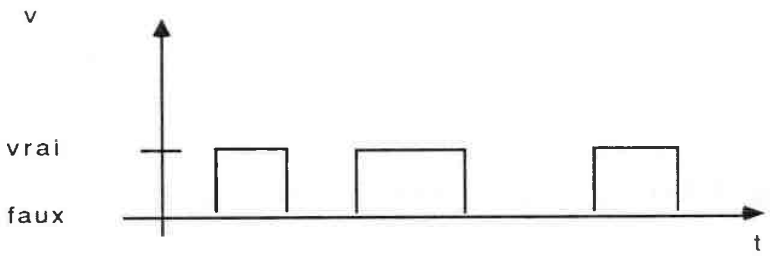


Figure 3.3

Comme exemple de variables binaires fugitives, on peut citer les variables d'entrée servant pour la détection du passage d'un chariot devant des postes de travail, et comme exemple de variables binaires non fugitives une variable d'entrée servant pour la détection de la présence d'une piéce à usiner dans un chariot.

3.3.2 VARIABLES CONTINUES

Il s'agit essentiellement de variables d'entrée servant à indiquer les positions successives des grandeurs physiques. Le chronogramme de la figure 3.4 donne le profil type des variations d'une variable continue.

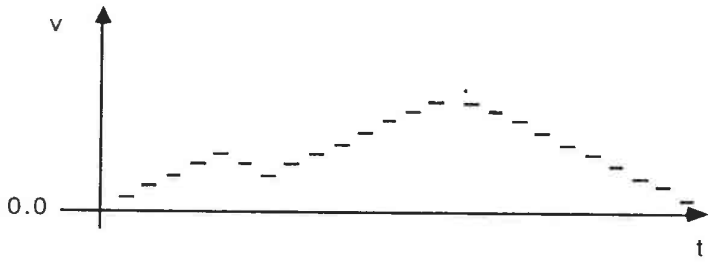


Figure 3.4

Les variables d'entrée servant à indiquer la température dans un haut fourneau, ou les variables "Neau", "Nmeth", "Nco_2", "Nair", servant à indiquer le niveau de l'eau et des gaz dans l'exemple de la pompe de Kramer (c.f. section 2.6.1), sont autant d'exemples qui illustrent cette notion de variables continues.

3.3 LES TYPES GENERIQUES DE GENERATEURS D'EVENEMENTS

Chacun des six types génériques que nous présentons ici est un modèle permettant l'instanciation d'une famille de générateurs d'événements, qui ont en commun le fait que les événements générés sont tous de même type (endogène ou exogène) et modifient le même type de variables (binaire fugitif, binaire non fugitif, ou continu). Chaque instanciation de ces types génériques permet au spécifieur de préciser les conditions d'occurrence des événements de modification d'une variable d'entrée, qui rendent compte de l'évolution d'une grandeur physique. Toutefois, dans un environnement complexe, l'occurrence d'un événement peut changer le cours normal de l'évolution de cette grandeur physique, et remettre donc en cause ces conditions. Aussi, le rôle du septième type générique sera de permettre au spécifieur de provoquer la purge d'un événement, lorsque celui-ci doit être invalidé du fait de l'occurrence d'un autre événement.

3.3.1 LES GENERATEURS D'EVENEMENTS EXOGENES BINAIRES FUGITIFS

Un générateur d'événements exogènes binaires fugitifs permet

d'engendrer selon une périodicité fixe ou aléatoire, des événements qui forcent une variable binaire fugitive à sa valeur significative le temps d'un cycle de simulation. Après le cycle de simulation, la variable concernée reprend sa valeur non significative.

Un générateur d'événements exogènes binaires fugitifs s'instancie en précisant les paramètres suivants:

-le nom d'une variable d'entrée du système de commande

-la valeur significative de cette variable

-et la loi de distribution de l'intervalle de génération des événements

3.3.2 LES GENERATEURS D'EVENEMENTS EXOGENES BINAIRES NON FUGITIFS

Un générateur d'événements exogènes binaires non fugitifs permet d'engendrer selon une périodicité fixe, des événements qui forcent une variable binaire non fugitive à vrai selon une probabilité p, et à faux selon une probabilité 1- p.

Il s'instancie en précisant les paramètres suivants:

-le nom d'une variable d'entrée du système de commande

-la probabilité avec laquelle cette variable prend la valeur vrai ou faux

-et la fréquence de génération des événements modifiant cette variable

3.3.3 LES GENERATEURS D'EVENEMENTS EXOGENES CONTINUS

Un générateur d'événements exogènes continus permet d'engendrer selon

une périodicité fixe, des événements qui font varier le contenu d'une variable continue d'un Δx qui peut être fixe ou aléatoire.

Il s'instancie en précisant les paramètres suivants:

-le nom d'une variable d'entrée du système de commande

-la loi de distribution des Δx correspondant aux variations de cette variable

-et la fréquence de génération des événements modifiant cette variable

3.3.4 LES GENERATEURS D'EVENEMENTS ENDOGENES BINAIRES FUGITIFS

A l'inverse des générateurs d'événements exogènes, les générateurs d'événements endogènes sont soumis à des conditions de génération qui portent sur l'état du système de commande. Un générateur d'événements endogènes binaires fugitifs permet, chaque fois que la condition de génération qui lui est associée devient vrai, de générer avec une probabilité p un événement de modification de variable binaire fugitive. La variable v_i à modifier peut être choisie avec une probabilité p_i parmi n variables. La somme des probabilités associées aux n variables doit alors être égale à p. Chaque événement généré force une variable à sa valeur significative au bout d'un temps Δt qui peut être fixe ou aléatoire. Après un cycle de simulation, la variable modifiée reprend sa valeur non significative.

Un générateur d'événements endogènes binaires fugitifs s'instancie en précisant

- l'état dans lequel doit se trouver le système de commande pour qu'un événement soit généré (c'est la condition de génération)

et pour chaque variable d'entrée du système de commande pouvant être modifiée

-son nom

-sa valeur significative

-la probabilité pour qu'un événement généré la concerne

-et la loi de distribution du temps Δt au bout duquel cet événement provoque sa modification

3.3.5 LES GENERATEURS D'EVENEMENTS ENDOGENES BINAIRES NON FUGITIFS

Les générateurs d'événements endogènes binaires non fugitifs se différencient des générateurs d'événements endogènes binaires fugitifs uniquement par le fait que les événements engendrés ne modifient pas les variables d'entrée que le temps d'un cycle de simulation. Les variables modifiées conservent leur modification après le cycle de simulation.

Un générateur d'événements endogènes binaires non fugitifs s'instancie en précisant :

-l'état dans lequel doit se trouver le système de commande pour qu'un événement soit généré (c'est la condition de génération)

et pour chaque variable d'entrée du système de commande pouvant être modifiée

-son nom

-la valeur qu'elle prend

-la probabilité pour qu'un événement généré la concerne

-et la loi de distribution du temps Δt au bout duquel cet événement provoque sa modification

3.3.6 LES GENERATEURS D'EVENEMENTS ENDOGENES CONTINUS

Les générateurs d'événements endogènes continus sont caractérisés d'une part par une condition d'activation et d'autre part par une condition d'arrêt de la génération, qui portent sur l'état du système de commande. Chaque fois que la condition d'activation qui lui est associée devient vraie, un générateur d'événements endogènes continus engendre, selon une périodicité fixe, des événements qui font varier le contenu d'une variable continue d'un Δx fixe ou aléatoire, jusqu'à ce que la condition d'arrêt devienne vraie. Le contenu de la variable d'entrée peut être forcée au début de chaque cycle de génération à une valeur particulière.

Un générateur d'événements endogènes continus s'instancie en précisant:

-le nom d'une variable d'entrée du système de commande

-l'état dans lequel doit se trouver le système de commande pour que la génération débute (c'est la condition d'activation de la génération)

-l'état dans lequel doit se trouver le système de commande pour que la génération s'arrête (c'est la condition d'arrêt de la génération)

-la valeur de départ de la variable (optionnel)

-la loi de distribution des Δx correspondant aux variations de la variable

-et la fréquence de génération des événements

3.3.7 LES GENERATEURS DE PURGES

Un générateur de purges permet, chaque fois qu'un événement donné est généré, de provoquer la purge de tous les événements que son occurrence invalide. La date prévue pour l'occurrence d'un événement qui fait l'objet d'une purge doit être supérieure à la date prévue pour l'occurrence de l'événement qui provoque sa purge. Un générateur de purge s'instancie en précisant les paramêtres suivants:

-le nom de la variable d'entrée dont les événements de modification provoquent des purges

-et la liste des variables d'entrée dont les événements de modification doivent être purgés

3.4 EXEMPLES

Les deux exemples que nous présentons ici montrent comment on peut utiliser les types génériques proposés pour instancier des générateurs d'événements modélisant le comportement d'un environnement en interaction avec un système de commande. Ces exemples s'appuient sur les deux applications déjà présentées en 2.6.

3.4.1 MODELISATION DU COMPORTEMENT DE L'ENVIRONNEMENT DU SYSTEME DE COMMANDE DE LA POMPE DE KRAMER

Pour simplifier l'exposé, nous allons nous restreindre au RdPI de la figure 2.27. Il décrit comment la mise en route et l'arrêt de la pompe doivent alterner en fonction de l'évolution de deux grandeurs physiques : le niveau de l'eau et le taux de méthane dans la galerie souterraine.

Comme nous l'avions déjà vu, l'évolution du taux de méthane dans la galerie n'est pas directement liée aux ordres engendrés par la partie commande,

alors qu'à l'inverse, le sens de l'évolution du niveau de l'eau dépend de l'état de fonctionnement de la pompe. Les événements de modification de la variable "Nmeth" qui rendent compte de l'évolution du méthane peuvent de ce fait être engendrés par un générateur d'événements exogènes continus, instancié avec les paramètres suivants:

-Nom de la variable d'entrée: Nmeth

-Fréquence de génération des événements de modification de la variable : toutes 5 unités de temps

-Loi de distribution des Δx correspondant à la variation du taux de méthane toutes les 5 unités de temps: uniformément distribué entre -0.2 et +0.2.

Quant aux événements de modification de la variable d'entrée "Neau" qui servent à rendre compte de l'évolution du niveau de l'eau, ils peuvent être engendrés par deux générateurs d'événements endogènes binaires continus instanciés de la façon suivante

1er générateur:

-Nom de la variable d'entrée : Neau

-Condition d'activation de la génération d'événements : $M(p_2) = 1$

-Condition d'arrêt de la génération d'événements : $M(p_2) = 0$

-Fréquence de génération des événements de modification de la variable : toutes les 5 unités de temps

-Loi de distribution des Δx correspondant à la baisse du niveau de l'eau toutes les 5 unités de temps : variation constante de - 0.45.

2ème générateur:

-Nom de la variable d'entrée : Neau

- Condition d'activation de la génération d'événements : $M(p_1) = 1$

-Condition d'arrêt de la génération d'événements : $M(p_1) = 0$

-Fréquence de génération des événements de modification de la variable : toutes les 5 unités de temps

-Loi de distribution des Δx correspondant à l'augmentation du niveau de l'eau toutes les 5 unités de temps : uniformément distribuée entre -0.1 et +0.2.

Le premier générateur engendre les événements qui rendent compte de la baisse du niveau de l'eau quand la pompe est en état de fonctionnement, et le deuxième générateur les événements qui rendent compte de l'augmentation du niveau de l'eau quand la pompe s'arrête de fonctionner.

3.4.2 MODELISATION DU COMPORTEMENT DE L'ENVIRONNEMENT DU SYSTEME DE COMMANDE DE L'ATELIER DE BOBINAGE

Dans cet exemple également, pour simplifier l'exposé, nous allons nous restreindre à deux grandeurs physiques: la position du système embarqué, et la taille atteinte par une bobine en cours de bobinage sur un bobinoir n.

La variation de la position du système embarqué dépend des ordres engendrés par la partie commande. En plus, pour l'observation de l'évolution de cette grandeur physique, on ne s'intéresse qu'aux passages du système embarqué devant un bobinoir. Cette observation s'effectue dans les variables d'entrée du tableau "Efdepse". De ce fait, pour chaque variable d'entrée "Efdepse[n]", on peut engendrer les événements qui modifient son contenu en

utilisant un générateur d'événements endogènes binaires fugitifs instancié avec les paramètres suivants:

-Nom de la variable d'entrée : Efdepse[n]

-Condition de génération d'un événement : $M(p_{10}) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 1

-Valeur significative de la variable d'entrée : vrai

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 5 * n unités.

Autrement dit, chaque fois que le système de commande engendre un ordre de déplacement du système embarqué, si le bobinoir à traiter est le bobinoir n, alors un événement est généré pour signaler le passage du système embarqué devant ce bobinoir au bout de 5*n unités de temps.

Pour ce qui concerne la taille d'une bobine en cours de bobinage sur un bobinoir n, on n'observe que le fait qu'elle a atteint une position nominale nécessitant une requête du système embarqué, et le fait qu'elle a atteint une position nominale nécessitant la coupe du fil. Dans le premier cas, l'observation se fait dans la variable d'entrée "Ereq[n]", et dans le second cas dans la variable d'entrée "Efbob[n]". On peut engendrer les événements de modification de ces deux variables à l'aide de générateurs d'événements endogènes binaires non fugitifs. Ces générateurs peuvent être instanciés de la façon suivante:

Cas de la variable "Ereq[n]", passage à faux

-Nom de la variable d'entrée : Ereq[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 1

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 0 unité.

Cas de la variable "Ereq[n]", passage à vrai

-Nom de la variable d'entrée : Ereq[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 1

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 1000 unités.

Cas de la variable "Efbob[n]", passage à faux

-Nom de la variable d'entrée : Efbob[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 1

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 0 unité.

Cas de la variable "Efbob[n]", passage à vrai

-Nom de la variable d'entrée : Efbob[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 1

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 1200 unités

Les générateurs d'événements que nous avons identifiés jusqu'ici ne précisent que le comportement normal de l'environnement par rapport aux deux grandeurs que nous avons retenues. Pour être complet, il faut aussi prendre en compte les incidents de fonctionnement qui correspondent ici aux casses de fil. Pour un bobinoir n, les casses de fil sont observées dans la variable d'entrée Ecasse[n]. Les événements qui modifient cette variable peuvent être engendrés par un générateur d'événements endogènes binaires non fugitifs instancié avec les paramètres suivants:

Cas du passage à faux

-Nom de la variable d'entrée : Ecasse[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 0.3

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : temps constant de 0 unités.

Cas du passage à vrai

-Nom de la variable d'entrée : Ecasse[n]

-Condition de génération d'un événement : $M(p_4) = 1 \wedge n = i$

-Probabilité de génération d'un événement : 0.3

-Loi de distribution du temps Δt au bout duquel les événements générés modifient la variable d'entrée : uniformément distribué entre 5 et 1200 unités de temps.

En d'autres termes, nous avons spécifié que chaque échange de bobine sur un bobinoir n doit provoquer pour ce bobinoir, la génération d'une requête du système embarqué, d'une indication de fin de bobinage, et dans 30% des cas une génération de casse de fil. Il est bien évident que chaque fois qu'un événement de casse de fil sera effectivement généré, il pourra remettre en cause les indications de fin de bobinage et de fin de course du système embarqué. Notre modélisation doit de ce fait être complétée par une instanciation de générateur de purges en précisant les paramètres suivants:

-Nom de la variable d'entrée dont les événements de modification provoque une purge: Ecasse[n]

-Liste des variables d'entrée dont les événements de modification doivent être purgés: Efbob[n], Efdepse[n]

3.4.3 REMARQUES

Comme on peut le voir à travers les deux exemples présentés, les principes de modélisation du comportement de l'environnement d'un système de commande sont simples. Le spécifieur identifie dans un premier temps les grandeurs physiques de l'environnement qui doivent être observées par le système de commande. Ensuite, il instancie les générateurs d'événements qui permettent de rendre compte le plus fidèlement possible de l'évolution de ces grandeurs physiques.

La structure du RdPI spécifiant le comportement du système de commande peut aussi guider le spécifieur. Dans ce cas, pour chaque groupe de transitions en conflit structurel, il instancie un générateur des événements qui peuvent provoquer le tir de ces transitions.

Chapitre 4

LA MODELISATION DU FLUX D'ACTIVITE

PLAN

4.0 INTRODUCTION

Du fait de l'enjeu économique que cela implique, l'**évaluation des performances** constitue un problème fondamental dans le cycle de vie d'un atelier de production. Ce problème se pose en général dès les premières investigations, mais peut aussi se poser dans des étapes très avancées du processus de conception, voire encore plus tard en phase d'exploitation. Il permet de déterminer les meilleurs compromis pour **réduire les encours et gérer de façon optimale l'utilisation des ressources, en fonction de la production à assurer**. Dans l'état actuel des connaissances, la simulation constitue l'un des meilleurs moyens pour résoudre de façon précise ce problème.

Dans les sections 4.1 et 4.2 de ce chapitre, nous présentons dans les grandes lignes les approches habituellement utilisées pour aborder l'évaluation des performances des ateliers de production à l'aide de la simulation. Ces approches peuvent être groupées en deux grandes familles: celles s'appuyant sur les langages orientés simulation, et celles s'appuyant sur les RdP.

Ensuite, nous présentons dans la section 4.3 l'approche que nous adoptons. Elle se fonde sur un modèle de description du flux d'activité, qui complète les modèles décrits aux chapitres 2 et 3, et permet au cours d'une simulation conjointe de la partie commande et de la partie opérative d'un atelier de production, de collecter les mesures nécessaires pour l'évaluation des performances.

Nous terminons ce chapitre en consacrant la section 4.4 à la présentation de quelques exemples d'illustration de cette approche.

4.1 L'EVALUATION DES PERFORMANCES A L'AIDE DES LANGAGES ORIENTES SIMULATION DISCRETE (GPSS, PAWS, Etc...)

Les langages de programmation orientés simulation discrète permettent de décrire de façon macroscopique les aspects pertinents du fonctionnement d'un atelier, en confondant partie commande, partie opérative, et flux d'activité, puis d'obtenir au cours d'une simulation du fonctionnement décrit, des statistiques fines sur les files d'attentes : longueur maximum, longueur minimum, temps d'attente, débit, taux d'occupation. Ils s'appuient en général pour cela sur

- des concepts de haut niveau (entités actives ou passives, activités, événements, files d'attente etc ....)

- des règles prédéfinies pour la gestion des files d'attente

- et des lois de distribution des intervalles d'arrivée des événements, des temps de service et des routages

qui permettent au spécifieur de formaliser, le plus simplement possible et à moindre coût, sa perception de l'évolution du monde réel.

Les inconvénients majeurs de ce type de langages se résument en quatre points essentiels :

1°) Ils sont malcommodes pour décrire finement les contraintes de synchronisation qui caractérisent le fonctionnement d'un atelier flexible.

2°) En plus, ils ne permettent pas de vérifier formellement les propriétés de ces contraintes.

3°) Par ailleurs, d'un point de vue méthodologique, le fait de ne pas séparer dans la description du fonctionnement d'un atelier la partie commande, la partie

opérative et le flux d'activité, ne rend pas aisé l'évaluation de différents scénarios.

4°) Enfin et surtout, des risques d'incohérence peuvent exister entre la logique du fonctionnement simulé et celle décrite dans la spécification du système de commande.

4.2 L'EVALUATION DES PERFORMANCES A L'AIDE DES SIMULATEURS DE RdP

Ces simulateurs [Alanche 84] [Valette 85 b] [Martin 87] [Hollinger 87] conduisent à décrire le fonctionnement d'un atelier de production à l'aide d'un RdP auquel on associe une interprétation. L'interprétation associée aux places permet de prendre en compte au cours d'une simulation la durée d'exécution des opérations, et celle associée aux transitions de resoudre les conflits de franchissement éventuellement sur des bases stochastiques.

L'intérêt de cette approche réside essentiellement dans le fait que les RdP permettent de décrire et de valider très aisément les contraintes de synchronisation qui caractérisent le fonctionnement des ateliers flexibles. Par ailleurs, comme nous l'avons déjà vu au chapitre 2, ce formalisme peut être utilisé pour la description du comportement du système de commande, ce qui permet, dans une démarche de conception, de maintenir une cohérence entre les différents modèles utilisés.

4.3 L'APPROCHE ADOPTEE

L'originalité de cette approche réside essentiellement dans le fait qu'elle permet de fonder la simulation sur trois descriptions distinctes mais complémentaires de la dynamique des ateliers de production :

-la description du flux d'activité
-La description du comportement du système de commande

-et la description du comportement de l'environnement

La description du flux d'activité permet de fixer les contraintes imposées par l'organisation du réseau de transport, des stocks intermédiaires, et des machines, en précisant les mouvements physiquement et logiquement possibles, puis d'en déduire au cours d'une simulation, des statistiques pertinentes pour l'évaluation des performances. La spécification du comportement du système de commande définit les stratégies de pilotage de l'atelier en précisant les règles d'enchainement des décisions à prendre pour coordonner le déroulement des différentes activités. La description du comportement de l'environnement quant à elle précise, en fonction des caractéristiques des composantes mécaniques, les lois qui régissent l'évolution des grandeurs physiques.

D'un point de vue méthodologique, cette approche offre en effet plusieurs avantages:

1°) Pour chacune de ces descriptions, on peut utiliser le formalisme le mieux adapté.

2°) Ces trois descriptions n'étant pas redondantes, le spécifieur peut changer l'une d'entre elle sans remettre en cause les deux autres. Ainsi, étant donné un ensemble de lois caractéristiques de l'évolution des grandeurs physiques et une organisation de l'atelier, le spécifieur peut évaluer différentes stratégies de pilotage, ou encore pour une stratégie de pilotage et une organisation, évaluer différentes hypothèses sur les lois d'évolution des grandeurs physiques.

3°) Enfin, par rapport à la spécification du comportement du système de commande, la description du flux d'activité offre une vue plus clarifiée des contraintes du fonctionnement que le système de commande doit assurer. Au cours d'une simulation, il est aisé de vérifier que l'évolution de la commande respecte bien ces contraintes. Inversement, le spécifieur est toujours assuré que le fonctionnement de l'atelier qu'il évalue respecte bien les contraintes exprimées dans la spécification du système de commande.

Le but de cette section est de présenter le modèle de description du flux d'activité. Ce modèle conduit à décrire le flux d'activité par un réseau de files d'attente, constitué comme dans un RdP classique d'un ensemble de places et d'un ensemble de transitions, reliées entre elles par des arcs orientés. Dans ce qui suit, nous précisons le rôle des marques, des places et des transitions dans ce modèle. Ensuite, nous comparons le formalisme proposé aux RdP, ce qui nous permettra de mieux préciser son intérêt.

4.3.1 LE ROLE DES MARQUES DANS LE MODELE DE DESCRIPTION DU FLUX D'ACTIVITE

Dans un réseau décrivant le flux d'activité, toute marque apparaîssant dans une place est **individualisée** et sert à modéliser une transaction caractérisée par quatre types d'attributs:

- Un identifiant de transaction (ID)
- Une priorité (PR)
- une classe (CL)
- et l'identifiant de la file d'attente à laquelle appartient la transaction (FL)

4.3.2 LE ROLE DES PLACES

Chaque place du réseau sert à modéliser une ou plusieurs files d'attente, et est caractérisée par la politique de gestion de ces files (PG), et leur borne supérieure (BS).

La politique FIFO (First In First Out) permet de gérer les transactions selon leur ordre d'arrivée, la politiques LIFO (Last In First Out) selon l'ordre inverse de leur arrivée, et la politique RAND dans un ordre quelconque.

Si la borne supérieure d'une file d'attente est inconnue, nous noterons par convention qu'elle est infinie (i.e BS = ∞).

Le but de la modélisation du flux d'activité est de permettre la collecte de

statistiques fines sur ces files d'attente : longueur maximum, longueur minimun, temps de transit, débit, taux d'utilisation.

4.3.3 LE ROLE DES TRANSITIONS

Chaque transition t_i d'un réseau correspond à un ensemble E_i de relations d'écoulement du flux d'activité (noté par convention par FA). Ces relations sont définies entre les files d'attente apparaîssant en entrée et en sortie de la transition. Chaque écoulement de flux retranche des transactions des files d'entrée, et ajoute des transactions aux files de sortie.

A un instant donné, les transactions qui participent en entrée dans un écoulement de flux sont déterminées par rapport à des identifiants de files. Dans chacune de ces files, une transaction est choisie en fonction de la politique de gestion qui lui est associée (FIFO, LIFO, RAND). Dans le cas de la politique RAND, ce choix peut être contraint par le fait que la transaction sélectionnée doit avoir le même identifiant que les transactions sélectionnées dans d'autres files. Ces critères de sélection peuvent être renforcés par la classe, si en plus l'on souhaite vérifier la validité des transactions sélectionnées.

Pour la désignation de ces différents critères de sélection, deux cas sont à considérer:

a) Les critères de sélection définis par les fonctions associés aux arcs

Si t_i est une transition représentant un ensemble E_i de relations d'écoulement, dans ce premier cas on étiquette chaque arc (p_j,t_i) de la transition t_i par une fonction

$$\varphi_{ij} : E_i \rightarrow \text{domaine(FL)} \ [\ \text{x domaine(CL)}\ ]$$

qui associe à chaque relation d'écoulement $e_k \in E_i$ un identifiant de file dans p_j et éventuellement une classe de transaction. Par défaut, les arcs sont étiquetés

par la fonction identité. Pour une relation d'écoulement c_k, ces fonctions définissent donc les files d'attente où les transactions doivent être sélectionnées, et éventuellement la classe d'appartenance de ces transactions.

Dans la figure 4.1 par exemple, la transition t_1 représente six relations d'écoulement de flux désignées par : $(stock_1,c_1)$, $(stock_2,c_1)$, $(stock_3,c_1)$, $(stock_1,c_2)$, $(stock_2,c_2)$, $(stock_3,c_2)$. Ces relations d'écoulement sont définies par l'ensemble E1. A partir de l'étiquetage des arcs (p_1,t_1) et (p_2,t_1), on peut déduire qu'une relation d'écoulement $(stock_i,c_j)$ utilise une transaction dans la file $stock_1$ de la place p_1 (c.f. définition de la fonction $\varphi 0$), et une transaction dans la file $stock_i$ de la place p_2 (l'arc (p2,t1) est étiquetée par la fonction identité). Dans les deux cas, la transaction sélectionnée doit appartenir à la classe c_j.

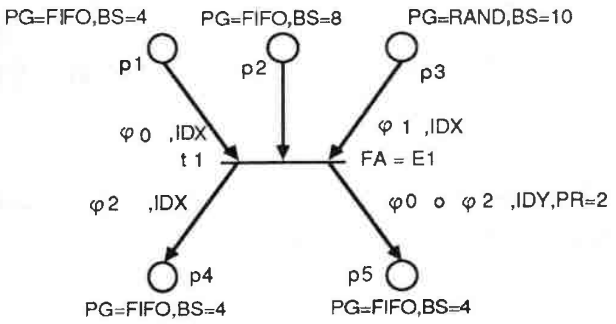


Figure 4.1

b) Les critères de sélection définis par les étiquettes associées aux arcs

Cette deuxième méthode de désignation des transactions en entrée d'une relation d'écoulement de flux consiste à associer à plusieurs arcs entrant une étiquette commune. Par convention, nous retiendrons que la syntaxe de cette étiquette est 'ID' suivi d'une lettre quelconque. De cette façon, on indique qu'à chaque sélection, les identifiants des transactions sélectionnées dans les files

d'attente liées aux arcs considérées doivent toujours être égaux entre eux. Si dans les files considérées, il existe à la fois des files FIFO ou LIFO et RAND, les transactions doivent être sélectionnées dans les files RAND de manière à ce que leur identifiant soit le même que celui des transactions en tête ou en queue des files FIFO ou LIFO.

Si nous revenons à l'exemple précédent de la figure 4.1, l'étiquetage de l'arc (p_3, t_1) permet de déduire qu'une relation d'écoulement de flux $(stock_i, c_j)$ utilise en plus une transaction dans la file $stock_1$ de la place p_3 (c.f. définition de la fonction $\varphi 1$ associée à l'arc (p_3, t_1)). L'arc (p_3, t_1) ayant une étiquette commune avec l'arc (p_1, t_1), et la politique de gestion des files associées à la place p_3 étant RAND, cette transaction doit être choisie de manière à ce que son identifiant soit égal à l'identifiant de la transaction sélectionnée en tête de la file $stock_1$ de la place p_1.

c) Extension des deux méthodes de désignation pour la caractérisation des transactions à produire

En utilisant la même notation, on peut définir les caractéristiques des attributs des transactions à produire dans les files de sortie lors d'un écoulement de flux :

Les fonctions associées aux arcs en sortie d'une transition définissent en fonction de la relation d'écoulement la classe des transactions à produire et l'identifiant des files où ces transactions doivent être produites.

Les étiquettes 'ID_' associées aux arcs en sortie permettent de déterminer les identifiants des transactions à produire dans les files de sortie qui leur sont associées. Si une étiquette apparaîssant sur un arc de sortie apparaît également sur un arc en entrée de la même transition, alors l'identifiant de la transaction en sortie prend la même valeur que l'identifiant de la transaction sélectionnée en entrée. Si au contraire l'étiquette associée à un arc de sortie n'apparaît sur

aucun arc d'entrée de la même transition, un nouvel identifiant est associé à toute transaction à produire dans la file de sortie correspondante. Si cette étiquette apparaît sur d'autres arcs de sortie, le même identifiant est chaque fois affecté à toutes les transactions à produire dans les différentes files de sortie correspondantes.

Enfin, on peut en plus associer à un arc de sortie la priorité qui doit être affectée aux transactions, la priorité par défaut étant 0.

Dans l'exemple de la figure 4.1, l'étiquetage de l'arc (t_1,p_4) permet de déduire qu'une relation d'écoulement de flux $(stock_i,c_j)$ produit dans la file $stock_i$ de la place p_4 une transaction de priorité 0 et de classe c_{j+1} ayant le même identifiant que la transaction sélectionnée dans la file $stock_1$ de la place p_1. De la même façon, l'étiquetage de l'arc (t_1,p_5) permet de déduire qu'une relation d'écoulement de flux $(stock_i,c_j)$ produit dans la file $stock_1$ de la place p_5 une transaction de priorité 2 et de classe c_{j+1} ayant un nouvel identifiant .

4.3.4 SYNCHRONISATION DU FLUX D'ACTIVITE A L'EVOLUTION DE LA COMMANDE

Comme nous l'avons déjà indiqué, le but essentiel de la modélisation du flux d'activité est de permettre la collecte de statistiques lors de la simulation conjointe de la partie commande et de la partie opérative d'un atelier. Pour cela, il est indispensable de préciser en plus les liens qui doivent exister entre les écoulements de flux possibles dans le réseau de files d'attente, et les événements de la partie commande. Ces liens sont précisés en associant à chaque transition du réseau de files d'attente, des transitions et/ou des événements apparaîssant dans le réseau de la partie commande. Par convention, nous noterons cette association par LN.

4.3.5 EVALUATION DU MODELE PAR RAPPORT AUX RdP

Le modèle que nous venons de présenter pour la description du flux d'activité est très voisin des RdP colorés. La différence tient essentiellement au fait que, dans ce modèle, les marques sont traitées dans les places comme dans des files d'attente. Il en résulte deux avantages : d'une part la cohérence entre les différents formalismes que nous utilisons, et d'autre part la clarté et la concision qu'apportent les descriptions à l'aide de files d'attente.

A titre d'illustration, la figure 4.2 représente à l'aide de ce modèle une file d'attente de taille 3, gérée selon la politique FIFO, et où peuvent transiter deux types de transactions (c et c'). La représentation de la même file à l'aide d'un RdP ordinaire et d'un RdP coloré figure en annexe A, chapitre A.7. Comme on peut le constater, la spécification d'une file d'attente à l'aide d'un RdP conduit à spécifier distinctement les contraintes de progression à chaque position de la file. Il s'ensuit que dans le réseau, les places n'ont pas toutes la même signification. Elles servent soit à représenter une position dans la file, soit au contrôle de la progression dans la file. Ceci est de nature à réduire la lisibilité de la spécification d'un réseau de files d'attente.

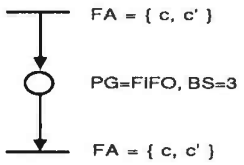


Figure 4.2

4.4 EXEMPLES

Nous illustrons la démarche de modélisation du flux d'activité à l'aide de trois exemples. Pour chacun de ces exemples, nous exposons d'abord le problème en termes de dimensionnement avant d'aborder sa modélisation.

4.4.1 L'EXEMPLE DE L'ATELIER DE BOBINAGE (c.f. section 2.6.2)

Dans cet exemple, si le nombre de bobinoirs par métier est trop important, les systèmes embarqués ne seront pas en mesure de répondre à temps aux requêtes de coupe de fil. A l'inverse, si le nombre de bobinoirs par métier est très faible, les systèmes embarqués seront sous utilisés. Le problème ici est donc de déterminer le nombre de bobinoirs à prévoir par métier pour éviter une sous exploitation des systèmes embarqués, tout en garantissant des délais raisonnables pour le traitement des bobines qui arrivent à leur position nominale. La description du flux d'activité de la figure 4.3 permet de résoudre ce problème. Au cours d'une simulation conjointe de la partie commande et de la partie opérative d'un métier, elle permet d'obtenir des statistiques sur le temps de réponse du système embarqué aux requêtes de coupe de fil. La place pp1 modélise une file d'attente de taille 1 pour chaque bobinoir. Lorsqu'une bobine arrive à sa position nominale sur un bobinoir i (ce qui correspond à l'occurence de l'événement ↑ Efbob[i]), une transaction est introduite dans la file i en franchissant la transition tt1(i). La prise en compte de cette nouvelle situation dans la partie commande par le tir de la transition t_{11}(Efbob[i]) provoque le retrait de la transaction de la place pp1.

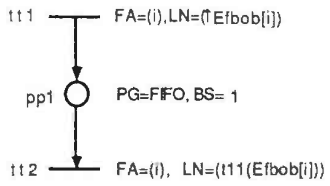


Figure 4.3

4.4.2 L'EXEMPLE DU CONVOYEUR

Cet exemple [Alla 86] concerne une section de travail que l'on peut retrouver dans divers types d'ateliers. Cette section permet l'usinage de trois types de pièces nécessitant trois types de postes de travail différents. On souhaite disposer les postes de travail de part et d'autre d'un convoyeur central qui sert à

l'acheminement des pièces (fig 4.4). Pour éviter la congestion de ce convoyeur central, des stocks intermédiaires sont prévus au niveau de chaque poste. Le problème pour ce type d'organisation est double :

- Déterminer le nombre de postes de travail de chaque type, et la taille des stocks intermédiaires, qui permettent d'éviter tout ralentissement des autres sections situées en amont et en aval, pour un ordonnancement des pièces en entrée.

- Garantir un bon niveau du taux d'utilisation des postes.

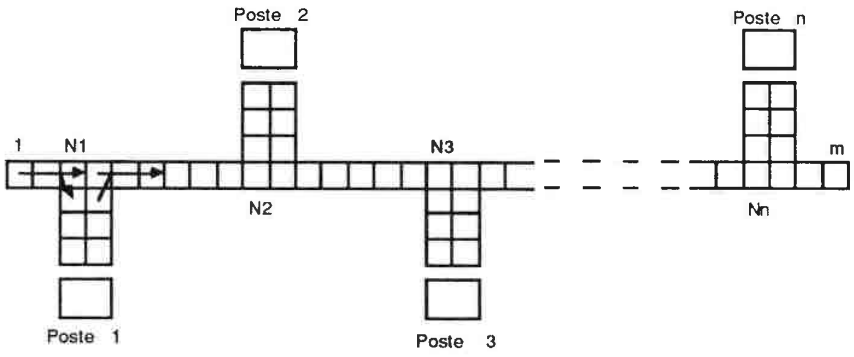


Figure 4.4

Ce problème peut être résolu par une série de simulations du fonctionnement de l'atelier. Sans faire au départ d'hypothèse sur la taille des stocks intermédiaires, on recherche le point où la diminution du nombre de postes n'influe plus sur l'amélioration de leur taux d'utilisation, et où le temps de transit des pièces dans la section commence à croître de façon sensible. On obtient ainsi non seulement le nombre de postes de travail à prévoir, mais aussi la taille des stocks intermédiaires qui correspond à la plus grande taille observée pendant la simulation.

La figure 4.5 décrit le fonctionnement de la section à l'aide d'un RdP coloré, en confondant partie commande et partie opérative [Alla 86] :

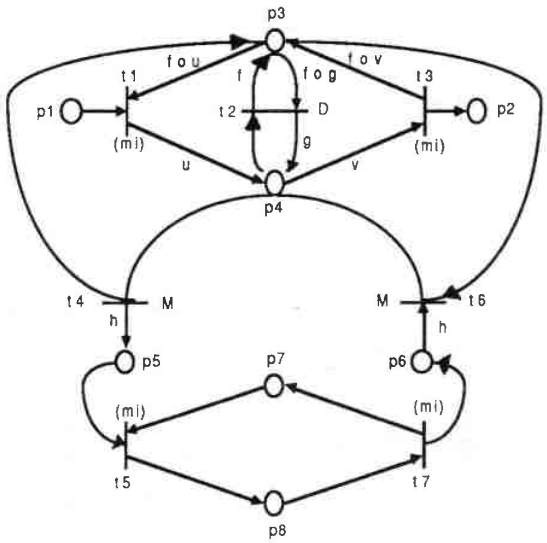
- Les couleurs m_i servent à différencier les postes d'usinage et les pièces (par rapport aux postes où elles doivent être usinées).

- Les couleurs e_j identifient les différentes portions du convoyeur.

- Les places p_3, p_4 et les transitions t_1, t_2, t_3 décrivent les règles de synchronisation liées à l'évolution des pièces sur le convoyeur selon le principe FIFO (cf. section A.7).

- Les places p_5 et p_6 modélisent les stocks intermédiaires des postes de travail. Ces stocks sont de capacité inconnue à priori.

- Les marques dans la place P_8 servent quant à elles à modéliser des postes en cours d'usinage, et les marques dans la place p_7 des postes en attente.



MARQUAGE INITIAL

$M(p3) = \sum_{j=1}^{m} e_j$ $M(p7) = \sum_{i=1}^{n} m_i$

ENSEMBLE DES COULEURS

$\{ m_i, e_j, <m_i, e_j> \}$ $\forall i \in (1,n)$ $j \in (1,m)$

$C = \{ <m_i, e_j> \}^{\forall i \in (1,n)\ j \in (1,m)}$

$M = \{ <m_i, e_{Ni}> \}^{\forall i \in (1,n)}$

$D = C - M$

DEFINITION DES FONCTIONS

$f(m_i,e_j) = e_j$

$u(m_i) = m_i, e_1$

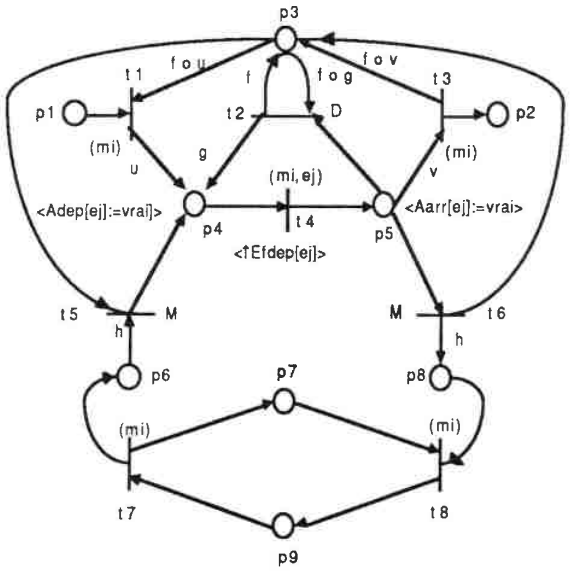
$g(m_i,e_j) = m_i,e_{j+1}$

$v(m_i) = m_i, e_m$

$h(m_i,e_j) = m_i$

Figure 4.5

Ce réseau peut être transformé de façon très simple en une description du comportement du système de commande (c.f. figure 4.6), et complété par une description du flux d'activité (c.f. figure 4.7). Dans la figure 4.7, pour chaque type de pièces, une file FIFO est associée à chaque place. Les places pp3, (pp2, pp4), et pp6 permettent ainsi d'obtenir des statistiques respectivement sur le taux d'utilisation des postes de travail, la taille des stocks intermédiaires, et le délai de transit des pièces.



MARQUAGE INITIAL

$M(p3) = \sum_{j=1}^{m} ej \qquad M(p7) = \sum_{i=1}^{n} mi$

ENSEMBLE DES COULEURS

{ mi, ej, <mi, ej> } $\forall i \in (1,n) \quad j \in (1,m)$

$C = \{ \langle mi, ej \rangle \}^{\forall i \in (1,n) \quad j \in (1,m)}$

$M = \{ \langle mi, eNi \rangle \} \forall i \in (1,n)$

$D = C - M$

DEFINITION DES FONCTIONS

f (mi,ej) = ej
u (mi) = mi, e1
g (mi,ej) = mi,ej+1
v (mi) = mi, em
h (mi,ej) = mi

VARIABLES D'ENTREE

Efdep : tableau [1..m] de booléen — Efdep[i]=vrai si fin de déplacement sur la portion i du convoyeur

VARIABLES DE SORTIE

Adep : tableau [1..m] de booléen — Adep[i]:=vrai déclenche un déplacement sur la portion i
Aarr : tableau [1..m] de booléen — Aarr:=vrai déclenche un arrêt sur la portion i

Figure 4.6

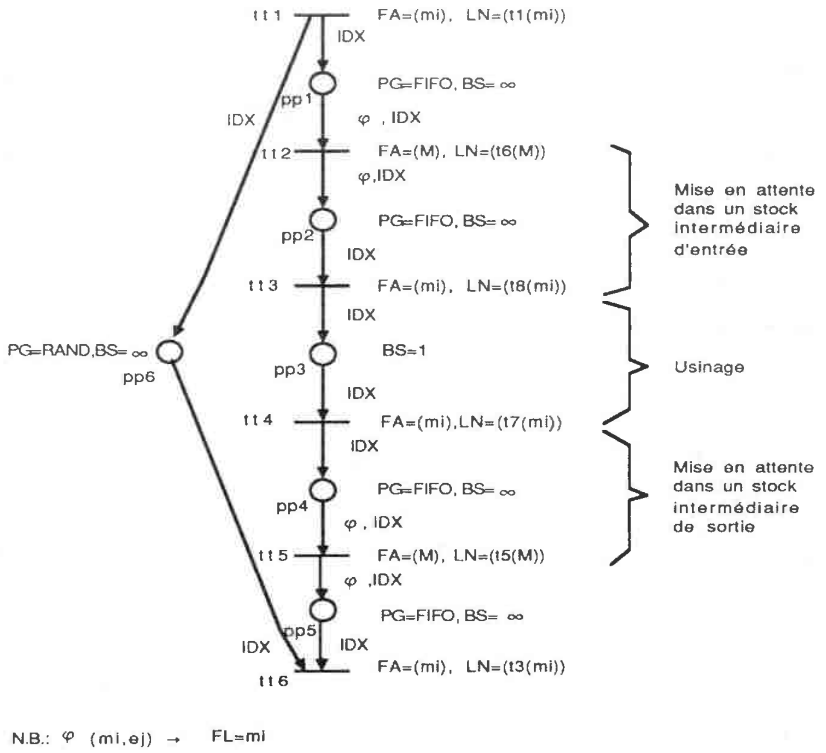


Figure 4.7

4.4.3 L'EXEMPLE DU SERVEUR DE POSTES D'USINAGE

Nous présentons un troisième exemple qui illustre bien l'intérêt de la démarche lorsque le problème du choix de la stratégie de pilotage se pose. Il s'agit d'une section composée de postes de travail identiques disposés de part et d'autre d'une voie centrale. Des chariots filoguidés se déplaçant sur la voie alimentent ces postes de travail en pièces non usinées en échange de pièces déjà usinées. Sachant qu'il n'existe pas de stocks intermédiaires au niveau des postes de travail, qu'un chariot ne peut transporter qu'une pièce à la fois, et qu'une pièce ne doit être usinée qu'une seule fois, le problème ici est de déterminer le nombre de postes de travail et une stratégie de desserte, qui permettent d'optimiser le temps de transit des pièces, et le taux d'utilisation des postes.

Deux stratégies de desserte peuvent être envisagées. Lorsqu'un chariot arrive à la station d'arrêt d'un poste, la stratégie la plus naïve consiste à desservir en priorité ce poste s'il n'est pas en cours d'usinage, ou dès que l'usinage en cours se termine, puis à ne faire avancer le chariot que dans les deux cas suivants :

-Le chariot est chargé d'une pièce usinée, et la station d'arrêt suivante est dégagée.

-Le chariot est chargé d'une pièce non usinée, le poste est en cours d'usinage, et la station d'arrêt suivante est dégagée.

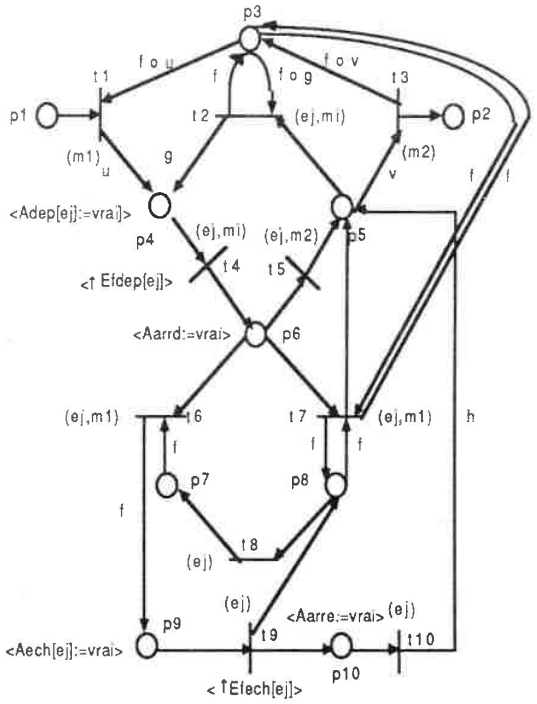
La figure 4.8 présente la spécification du système de commande correspondant à cette stratégie :

-Les couleurs m_i servent à distinguer les marques correspondant à des pièces non usinées, des marques correspondant à des pièces usinées.

-Les couleurs e_i identifient les postes de travail.

-Les places p_3, p_4, p_5 et les transitions t_1, t2, t3 décrivent les contraintes d'évolution des chariots sur la voie selon le principe FIFO.

-La place p_6 sert à modéliser l'arrivée des chariots aux stations d'arrêt des postes. Le marquage de la place p_8 indique les postes en cours d'usinage, et le marquage de la place p_7 les postes en attente. Lorsqu'un chariot arrive dans une station, trois cas peuvent se produire : la transition t_5 est franchie si le chariot est chargé d'une pièce usinée. La transition t_7 est franchie si le chariot est chargé d'une pièce non usinée, et si le poste est en cours d'usinage et la station d'arrêt suivante dégagée. Enfin, la transition t_6 est franchie si le chariot est chargé d'une pièce non usinée, et si le poste n'est pas en cours d'usinage.



MARQUAGE INITIAL

$M(p3) = \sum_{j=1}^{n} e_j$ $M(p7) = \sum_{i=1}^{n} e_j$

ENSEMBLE DES COULEURS

{ mi, ej, <ej, mi> } $\forall i \in (1,n)$ $j \in (1,m)$

FONCTIONS

f(ej,mi) = ej
u(mi) = e1, mi
g(ej, mi) = ej+1, mi
v(mi) = em, mi
h(ej) = ej, m2

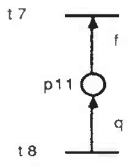
VARIABLES D'ENTREE

Efdep : tableau [1..n] de booléen	Efdep[i]=vrai indique une arrivée à la station i
Efech : tableau [1..n] de booléen	Efech[i]=vrai indique une fin d'échange à la station i

VARIABLES DE SORTIE

Aarrd : tableau [1,n] de booléen	Aarrd[i] provoque un arrêt à la station i
Adep : tableau [1..n] de booléen	Adep[i] provoque un déplacement vers la station i
Aech : tableau [1..n] de booléen	Aech[i] déclenche un échange à la station i
Aarre : tableau [1..n] de booléen	Aarre[i] termine un échange à la station i

Figure 4.8



MARQUAGE INITIAL

$M(p11) = \sum_{j=1}^{n} j * e_{n-j+1}$

FONCTIONS

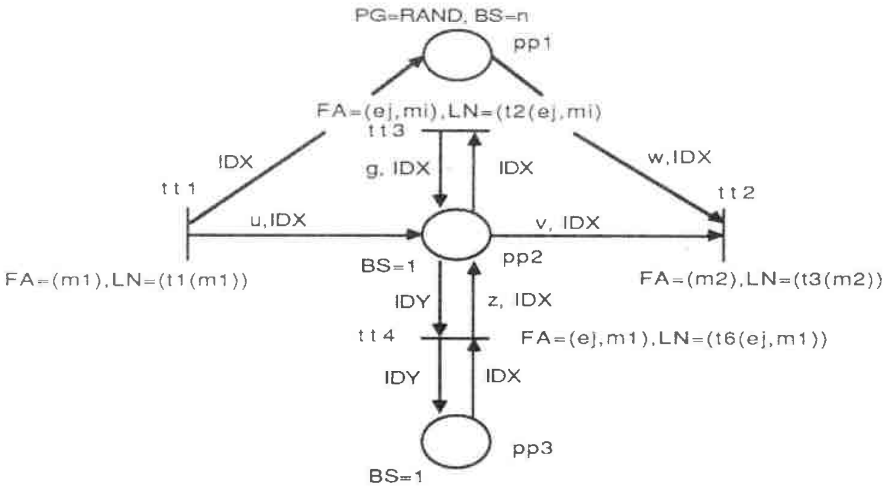
$q(\Theta i) = \sum_{j=1}^{i} e_j$

Figure 4.9

-Les places p_9, p_{10} et les transitions t_9, t_{10} décrivent la procédure d'échange.

Une autre stratégie peut consister à ne faire avancer un chariot que si l'on est sûr de trouver un poste en attente en amont. Ceci équivaut à compléter le réseau de la figure 4.8 par les contraintes exprimées dans le réseau de la figure 4.9. Si n désigne le nombre de postes, pour une station d'arrêt de rang i, la place p_{11} contient au départ n - i marques, chaque marque correspondant à un poste en attente en amont. Si un chariot arrive à cette station, la transition t_7 n'est franchie que si en plus toutes ces marques n'ont pas déjà été consommées. Par ailleurs, tout franchissement de la transition t_8 correspondant à une fin d'usinage dans un poste, produit une marque dans p_{11} pour ce poste et pour tous les postes situés en aval.

La figure 4.10 correspond à la description du flux d'activité dans la section de travail. Cette description peut être utilisée pour évaluer les deux stratégies présentées plus haut. Chaque file d'attente de la place pp2 permet d'indiquer la présence d'une pièce devant un poste de travail, le passage d'une pièce d'une file à une autre étant modélisé par le tir de la transition tt3. Par ailleurs, chaque file d'attente de la place pp3 permet d'indiquer la présence d'une pièce dans un poste de travail, et les tirs de la transition tt4 de modéliser la procédure d'échange d'une pièce. Enfin, les transactions contenues dans la file d'attente modélisée par la place pp1 correspondent à tout instant aux pièces présentes dans la section. De ce fait, la place pp_1 permet d'obtenir des statistiques sur le temps de transit des pièces, et la place p_8 (fig. 4.9) des statistiques sur le taux d'utilisation des postes.



MARQUAGE INITIAL

$M(pp1) = \sum_{1}^{n} m1$ $\qquad$ $M(pp3) = \sum_{j=1}^{n} (ej, m2)$

DEFINITION DES FONCTIONS

g(ej,mi) -> FL=ej+1, CL=mi
u(mi) -> FL=e1, CL=mi
v(mi) -> FL=en, CL=mi
W(mi) -> FL=e1, CL=m1
z(ej,mi) -> FL=ej, CL=m2

Figure 4.10

DEUXIEME

PARTIE

Chapitre 5

LES OUTILS D'AIDE A LA CONCEPTION : GENERALITES

PLAN

5.0 INTRODUCTION

Nous consacrons cette deuxième et dernière partie du mémoire à la présentation des principes de deux outils: PETRI-C et PETRI-S. Ces outils complètent les modèles présentés dans la première partie et permettent d'aborder la conception d'un système de commande avec toute la rigueur qu'impose ce type d'application. PETRI-C aide le concepteur à décrire le comportement de son système avec de bonnes propriétés, et PETRI-S à vérifier que cette description est conforme au service attendu.

Dans le présent chapitre, un survol des diverses formes que prennent de façon générale les outils d'aide à la spécification nous permettra dans un premier temps de bien situer nos propositions par rapport à l'état de l'art. Ensuite, nous présenterons sommairement PETRI-S (décrit de façon plus détaillée dans [Boudebous 88]), avant d'aborder dans les chapitres 6 et 7 la présentation des principes de PETRI-C.

5.1. LA CERTIFICATION A POSTERIORI

La certification à posteriori constitue l'approche la mieux connue et la plus générale pour garantir formellement les propriétés d'un logiciel. Dans cette approche, le concepteur commence d'abord par construire de façon empirique sa spécification. Un système de preuve lui permet alors ensuite, de raisonner sur cette spécification et de vérifier formellement certaines de ses propriétés.

FLoyd [Floyd 68] et Hoare [Hoare 69] ont été parmi les premiers à poser les fondements de cette méthode selon une approche axiomatique, en proposant une axiomatisation des langages qui permet de conduire des preuves sur les propriétés liées à la sureté de fonctionnement des programmes séquentiels. Par la suite, d'autres travaux étendront cette méthode aux systèmes parallèles [Owicki 76] [Keller 76] [Lamport 80] à la vérification d'autres classes de propriétés telles que celles liées à la vivacité [Pnueli 81] [Manna 82] [Owicki

82] , puis l'adapteront aux démarches de conception descendantes et ascendantes [Milner 80] [Lamport 83] [Chen 83] [Chandy 86].

L'inconvénient majeur de ce type de méthode réside essentiellement dans le fait que l'exhibition d'une preuve reste très intuitive et difficilement mécanisable. De ce fait, la certification des propriétés d'un logiciel ne peut se faire qu'au prix d'une baisse considérable de productivité.

Les techniques de vérification des propriétés d'un RdP fondées sur l'analyse des marquages accessibles, ou sur l'algèbre linéaire, procède de la même démarche. Comme nous l'avons indiqué en annexe A, ces techniques sont mécanisables, mais ne sont malheureusement pas toujours applicables.

5.2. LA CERTIFICATION A PRIORI

A l'inverse des méthodes fondées sur la certification à posteriori, les méthodes qui se fondent sur la certification à priori visent à imposer une discipline qui garantit par construction les propriétés d'une spécification. L'intérêt primordial de telles méthodes réside dans le fait qu'elles évitent ainsi au spécifieur les étapes délicates de validation tout en lui permettant de se rendre compte le plus tôt possible des incohérences dans ses intentions. En contrepartie, ces méthodes induisent en général une limitation quant à la classe des problèmes que l'on peut ainsi spécifier.

C'est essentiellement, autour des RdP que la certification à priori s'est le plus développée.

Valette [Valette 76] , [Valette 79] a été parmi les premiers à aborder ce type de problème en proposant pour deux RdP R et R', une condition nécessaire et suffisante pour qu'une substitution d'une transition de R par R' préserve les propriétés de R. Les réseaux qui satisfont cette condition sont appelés respectivement "réseaux bien formés" et "blocs bien formés". Ainsi, par exemple, tout réseau complexe construit par raffinements successifs du réseau embryon bien formé de la figure 5.1, en utilisant les blocs bien formés de la

figure 5.2, est assuré d'être comme le réseau embryon, sain et vivant. (On notera le parallèle entre les blocs bien formés, et les structures de contrôle habituellement proposées pour la programmation structurée). Le théorème de la substitution proposé dans [André 81] généralise cette approche.

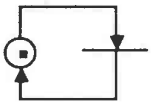


Figure 5.1

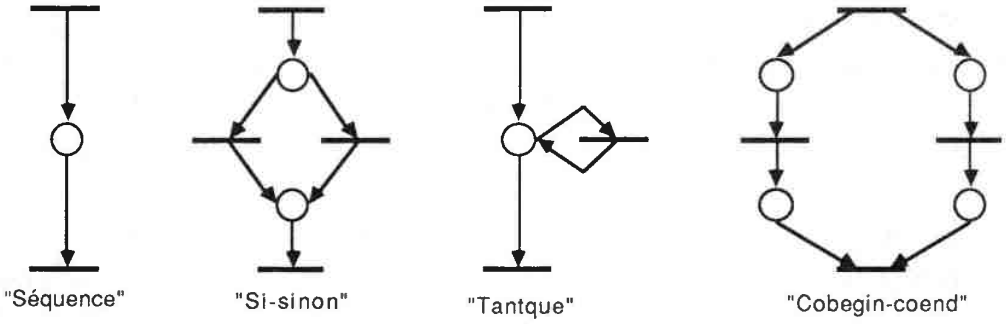


Figure 5.2

On retrouve dans [Berthelot 85 b] une approche similaire fondée sur les règles de réduction. Ces règles qui sont en général utilisées pour réduire les RdP afin de les rendre plus facilement analysables, peuvent aussi servir comme outils de raffinement. Ainsi, partant d'un RdP embryon spécifiant le service que doit assurer un protocole de communication, Berthelot montre comment il est possible de le transformer par affinements successifs pour tenir compte des erreurs de transmission et de leur recouvrement, tout en garantissant le même service et les mêmes propriétés que le réseau embryon. L'inconvénient majeur de cette approche réside essentiellement dans le fait qu'elle nécessite un effort intellectuel important, les règles de réduction n'ayant pas été conçues au départ pour servir à la description du séquencement des opérations. En contrepartie,

elle garantit la construction d'une classe importante de RdP, les RdP vivants persistants et bornés, ce qui est d'un intérêt réel pour certains types d'applications.

Une autre approche non moins intéressante consiste à exploiter les propriétés des règles de réduction conjointement avec le théorème de Commorer [Commoner 72]. Comme nous l'avons indiqué en annexe A (chapitre A.4), ce théorème donne sur des bases structurelles une condition de vivacité pour des classes particulières de réseau. Ainsi, on peut étendre les primitives de raffinement proposées dans [Valette 76] , [Valette 79] à d'autres types de primitives dont la vérification des conditions d'application se fonde sur ce théorème. C'est l'approche retenue dans [Guyot 87] pour décrire en deux étapes principales la dynamique des systèmes d'information. Dans une première étape, les primitives "séquence", "parallélisme", "simultanéité" et "précédence" permettent de raffiner un réseau embryon (cf. Figure 5.1), en un graphe d'événements. L'existence d'une condition nécessaire et suffisante de vivacité pour ce type de réseau (c.f. annexe A chapitre A.4) permet au cours du raffinement de vérifier que l'application des primitives "simultanéité" et "précédence" préservent la vivacité du réseau. Le graphe d'événements obtenu est alors ensuite raffiné à son tour en utilisant les primitives "séquence", "parallélisme" et "alternative" qui préservent la vivacité de tout réseau.

5.3 LES ALGORITHMES DE SYNTHESE

D'autres classes de propositions visant à apporter une aide au spécifieur se fondent sur des algorithmes de synthèse qui permettent de transformer globalement une spécification.

Dans certains cas, le but poursuivi est de retranscrire une spécification en améliorant ses qualités techniques sans modifier les aspects essentiels du comportement décrit.

Ainsi par exemple, on peut chercher à transformer une spécification non exécutable, où les règles de calcul sont implicites, en une spécification

exécutable où ces règles sont exprimées de façon plus explicite. C'est le cas par exemple dans [Manna 84] où l'on propose un algorithme fondé sur le calcul propositionnel permettant de passer d'une spécification de comportement exprimée en logique temporelle, à un module de synchronisation de tâches parallèles en CSP [Hoare 78] .

Dans le même esprit, on peut chercher à intégrer dans une spécification opérationnelle décrivant la logique des interactions dans l'univers d'un problème, différentes contraintes d'implémentation (contraintes de performance, contraintes de répartition, contraintes de robustesse et de maintenabilité, etc...) [Jackson 78] [Sintzoff 79] [Mahadevan 83] [Zave 84] [Zave 85] [Hikita 85].

D'autres propositions fondées sur le même principe conduisent à un enrichissement de la logique de base de la spécification initiale.

On peut citer par exemple [Ramamoorthy 85] où l'on propose un algorithme (applicable dans la conception d'un protocole de communication) qui permet de dériver le comportement d'une entité distante à partir d'un RdP décrivant le comportement du partenaire de communication de cette entité, ou encore [Zaffiropulo 80] , où l'on propose la synthèse des opérations de réception dans une description incomplète de protocole.

5.4 LE PROTOTYPAGE RAPIDE

On peut enfin citer les différentes formes d'aide au prototypage rapide qui permettent au concepteur d'aborder la clarification des besoins exprimés par les utilisateurs et de mettre au point sa spécification [Floyd 84] [Lee 85] [Kemmerer 85].

5.5 L'APPROCHE ADOPTEE

L'approche que nous adoptons s'appuie à la fois sur le principe de la certification à priori, sur celui des transformations de spécification, et sur le prototypage rapide. Elle est particulièrement bien adaptée à notre démarche globale de conception fondée sur une séparation très franche entre la description du comportement externe d'un système de commande, et le choix de sa structure interne.

Au niveau conceptuel, cette approche aide le concepteur à construire un réseau global décrivant le comportement externe de son système de commande avec de bonnes propriétés, puis au niveau organisationnel, à transformer (sur la base des choix de regroupement des entrées et des sorties) ce réseau global en sous-réseaux pouvant être répartis sur un réseau de processeurs communicant par échanges de messages. Ainsi, le concepteur peut-être dispensé de la vérification de certaines propriétés de la spécification du comportement externe de son système, de la transformation de cette spécification en une spécification décrivant le comportement interne de ce système, et de la vérification de la cohérence entre ces deux spécifications.

A chacune des deux étapes, un prototypage rapide permet de vérifier par simulation [Boudebous 88] la conformité du fonctionnement du système par rapport aux besoins exprimés par l'utilisateur, et d'évaluer les performances de l'activité commandée sous différentes hypothèses, en s'appuyant sur les trois modèles présentés dans la première partie :

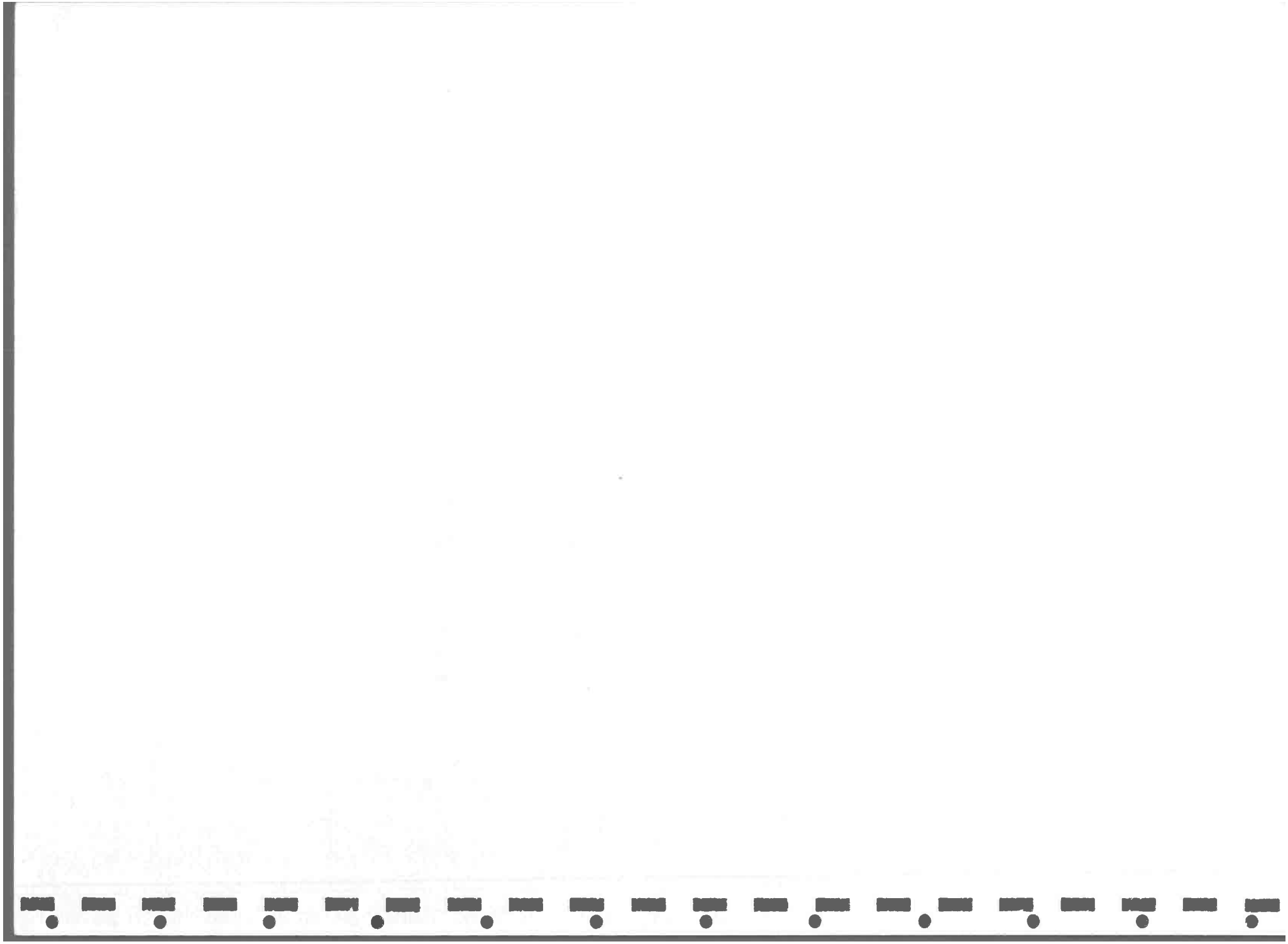
- La description du comportement du système de commande
- La modélisation du flux d'activité
- Et la modélisation du comportement de l'environnement

Dans ce qui suit, nous abordons tour à tour les principes de raffinement et

de synchronisation qui permettent de construire progressivement un réseau global décrivant le comportement externe d'un système de commande avec de bonnes propriétés (chapitre 6), et les principes de décomposition qui permettent de transformer ce réseau global en sous-réseaux décrivant le comportement interne de ce système (chapitre 7).

Chapitre 6

AIDE A LA CONCEPTION DU COMPORTEMENT EXTERNE



PLAN

6.0 INTRODUCTION

Comme nous l'avons déjà souligné, la spécification du comportement externe consiste essentiellement à définir la relation d'ordre qui doit être maintenue entre les événements d'entrée et de sortie d'un système de commande. Les principes de construction que nous présentons dans ce chapitre ont pour objectif de contribuer à aider le concepteur à définir en deux étapes cette relation. La première étape est consacrée à la définition des contraintes logiques du séquencement des ordres en sortie, et la deuxième étape à la prise en compte en entrée des contraintes de synchronisation avec l'environnement.

L'aide que nous apportons au concepteur tout au long de cette démarche se fonde en grande partie sur le principe de la certification à priori. Elle lui permet de s'assurer de la vivacité du réseau qu'il est en train de construire, et garantit la préservation des invariants sur le marquage des places qui permettent de caractériser la sûreté de fonctionnement de son système. Par ailleurs, la discipline imposée permet de guider le concepteur dans son raisonnement.

Les sections 6.1 et 6.2 sont consacrées à la présentation des principes de construction , et la section 6.3 à l'illustration de la démarche sur un exemple. Pour terminer, nous comparerons la démarche préconisée aux approches évoquées au chapitre 5 (section 6.4), et nous discuterons de ses extensions possibles (section 6.5).

6.1 DEFINITION DES CONTRAINTES LOGIQUES DU SEQUENCEMENT DES ORDRES EN SORTIE

Dans ce qui suit, nous désignons par "opération élémentaire" une opération de commande impulsionnelle ou continue, ou encore l'opération vide notée NOP (c.f section 2.4.2).

Soit alors E_s un ensemble d'étiqettes représentant l'ensemble des opérations élémentaires d'un système de commande S. Dans la démarche que nous présentons, les contraintes logiques du séquencement des ordres en sortie de S se définissent en construisant un réseau ordinaire étiqueté $R_E = < R; M_0; E_s, c>$ (c.f. annexe A chapitre A.5) dont le langage $L_E(R_E)$ définit l'ensemble des séquences d'opérations élémentaires que le fonctionnement de S peut logiquement engendrer.

Pour constuire R_E, le spécifieur identifie dans un premier temps les processus séquentiels de l'atelier qu'il représente par des réseaux embryons ayant de bonnes propriétés. Le détail de ces processus et leurs interdépendances sont ensuite affinés en appliquant dans un ordre prédéterminé :

-trois **primitives de raffinement**

"La séquence"
"L'alternative"
"La répétitive"

-et trois **primitives de synchronisation**

"L'exclusion mutuelle"
"La précédence"
"La simultanéité"

qui permettent de s'assurer de la préservation de ces bonnes propriétés .

Dans cette démarche constructive qui consiste pour une grande part à remplacer récursivement une opération macroscopique par une séquence d'opérations plus élémentaires, le théorème sur la substitution d'une transition par un bloc constitue une résultat fondamental.

Définition D6.1 : Substitution d'une transition par un block [Valette 76]

Soit R' = < P', T' ; Pre', Post' ; M_0' > un RdP. R' est un bloc , ssi il ne possède que une et une seule transition initiale (t_{ini}), et une et une seule transition finale (t_{fin}) :

(1) Pre' (., t_{ini}) = 0, Post '(., t_{ini}) > 0

(2) Pre' (., t_{fin}) > 0, Post' (., t_{fin}) = 0

(3) $\forall$ t $\in$ T - { t_{ini}, t_{fin} } : Pré (., t) > 0, Post' (., t) > 0.

Soit R = < P, T ; Pre, Post ; M_0 > un RdP tel que

(4) P $\cap$ P' = ϕ, T $\cap$ T' = ϕ.

La substitution d'une transition $t_s \in T$ par le bloc R' donne un RdP R" = < P", T" ; Pre", Post" ; M_0" > défini comme suit :

(5) P" = P + P'

(6) T" = T - { t_s } + T'

(7) M_0" = M_0 + M_0'

(8) $\forall$ t $\in$ T - { t_s } :

Pre" (p, t) = Pre (p, t) si p $\in$ P et 0 sinon

Post" (p, t) = Post (p, t) si p $\in$ P et 0 sinon

(9) $\forall$ t $\in$ T' - { t_{ini}, t_{fin} } :

Pre" (p, t) = Pre' (p, t) si p $\in$ P' et 0 sinon

Post" (p, t) = Post' (p, t) si p $\in$ P' et 0 sinon

(10) Pre" (p, t_{ini}) = Pre (p, t_s) si p $\in$ P et 0 sinon

Post" (p, t_{ini}) = Post' (p, t_{ini}) si p $\in$ P' et 0 sinon

(11) Pre" (p, t_{fin}) = Pre' (p, t_{fin}) si p $\in$ P' et 0 sinon

Post" (p, t_{fin}) = Post (p, t_s) si p $\in$ P et 0 sinon . $\otimes$

Théorème T6.1 [Valette 76]

Si la transition t_s vérifie la condition suivante

(12) $\forall M \in A(R ; M_0) : M < 2 * Pre(., t_s)$,

on dit qu'elle n'est pas t-sensibilisée simultanément avec elle-même. Dans ces conditions, dans R", si l'on franchit la transition t_{ini}, elle ne pourra plus être franchie tant que t_{fin} n'aura pas été franchie.⊗

Nous abordons à présent la présentation du réseau embryon, des primitives , puis de la grammaire du langage qui régit l'ordre d'application des primitives.

6.1.1 LE RESEAU EMBRYON

Le réseau embryon est décrit par la figure 6.1. Il est instanciable à l'aide de la primitive

EMB (O_x*)

et sert à définir la spécification initiale d'un processus séquentiel. Dans cette spécification initiale, le processus décrit réalise de façon répétitive une opération O_x dite embryonnaire. A ce stade de la spécification, l'ensemble des séquences possibles d'opérations du processus peut donc être décrit de façon triviale par le langage suivant :

(O_x*)

L'objet du raffinement sera de transformer progressivement l'opération embryonnaire en opérations de plus en plus simples liées par des contraintes de séquencement jusqu'à ce que chaque opération soit une opération élémentaire, et l'objet de la synchronisation de définir les contraintes d'entrelacement entre

les opérations élémentaires des différents processus séquentiels.

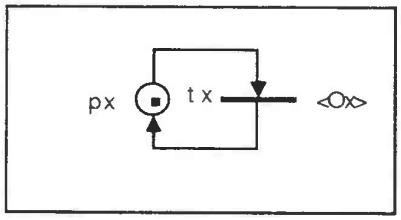


Figure 6.1

6.1.2 LES PRIMITIVES DE RAFFINEMENT ET DE SYNCHRONISATION

a) La primitive "séquence"

La primitive de raffinement "séquence", notée

$SEQ\ (O_x, (O_{x1} ; O_{x2}))$,

permet de remplacer dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences de l'opération O_x par deux opérations O_{x1} et O_{x2} qui s'exécutent dans un ordre décrit par :

$(O_{x1} ; O_{x2})$

pour réaliser la même activité que O_x. (';' désigne ici la concaténation).

Définition D6.2

Soit $R_E = < R ; M_0 ; E_s ; e >$ avec $R = < P, T ; Pre, Post >$ un RdP étiqueté où

(1) $\exists\ t_x \in T$ t. q. $e(t_x) = O_x$, $\forall\ M \in A\ (R ; M_0) : M < 2 * Pre\ (., t_x)$, $t_x^{\cdot} \neq \varnothing$

(O_x *est l'étiquette d'une transition* t_x *qui n'est pas t-sensibilisée*

simultanément avec elle même et qui possède au moins une place en sortie).

L'application de la primitive SEQ (O_x, (O_{x1} ; O_{x2})) sur R_E transforme ce réseau en un réseau $R_E' = < R' ; M_0' ; E_s' ; e' >$ avec $R' = < P', T' ; Pre', Post' >$ défini comme suit :

(2) $P' = P + \{ p_s \}$ (*Une nouvelle place* p_s *est incluse dans* P').

(3) $T' = T - \{ t_x \} + \{ t_h, t_f \}$

(*Dans* T' *la transition* t_x *est remplacée par deux nouvelles transitions* t_h *et* t_f)

(4) $\forall p \in P' : Pre' (p, t_h) = Pre (p, t_x)$ si $p \in P$ et 0 sinon

(t_h *reçoit en entrée les places qui étaient en entrée de* t_x).

(5) $\forall p \in P' : Post' (p, t_h) = 1$ si $p = p_s$ et 0 sinon

(t_h *ne reçoit en sortie que la place* p_s).

(6) $\forall p \in P' : Pre' (p, t_f) = 1$ si $p = p_s$ et 0 sinon

(t_f *ne reçoit en entrée que la place* p_s).

(7) $\forall p \in P' : Post' (p, t_f) = Post (p, t_x)$ si $p \in P$ et 0 sinon

(t_f *reçoit en sortie les places qui étaient en sortie de* t_x).

(8) $\forall t \in T'-\{t_h, t_f\}, \forall p \in P' : Pre' (p, t) = Pre (p, t)$ si $p \in P$ et 0 sinon

$Post'(p,t) = Post (p,t)$ si $p \in P$ et 0 sinon

(*Les autres transitions ont mêmes places en entrée sortie dans* R *et* R').

(9) $\forall p \in P' : M_0' (p) = M_0 (p)$ si $p \in P$ et 0 sinon.

(*Le marquage initial de* p_s *est* 0 ; *les autres places ont même marquage dans* R *et* R')

(10) $E_s' = E_s + \{O_{x1}, O_{x2} \}$

(*Le vocabulaire de* R_E' *est étendu à* O_{x1} *et* O_{x2})

(11) $e'(t_h) = O_{x1}$ et $e'(t_f) = O_{x2}$

(t_h *et* t_f *sont étiquetées respectivement par* O_{x1} *et* O_{x2}.

(12) $\forall\, t \in T - \{ t_x \}\ \ e'(t) = e(t)$

(*Les autres transitions sont étiquetées de la même façon dans* R *et* R'). ⊗

L'application de la primitive SEQ sur un réseau revient donc à substituer un bloc à une transition comme indiqué dans la figure 6.2.

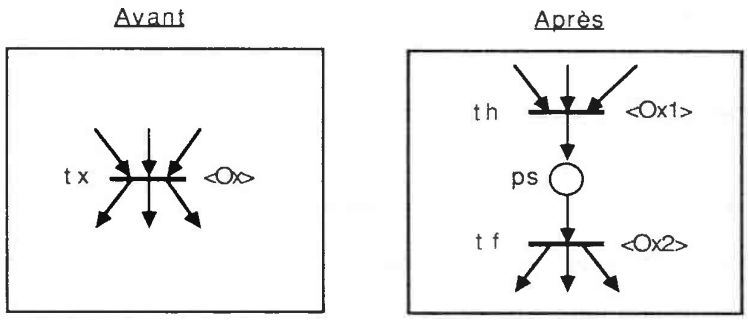


Figure 6.2

La transition t_x n'étant pas t-sensibilisée simultanément avec elle-même, le théorème 6.1, nous assure que si la transition t_h est franchie, elle ne sera plus franchissable jusqu'à ce que la transition t_f soit franchie. Par ailleurs, si la transition t_h est franchie, la marque produite dans la place p_s ne permet qu'un seul tir de t_f. On est ainsi assuré que l'image par e' de Proj ({ t_h, t_f }, L (R', M_0')) sera $(O_{x1} ; O_{x2})^*$.

b) La primitive "alternative"

La primitive de raffinement "alternative", notée

ALT (O_x, (O_{x1} | O_{x2})) ,

permet de remplacer dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences de l'opération O_x par deux opérations O_{x1} et O_{x2} qui s'exécutent dans un ordre décrit par

(O_{x1} | O_{x2})

pour réaliser la même activité que O_x. ('|' désigne ici l'alternative).

Définition D6.3 :

Soit R_E = < R ; M_0 ; E_s ; e > avec R = < P, T ; Pre, Post > un RdP étiqueté où

(1) $\exists\, t_x \in T$ t.q $e(t_x) = O_x$, $\forall\, M \in A(R ; M_0)$: $M < 2 * Pre(., t_x)$, $\cdot t_x \neq \varnothing$, $t_x^{\cdot} \neq \varnothing$

(*O_x est l'étiquette d'une transition t_x qui n'est pas t-sensibilisée simultanément avec elle même et qui possède au moins une place en entrée et une place en sortie.*)

L'application de la primitive ALT (O_x, (O_{x1} | O_{x2})) sur R_E transforme ce réseau en un réseau étiqueté R_E' = <R', M_0' ; E_s', e' > avec R' = < P', T' ; Pre', Post' > défini comme suit :

(2) P' = P

(3) T' = T - { t_x } + { t_{x1}, t_{x2} }

(*Dans T' la transition t_x est remplacée par deux nouvelles transitions t_{x1} et t_{x2}*)

(4) Pre' (., t_{x1}) = Pre'(., t_{x2}) = Pre (., t_x)

(t_{x1}, t_{x2} *reçoivent en entrée les places qui étaient en entrée de* t_x)

(5) Post' (., t_{x1}) = Post' (., t_{x2}) = Post (., t_x)

(t_{x1} et t_{x2} *reçoivent en sortie les places qui étaient en sortie de* tx)

(6) $\forall\ t \in T' - \{ t_{x1}, t_{x2} \}$: Pre' (., t) = Pre (., t), Post' (., t) = Post (., t)

(*Les autres transitions ont mêmes places en entrée et en sortie dans* R *et* R')

(7) $M_0' = M_0$

(8) $E_s' = E_s + \{ O_{x1}, O_{x2} \}$ (*Le vocabulaire* E_s' *est étendu à* O_{x1} *et* O_{x2} }

(9) $e'(t_{x1}) = O_{x1}$ et $e'(t_{x2}) = O_{x2}$

(t_{x1} *et* t_{x2} *sont étiquetées respectivement par* O_{x1} *et* O_{x2})

(10) $\forall\ t \in T - \{ t_x \}$ e'(t) = e(t)

(*Les autres transitions sont étiquetées de la même façon dans* R_E *et* R_E').⊗

L'application de la primitive ALT sur un réseau revient donc à remplacer une transition par deux transitions comme décrit dans la figure 6.3.

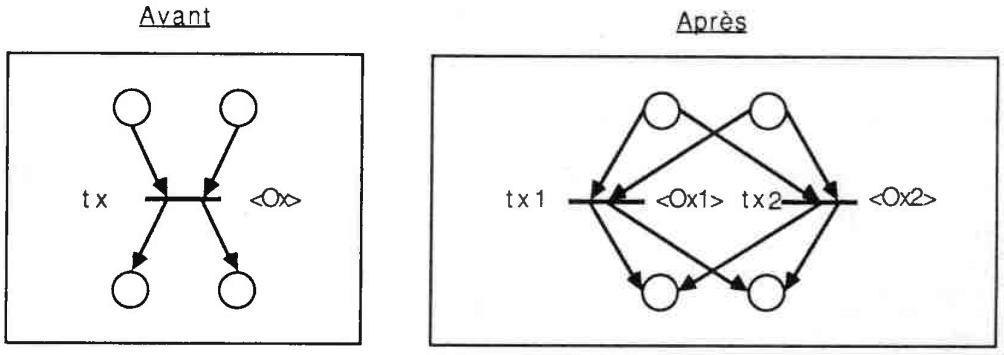


Figure 6.3

Tout marquage qui rend franchissable t_{x1} rend également franchissable t_{x2}. On est donc assuré que l'image par e' de Proj ({ t_{x1}, t_{x2} }, L(R'; M_0')) sera $(O_{x1} \mid O_{x2})^*$.

c) **La primitive "répétitive"**

La primitive de raffinement "répétitive", notée

REP (O_x, (O_{x1} ; O_{x3}* ; O_{x2})) ,

permet de remplacer dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences de l'opération O_x par trois opérations O_{x1}, O_{x2} et O_{x3}, qui s'exécutent dans un ordre décrit par (O_{x1} ; O_{x3}* ; O_{x2}) pour réaliser la même activité que O_x. (Ici également ';' désigne la concaténation. '*' indique que l'opération O_{x3} peut être exécutée zéro ou plusieurs fois).

Définition D6.4

Soit R_E = < R ; M_0 ; E_s ; e > avec R = < P, T ; Pre, Post > un RdP étiqueté où

(1) $\exists\, t_x \in T$ t.q. $e(t_x) = O_x$, $\forall\, M \in A(R ; M_0) : M < 2 * Pre(., t_x)$, $t_x^{\bullet} \neq \varnothing$.

(O_x *est l'étiquette d'une transition* t_x *qui n'est pas t-sensibilisée simultanément avec elle même et qui possède au moins une place en sortie*).

L'application de la primitive REP (O_x, (O_{x1} ; O_{x3}* ; O_{x2})) sur R_E transforme ce réseau en un réseau R_E' = < R' ; M_0' ; E_s', e' > avec R' = < P', T' ; Pre', Post' > défini comme suit.

(2) $P' = P + \{ p_s \}$

(P' *contient en plus une nouvelle place* p_s)

(3) $T' = T - \{ t_x \} + \{ t_h, t_n, t_f \}$

(*Dans* T', t_x *est remplacée par trois nouvelles transitions* t_h, t_n et t_f).

(4) $\forall\, p \in P'$: $Pre'(p, t_h) = Pre(p, t_x)$ si $p \in P$ et 0 sinon

(t_h *reçoit en entrée les places qui étaient en entrée de* t_x)

(5) $\forall\ p \in P' : Post'(p, t_h) = 1$ si $p = p_s$ et 0 sinon

(t_h *ne reçoit en sortie que la place* p_s)

(6) $\forall\ p \in P' : Pre'(p, t_f) = 1$ si $p = p_s$ et 0 sinon

(t_f *ne reçoit en entrée que la place* p_s)

(7) $\forall\ p \in P' : Post'(p, t_f) = Post(p, t_x)$ si $p \in P$ et 0 sinon

(t_f *reçoit en sortie les places qui étaient en sortie de* t_x).

(8) $\forall\ t \in T' - \{t_n, t_h, t_f\}, \forall\ p \in P'$: $Pre'(p, t) = Pre(p, t)$ si $p \in P$ et 0 sinon

$Post'(p, t) = Post(p, t)$ si $p \in P$ et 0 sinon

(*Les autres transitions, exceptée* t_n, *ont mêmes places en entrée-sortie dans* R *et* R')

(9) $\forall\ p \in P' : Pre'(p, t_n) = 1$ si $p = p_s$ et 0 sinon

(t_n *ne reçoit en entrée que la place* p_s)

(10) $\forall\ p \in P' : Post'(p, t_n) = 1$ si $p = p_s$ et 0 sinon

(t_n *ne reçoit en sortie que la place* p_s)

(11) $E_s' = E_s + \{ O_{x1}, O_{x2}, O_{x3} \}$

(*Le vocabulaire* E_s' *est étendu à* O_{x1}, O_{x2} *et* O_{x3})

(12) $e'(t_h) = O_{x1}$, $e'(t_f) = O_{x2}$, $e'(t_n) = O_{x3}$

(t_h, t_f *et* t_n *sont étiquetées respectivement par* O_{x1}, O_{x2} *et* O_{x3})

(13) $\forall\ t \in T - \{ t_x \}\ e'(t) = e(t)$

(*Les autres transitions sont étiquetées de la même façon dans* R_E *et* R_E'). ⊗

L'application de la primitive "REP" sur un réseau consiste donc à remplacer une transition par un bloc comme indiqué dans la figure 6.4.

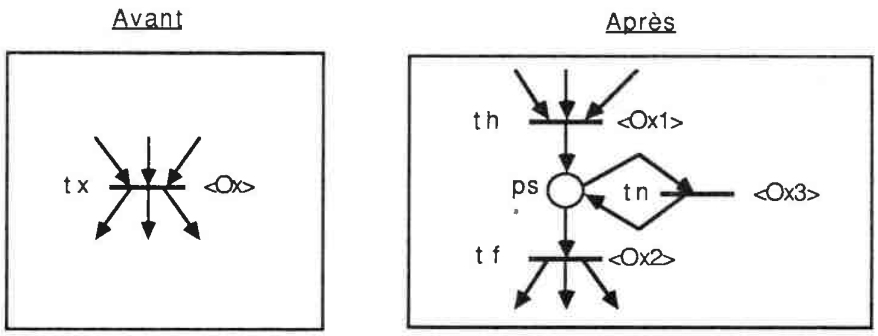


Figure 6.4

La transition t_x n'étant pas t-sensibilisée simultanément avec elle même, le théorème 6.1 nous assure dans ce cas également que si la transition t_h est franchie, elle ne sera plus franchissable jusqu'à ce que la transition t_f soit franchie. Par ailleurs, lorsque la transition t_h est franchie, la marque produite dans la place p_s ne peut servir que pour un seul tir de t_f. Entre un tir de t_h et un tir de t_f, cette marque peut servir à franchir plusieurs fois t_n. On est donc assuré que l'image par e' de Proj ({ t_h, t_f, t_n }, L (R', M_0')) sera $(O_{x1} ; O_{x3}^*, O_{x2})^*$.

d) La primitive "exclusion mutuelle"

La primitive de synchronisation "exclusion mutuelle", notée

$EXC (O_{x1} \oplus O_{x2} \oplus ... \oplus O_{xn})$,

permet de contraindre dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences des opérations non élémentaires contenues dans { O_{x1}, O_{x2}, ..., O_{xn} } à respecter l'ordre décrit par $(O_{x1} \oplus O_{x2} \oplus .... \oplus O_{xn})$ (où ' $\oplus$ ' désigne l'exclusion mutuelle).

Définition D6.5 :

Soit $R_E = < R ; M_0 ; E_s ; e >$ avec $R = < P, T ; Pre, Post >$ un RdP étiqueté où

(1) $\exists \{ t_{x1}, t_{x2}, ..., t_{xn} \} \subset T$ t. q. $e(t_{xi}) = O_{xi}$ pour i = 1, ..., n

(*Chaque* O_{xi} *est l'étiquette d'une transition* t_{xi}).

L'application de la primitive EXC ($O_{x1} \oplus O_{x2} \oplus ... \oplus O_{xn}$) sur R_E transforme ce réseau en un réseau $R_E' = < R' ; M_0' ; E_s' ; e' >$ avec $R' = < P', T' ; Pre', Post' >$ défini comme suit :

(2) $P' = P + \{ p_I \}$ (*Une nouvelle place* p_I *est incluse dans* P')

(3) $T' = T$

(4) $\forall\ t \in \{ t_{x1}, t_{x2}, ..., t_{xn} \}, \forall\ p \in P' : Pre'(p, t) = Pre(p, t)$ si $p \in P$ et 1 sinon

(*Dans* R_E', *chaque transition* t_{xi} *conserve en entrée les mêmes places que dans* R_E, *et reçoit en plus la place* p_I).

(5) $\forall\ t \in \{ t_{x1}, t_{x2}, ..., t_{xn} \}, \forall\ p \in P' : Post'(p, t) = Post(p, t)$ si $p \in P$ et 1 sinon

(*Dans* R_E', *chaque transition* t_{xi} *conserve en sortie les mêmes places que dans* R_E, *et reçoit en plus la place* p_I).

(6) $\forall\ t \in T - \{ t_{x1}, t_{x2}, ..., t_{xn} \}, \forall\ p \in P'$:

$Pre'(p, t) = Pre(p, t)$ si $p \in P$ et 0 sinon

$Post'(p, t) = Post(p, t)$ si $p \in P$ et 0 sinon

(*Toutes les autres transitions ont mêmes places en entrée et en sortie dans* R *et* R').

(7) $\forall\ p \in P' : M_0'(p) = M_0(p)$ si $p \in P$ et 1 sinon

(*Le marquage initial de* p_I *dans* R' *est* 1. *Les autres places ont le même marquage initial dans* R *et* R').

(8) $E_s' = E_s$

(9) $e' = e$.

L'application de la primitive "EXC" sur un réseau consiste donc à mettre une place p_I dont le marquage initial est 1, en entrée et en sortie des transitions t_{x1}, ..., t_{xn} (cf. figure 6.5).

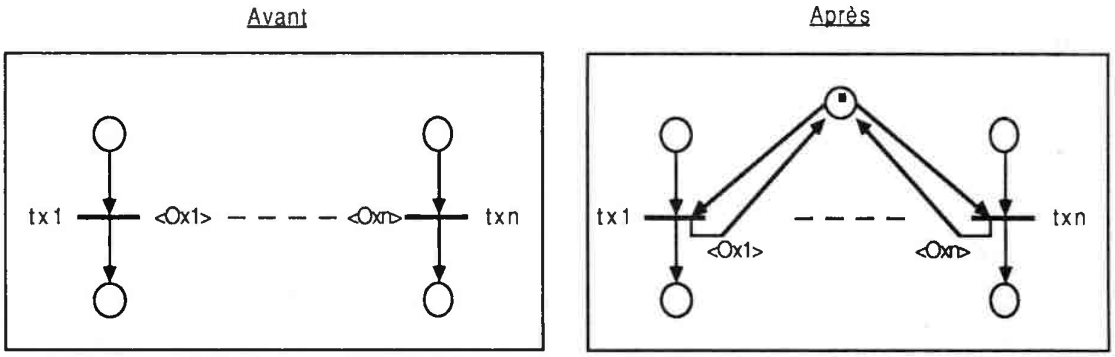


Figure 6.5

Cette place commune garantit que lorsque les opérations O_{x1}, O_{x2}, ..., O_{xn} seront raffinées, elles continueront à s'exécuter en exclusion mutuelle.

e) La primitive "précédence"

La primitive de synchronisation "précédence", notée

PRE (((O_{x1} | ..... | O_{xn}) ; (O_{y1} | ..... | O_{ym})), m_c)) ,

permet de contraindre dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences de deux ensembles d'opérations { O_{x1}, ..., O_{xn} } et { O_{y1}, ..., O_{yn} } à se synchroniser selon le schéma producteurs/consommateurs, donc à respecter l'assertion suivante :

Pour toute séquence d'opérations s, si la longueur de la projection de s sur $\{O_{x1}, ..., O_{xn}\}$ (représentant la séquence de productions) est égale à k, alors la longueur de la projection de s sur $\{O_{y1},.....,O_{ym}\}$ (représentant la séquence de consommations) doit au plus être égale à $k + m_c$.

En d'autres termes, cela revient à dire que l'on contraint la $(k + m_c)$ ième exécution dans l'ensemble d'opérations $\{O_{y1}, ..., O_{ym}\}$ à être précédée nécessairement d'au moins k exécutions dans l'ensemble d'opérations $\{O_{x1},....,O_{xn}\}$.

Définition D6.6 :

Soit $R_E = < R ; M_0 ; E_s ; e >$ avec $R = < P, T ; Pre, Post >$ un RdP étiqueté où

(1) $\exists \{ t_{x1}, ..., t_{xn}, t_{y1}, ..., t_{ym} \} \subset T$ t. q. $e(t_{xi}) = O_{xi}$ pour i = 1, ..., n

$e(t_{yj}) = O_{yj}$ pour j = 1, ..., m

(*Chaque* O_{xi} *est l'étiquette d'une transition* t_{xi}, *chaque* O_{yi} *est l'étiquette d'une transition* t_{yi}).

(2) Si $m > 1$ $\forall t \in \{t_{y1},.....,t_{ym}\}$ et $\forall p \in {}^{\cdot}t : |p^{\cdot}| = \{ t \}$

(*Si* $m > 1$, *aucune des transitions* t_{y1}, ... , t_{ym} *ne partage ses places d'entrée*)

et soit m_c un entier positif.

L'application de la primitive PRE $((O_{x1} | ..... | O_{xn}) ; (O_{y1} | .... | O_{ym}))$, m_c) sur R_E transforme ce réseau en un réseau $R_E' = < R' ; M_0' ; E_s' ; e' >$ avec $R' = < P', T' ; Pre', Post' >$ défini comme suit :

(3) $P' = P + \{ p_c \}$ (*Une nouvelle place* p_c *est incluse dans* P')

(4) $T' = T$

(5) $\forall t \in \{ t_{x1}, ..., t_{xn} \}$, $\forall p \in P'$: Pre'(p, t) = Pre(p, t) si $p \in P$ et 0 sinon

(*Les places en entrée de* t_{xi} *sont les mêmes dans* R *et* R')

(6) $\forall t \in \{ t_{x1}, ..., t_{xn} \}$,$\forall p \in P'$: Post' (p, t) = Post (p, t) si $p \in P$ et 1 sinon

(*Les places en sortie de* t_{xi} *dans* R' *incluent les places en sortie de* t_{xi} *dans* R *et la place* p_c).

(7) $\forall t \in \{ t_{y1}, ..., t_{ym} \}$, $\forall p \in P'$: Pre' (p, t) = Pre (p, t) si $p \in P$ et 1 sinon

(*Les places en entrée de* t_{yi} *dans* R' *incluent les places en entrée de* t_{yi} *dans* R *et la place* p_c)

(8) $\forall t = \{ t_{y1}, ..., t_{ym} \}$, $\forall p \in P'$: Post' (p, t) = Post (p, t) si $p \in P$ et 0 sinon

(*Les places en sortie de* t_{yi} *sont les mêmes dans* R *et* R')

(9) $\forall t \in T - \{ t_{x1}, ..., t_{xn}, t_{y1}, ..., t_{ym} \}$, $\forall p \in P'$:

Pre' (p, t) = Pre (p,t) si $p \in P$ et 0 sinon

Post' (p, t) = Post (p, t) si $p \in P$ et 0 sinon

(*Toutes les autres transitions ont mêmes places en entrée et en sortie dans* R *et* R').

(10) $\forall p \in P'$: $M_0'(p)$ est égal à $M_0(p)$ si $p \in P$ et à m_c sinon

(11) $E_S' = E_S$

(12) e' = e.

L'application de la primitive "PRE" sur un réseau consiste donc à mettre une place p_c dont le marquage initial est m_c en sortie des transitions t_{x1}, ..., t_{xn} et en entrée des transitions t_{y1}, ..., t_{ym} (cf. figure 6.6).

La place p_c apparaît ainsi comme une place de communication asynchrone entre les deux ensembles de transitions étiquetées. Son marquage initial s'interprête comme le nombre de consommations par anticipation qui peuvent être effectuées par les consommateurs. On est ainsi assuré que les contraintes d'ordonnancement qu'est sensée introduire la primitive "PRE" sur

les opérations { O_{x1}, ..., O_{xn}, O_{y1}, ... O_{ym} } seront respectées.

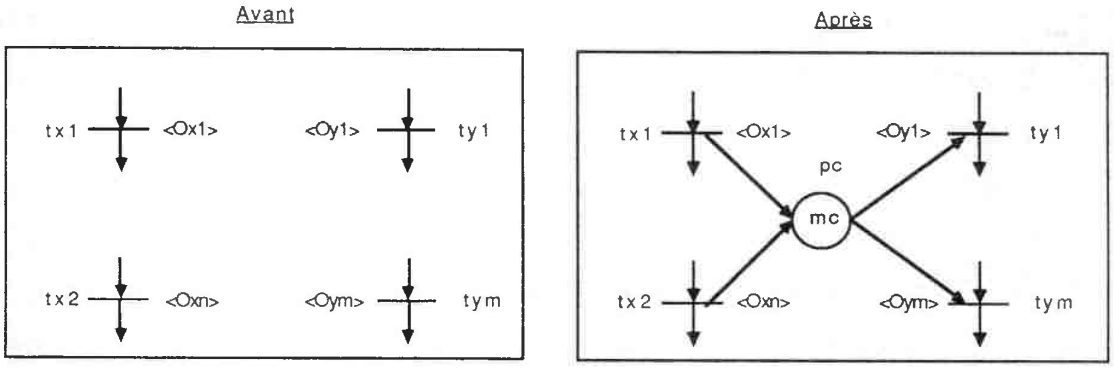


Figure 6.6

f) La primitive "simultanéité"

La primitive de synchronisation "simultanéité", notée

SIM (O_{x1} || ..... || O_{xn}) ,

permet de contraindre dans le langage décrivant l'ensemble des séquences d'opérations possibles d'un système de commande, les occurrences des opérations contenues dans { O_{x1}, ..., O_{xn} } à respecter l'ordre décrit par (O_{x1} ||....|| O_{xn}), (où "||" désigne la simultaneïté d'exécution).

Définition D6.7 :

Soit R_E = < R ; M_0 ; E_s ; e > avec R = < P, T ; Pre, Post > un RdP étiqueté où

(1) $\exists$ { t_{x1}, ..., t_{xn} } $\subset$ T t. q. $e(t_{xi}) = O_{xi}$ pour i = 1, ..., n

(*Chaque* O_{xi} *est l'étiquette d'une transition* t_{xi})

(2) $\forall$ t $\in$ { t_{x2}, ... , t_{xn} } , $\forall$ p $\in$ $^{\cdot}$t : p$^{\cdot}$ = { t }

(*Aucune des transitions* t_{x2}, ... , t_{xn} *ne partage ses places d'entrée*).

L'application de la primitive SIM ($O_{x1} \parallel ... \parallel O_{xm}$) sur R_E transforme ce réseau en un réseau $R_E' = < R' ; M_0' ; E_s' ; e' >$ avec $R' = < P', T' ; Pre', Post' >$ défini comme suit (N.B. : l'ensemble $\{ t_{x1}, ..., t_{xn} \}$ est désigné par T_x) :

(3) $P' = P$

(4) $T' = T - T_x + \{t_x\}$

(*L'ensemble de transitions* T_x *est remplacé par une seule transition* t_x).

(5) $\forall p \in P', \forall t \in T_x : Pre'(p, t_x) = Pre(p, t)$

(t_x *reçoit en entrée toutes les places qui étaient en entrée des transitions* $t_{x1}, ..., t_{xn}$)

(6) $\forall p \in P', \forall t \in T_x : Post'(p, t_x) = Post(p, t)$

(t_x *reçoit en sortie toutes les places qui étaient en sortie des transitions* $t_{x1}, ..., t_{xn}$)

(7) $\forall t \in T' - \{ t_x \} : Pre'(., t) = Pre(., t), \quad Post'(., t) = Post(., t)$

(*Les autres transitions ont mêmes places en entrée et en sortie dans* R *et* R').

(8) $M_0' = M_0$

(9) $E_s' = E_s + \{ O_x \}$

(E_s' *est étendu à* O_x *qui sert à représenter l'exécution simultanée de* $O_{x1}, ..., O_{xn}$)

(10) $\forall t \in T' \quad e'(t) = e(t)$ si $t \in T$ et O_x sinon

(t_x *est étiquetée par* O_x ; *les autres transitions conservent les mêmes étiquettes dans* R_E *et* R_E'). ⊗

L'application de la primitive "SIM" sur un réseau R consiste donc à

fusionner des transitions t_{x1}, ..., t_{xn}, pour obtenir une transition unique t_x dont la pré-condition et la post-condition de franchissement sont la conjonction des pré-conditions, et post-conditions de franchissement des transitions t_{x1}, ..., t_{xn}. (cf. figure 6.7).

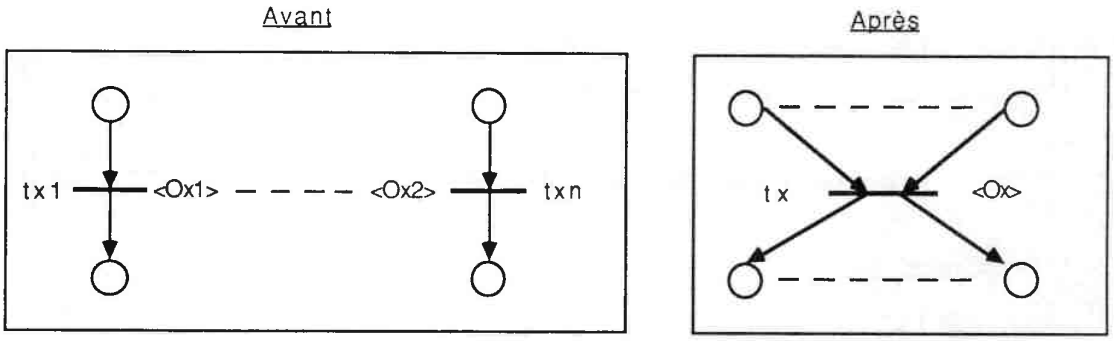


Figure 6.7

Comme l'étiquette O_x associée à la transition t_x sert à représenter l'exécution simultanée de O_{x1}, ..., O_{xn}, on est ainsi assuré que les contraintes que sont sensées introduire la primitive SIM seront toujours respectées.

6.1.3 LES PROPRIETES CARACTERISTIQUES DE L'EMBRYON ET DES PRIMITIVES DE RAFFINEMENT ET DE SYNCHRONISATION

a) Les propriétés de l'embryon

Propriété P6.1 :

On peut noter que le réseau embryon appartient à la classe des machines à états: Pre (p_x, t_x) = Post (p_x, t_x) = 1 (c.f. figure 6.1).

Il possède également les propriétés essentielles qui permettent de caractériser le bon fonctionnement d'un processus séquentiel dans un atelier. Il

est :

- vivant
- et possède un invariant de place (le marquage de la place p_x est toujours égal à 1) qui garantit que le processus modélisé ne peut être simultanément dans deux états à la fois. ⊗

Les primitives de raffinement et de synchronisation doivent en particulier permettre la préservation de ses propriétés essentielles.

b) Préservation de la classe du réseau par SEQ, ALT et REP

Propriété P6.2:

Soit $R_E = < R ; M_0 ; E_s ; e >$ avec $R = < P, T ; Pre, Post >$ un RdP étiqueté. Si R appartient à la classe des machines à états, alors tout RdP étiqueté $R_E' = < R' ; M_0' ; E_s', e' >$ avec $R' = < P', T' ; Pre', Post' >$ obtenu en appliquant l'une des primitives SEQ, ALT ou REP sur le réseau R_E est tel que R' appartient également à la classe des machines à états.⊗

(c.f. démonstration en annexe B , chapitre B.1)

c) Préservation de la vivacité et du caractère conservatif par SEQ, ALT, REP et EXC

Propriété P6.3:

Si $R_E = < R ; M_0 ; E_s, e >$ avec $R = < P, T ; Pre, Post >$ est un RdP étiqueté vivant et conservatif, alors le réseau $R_E' = < R' ; M_0' ; E_s' ; e' >$ avec $R' = <P', T' ; Pre', Post'>$ obtenu en appliquant l'une des primitives SEQ, ALT, REP, ou EXC sur R_E est aussi un RdP étiqueté vivant et conservatif.⊗

(c.f. démonstration en annexe B chapitre B.2)

On notera tout particulièrement que dans le cas de la primitive ALT, les supports de semi-flots (c.f. définition en annexe A chapitre A.3) et leur pondération restent inchangés. La primitive EXC introduit un nouveau support de semi-flots ne contenant que la place p_I pondérée à 1. Dans le cas des primitives SEQ et REP, les nouveaux supports de semi-flots s'obtiennent en remplaçant dans les anciens supports toute place $p \in t_x^{\cdot}$ par (p, p_S). Le poids de p reste chaque fois inchangé, et le poids de p_S égal au poids de p.

d) Cas des primitives PRE et SIM

A l'inverse des primitives SEQ, ALT, REP, et EXC, l'application des primitives PRE et SIM ne préserve pas de façon systématique la vivacité d'un réseau. Cependant, les contraintes d'application imposées en D6.7 (2) permettent de traduire de façon équivalente la primitive SIM à l'aide de la primitive PRE. Par ailleurs, les contraintes imposées pour l'application de la primitive PRE (c.f. D6.6 (2)) permettent de restreindre le réseau en construction à un graphe simple. Le théorème de Commoner étant applicable pour cette classe de réseaux (c.f. annexe A, chapitre A.4) , on peut ainsi obtenir par ce biais une condition suffisante de vivacité basée sur la structure du réseau pour l'application de ces deux primitives.

Propriété P6.4 :

Soit $R_E = < R ; M_0 ; E_s ; e >$ un RdP étiqueté avec $R = < P, T ; Pre, Post >$.

Soient

-$R_E' = < R' ; M_0' ; E_s', e' >$ avec $R' = < P', T'; Pre', Post' >$ un RdP étiqueté obtenu en appliquant à R_E la primitive SIM $(O_{x1} \| ... \| O_{xn})$,

-et R_E" = < R", M_0"; E_s"; e" > avec R" = < P", T"; Pre", Post" > un RdP obtenu en appliquant à R_E la séquence de primitives suivante:

Pour chaque i = 2, ..., n on effectue le raffinement

SEQ (O_{xi}, (Ox_{id} ; Ox_{if}))

puis ensuite les deux synchronisations

PRE ((Ox_{id} ; O_{x1}), 0)

PRE ((O_{x1} ; Ox_{if}), 0) .

Le réseau R" est vivant et conservatif ssi R' est vivant et conservatif.⊗

(c.f. démonstration en annexe B chapitre B.3)

Propriété P6.5:

Soit R_E = < R ; M_0 ; E_s ; e > avec R = < P, T ; Pre, Post > un RdP étiqueté. Si R est un graphe simple, alors tout réseau R_E' = < R' ; M_0' ; E_s' e' > avec R' = < P', T', Pre', Post' > obtenu en appliquant sur R_E la primitive PRE (((O_{x1}| ..... | O_{xn}) ; (O_{y1} | ... | O_{ym})), m_c) est tel que R' est également un graphe simple.⊗

(c.f. démonstration en annexe B chapitre B.4)

Propriété P6.6 :

Soient :

- R_E = < R ; M_0 ; E_s ; e > avec R = < P, T ; Pre, Post > un RdP étiqueté
- F = { f | f ∈ $\mathbb{N}^m$, f^t * C = O } l'ensemble des semi-flots associés à R.

Si $R_E' = < R' ; M_0' ; E_s', e' >$ avec $R' = < P', T' ; Pre', Post' >$ désigne un RdP étiqueté obtenu à partir de R_E par application de la primitive "PRE", alors

$F' = \{ f' \mid f' \in \mathbb{N}^{m+1}, \; f'(p) = f(p)$ si $p \in P$ et 0 sinon $\}$

constitue un ensemble de semi-flots pour le réseau R'. ⊗

Si $I \in P$ est un support de semi-flots pour R, après application de PRE, I reste donc un support de semi-flots pour R'.

(c.f. démonstration en annexe B chapitre B.5)

6.1.4 L'ORDRE D'APPLICATION DES PRIMITIVES

La grammaire que nous présentons dans cette section définit un ordre d'application des primitives de raffinement et de synchronisation. L'ordre retenu tire parti des propriétés caractéristiques de ces primitives pour garantir la vivacité des réseaux que l'on peut constuire, et pour déterminer de façon systématique les invariants de places qui caractérisent les propriétés liées à la sûreté de fonctionnement du système modélisé, telle l'impossibilité pour un processus séquentiel d'être simultanément dans deux états à la fois, ou l'impossibilité pour des processus séquentiels d'être simultanément dans une section critique.

<u>La grammaire</u>

(1) Créer-Réseau ::= Créer-Embryons Raffiner-Embryons Composer Raffiner-A-Nouveau.

(2) Créer-Embryons ::= EMB Créer-Embryons | ε

(3) Raffiner-Embryons ::= Opération-De-Raffinement Raffiner-Embryons | ε

(4) Opération-De-Raffinement ::= SEQ | ALT | REP

(5) Composer ::= Opération-De-Synchronisation

Composer | ε

(6) Opération-De-Synchronisation ::= PRE | SIM

(7) Raffiner-A-Nouveau ::= Opération-De-Raffinement
Raffiner-à-Nouveau
| EXC Raffiner-A-Nouveau | ε

La règle initiale de la grammaire

Elle correspond à la règle (1). Elle décrit le processus de construction d'un réseau comme étant constitué de quatre étapes qui se déroulent en séquence :

- La première étape concerne la création d'un réseau embryon pour chaque processus séquentiel de l'atelier.
- La seconde étape concerne le raffinement des réseaux embryons.
- La troisième étape la composition parallèle des réseaux raffinés.
- Et la quatrième étape le raffinement du réseau global obtenu par composition parallèle.

L'étape "Créer-Embryons"

La règle (2) décrit l'étape "Créer-Embryons" comme une succession d'applications de la primitive "EMB". L'aboutissement de cette étape est la création de réseaux embryons correspondant chacun à la spécification initiale d'un processus séquentiel de l'atelier.

La propriétés P6.1 nous assure que chacun de ces réseaux, disons R_{Ei}, appartient à la classe des machines à états, est vivant, et est caractérisé par un invariant de place qui assure que le processus modélisé ne peut pas être simultanément dans plusieurs états à la fois.

L'étape "Raffiner-Embryons"

Les règles (3) et (4) décrivent l'étape "Raffiner-Embryons" comme une

succession d'applications des primitives "SEQ", "ALT" et "REP" sur les réseaux embryons créés à l'étape précédente.

Les propriétés caractéristiques de ces trois primitives garantissent que les réseaux résultants auront les mêmes propriétés que les réseaux embryons. En particulier, pour chaque réseau résultant $R_{Ei} = < R_i, M_{0i} ; E_{si} ; e_i >$ avec $R = < P_i, T_i ; Pre_i, Post_i >$, on peut dire :

- que R_i appartient à la classe des machines à états (cf. propriété P6.2) ;
- que R_i est vivant (cf. propriété P6.3) ;
- et que R_i est caractérisé par un invariant de places

$\forall M \in A(R_i ; M_{0i}) : \Sigma_{p \in Pi} M(p) = \Sigma_{p \in Pi} M_{0i}(p) = 1$ (cf. propriété P6.2)

qui assure que le processus séquentiel modélisé ne peut pas être simultanément dans plusieurs états à la fois.

L'étape "composer"

Les règles (5) et (6) décrivent l'étape "composer" comme une étape de définition des contraintes de synchronisation devant exister entre les processus séquentiels décrits par les réseaux issus de l'étape précédente, en utilisant les primitives "PRE" et "SIM".

Comme les graphes simples inclus les machines à états, on peut considérer qu'au départ de cette étape, le regroupement des différents réseaux de l'étape précédente constitue un graphe simple. A partir de là, les propriétés P6.4 et P6.5 garantissent l'existence d'une condition suffissante basée sur la structure du réseau pour vérifier que les applications successives des primitives "PRE" et "SIM" préservent la vivacité. Par ailleurs, en se référant à la propriété P6.6, on est également assuré que l'application de ces deux primitives préserve l'ensemble des invariants de places de l'étape précédente.

L'étape "Raffiner-A-Nouveau"

Les règles (7) et (4) décrivent l'étape "Raffiner-A-Nouveau" comme une succession d'applications, des primitives "SEQ", "ALT", "REP" et "EXC" sur le réseau composé issu de l'étape précédente.

La propriétés P6.3 garantit au cours de cette dernière étape la préservation de la vivacité du réseau résultant. Par ailleurs, elle nous assure que les processus séquentiels qui étaient caractérisés dans les étapes précédentes par des invariants de places seront toujours caractérisés par des invariants de places de même nature. Si P_S désigne l'ensemble des places qui servent à décrire l'état d'un processus séquentiel S, alors pour tout marquage accessible M du réseau résultant $\Sigma_{p \in Ps} M(p) = 1$. Ce qui signifie que tout processus séquentiel dans le réseau final ne pourra jamais être simultanément dans plusieurs états différents.

Enfin, la propriété P6.3 nous assure de l'existence d'un invariant de places qui garantit la bonne utilisation de chaque ressource partagée dans le système modélisé. Si P_r désigne l'ensemble des places qui servent à décrire l'état de l'utilisation d'une ressource par un ensemble de processus séquentiels, on est assuré que pour tout marquage accessible M du réseau final $\Sigma_{p \in Pr} M(p) = 1$. Ce qui signifie que la ressource ne peut être utilisée qu'en exclusion mutuelle.

6.2 PRISE EN COMPTE DES SYNCHRONISATIONS AVEC L'ENVIRONNEMENT

Le réseau que l'on obtient à l'issue de la démarche présentée décrit de façon incomplète le comportement d'un système de commande. Il ne spécifie pas pour l'instant l'état que doivent avoir les grandeurs physiques de la partie opérative lors du déclenchement des opérations de commande. Dans cette section, nous présentons un algorithme qui permet de transformer de façon

systèmatique cette spécification incomplète en une spécification complète exprimée en terme de RdPI. Les transformations introduites par cet algorithme consiste donc pour l'essentiel à contraindre la production des événements de sortie selon la relation d'ordre définie dans la section précédente, à être synchronisée en plus à l'évolution d'un environnement. La spécification qui en résulte peut de ce fait être plus contraignante. Nous examinons ensuite les propriétés du comportement du système sous les nouvelles contraintes ainsi définies.

6.2.1 L'ALGORITHME DE TRANSFORMATION

Soit $R_E = < R ; M_0 ; E_s ; e >$ avec $R = < P, T ; Pre, Post >$ un RdP étiqueté issu de l'étape précédente. Ce réseau est donc tel que le langage $L_E(R_E)$ défini l'ensemble des séquences d'opérations élémentaires que peut engendrer un système de commande, si on ne tient pas compte des synchronisations avec l'environnement. A chaque transition t_x de R, nous associons en plus les deux informations complémentaires suivantes : pr_x et op_x.

- pr_x est un prédicat qui définit les conditions sur l'état de l'environnement qui doivent être réunies pour que l'opération élémentaire associée à l'étiquette $e(t_x)$ puisse être déclenchée. Ce prédicat s'exprime simplement en termes de conditions, d'événements se produisant dans l'environnement, et de politique de prise en compte de ces événements. Si la contrainte de séquencement définie par le réseau suffit pour déclencher l'opération associée à l'étiquette $e(t_x)$, nous considérons pr_x comme étant le prédicat toujours vrai pr_0.

- op_x définit l'opération associée à l'étiquette $e(t_x)$. Comme nous l'avions déjà noté, cette opération peut être vide, i.e. correspondre à NOP.

Si l'on désigne respectivement par OP et PR l'ensemble des opérations et l'ensemble des prédicats ainsi associés aux transitions, et par V l'ensemble des variables sur lesquelles ces opérations et ces prédicats sont définies, le RdPI $R_I = < R'; V ; OP ; PR ; \Phi ; \Psi >$ avec $R' = < P', T' : Pre', Post' >$ décrivant de façon complète le comportement externe du système de commande s'obtient à partir de R_E en effectuant les transformations suivantes :

(1) Au départ $P' = P$, $T' = T$, $Pre' = Post' = 0$,

$\forall p \in P' : \Phi(p) = NOP$, $\forall t \in T' : \Psi(t) = pr_0$

(2) Pour chaque transition $t_x \in T$ telle que :

.pr_x désigne le prédicat associé à t_x

. op_x désigne l'opération associée à t_x

. P_x désigne l'ensemble des places situées en sortie de t_x qui n'ont pas été introduites dans P par une primitive de synchronisation PRE ou EXC

. K_x désigne l'ensemble des places situées en sortie de t_x qui ont été introduites dans P par une primitive de synchronisation PRE ou EXC

on effectue sur le réseau R' les transformations caractérisées comme suit :

(3) $Pre' (., t_x) = Pre (., t_x)$

(t_x *conserve les mêmes places en entrée dans* R *et* R')

(4) $\Psi(t_x) = pr_x$

(5) $\forall p \in K_x : Post' (p, t_x) = Post (p, t_x)$

(t_x *conserve systématiquement en sortie les places de* K_x).

(6) Si $(P_x = \{ p_x \}$ et $\cdot p_x = t_x)$ ou $op_x = NOP$

(P_x *contient une seule place* p_x *qui n'a comme entrée que la transition*

t_x, *ou une opération vide est associée à* t_x)

alors

-$\forall$ p $\in$ P_x : Post' (p, t_x) = Post (p, t_x)

(*les places contenues dans* P_x *restent en sortie de* t_x)

-si $P_x = \{p_x\}$ $\Phi(p_x) = op_x$

(*si* P_x *ne contient qu'une seule place,* op_x *est associée à cette place*)

(*Cette règle de transformation est schématisée par la figure* 6.8).

(7) Si $P_x = \{px\}$ et $|p_x| > 1$ et $op_x \neq$ NOP)

(P_x *contient une seule place qui a en entrée plusieurs transitions et une opération non vide est associé à* t_x)

ou $|P_x| > 1$ et $op_x \neq$ NOP)

(P_x *contient plus d'une place et une opération non vide est associée à* t_x)

alors

-P' $\leftarrow$ P' + { p_{xs} }

(*Une nouvelle place* p_{xs} *est ajoutée à* P')

-T' $\leftarrow$ T' + { t_{xf} }

(*Une nouvelle transition* t_{xf} *est ajoutée à* T')

-Post' (p_{xs}, t_x) = 1 (*La place* p_{xs} *est placée en sortie de* t_x)

-Pre' (p_{xs}, t_{xf}) = 1 (*La place* p_{xs} *est placée en entrée de* t_{xf})

-$\forall$ p $\in$ P_x : Post' (p, t_{xf}) = Post (p, t_x)

(*Les places contenues dans* P_x *sont placées en sortie de* t_{xf})

-$\Phi(p_{xs}) = op_x$

(*L'opération* op_x *est associée à la nouvelle place* p_{xs})

(*Cette règle de transformation est schématisée par la figure* 6.9).$\otimes$

Le principe général de l'algorithme de transformation s'articule donc autour de deux types de transformations essentielles.

La première transformation consiste à associer à chaque transition t_x du réseau initial, les conditions sur l'environnement pour que l'opération op_x qui lui était associée puisse être déclenchée. Après cette transformation, le tir de la transition t_x ne servira donc plus à modéliser l'exécution de op_x, selon les contraintes d'ordre souhaitées, mais servira plutôt à traduire le fait que les conditions sur l'état de l'environnement sont réunies pour que ce traitement puisse être exécuté selon l'ordre souhaité.

La deuxième transformation consiste à associer à une place sortie de t_x le traitement op_x. Ceci revient à définir la conséquence du tir de la transition t_x dans le réseau transformé comme étant l'exécution de op_x. Il est bien évident que pour que les contraintes d'ordre initialement définies soient respectées, il est primordial que l'opération associée à la place sortie de t_x ne puisse pas être exécutée comme la conséquence du franchissement de tout autre transition t_y différente de t_x. Ceci équivaut à dire que cette place sortie de t_x ne doit pas être également en sortie d'autres transitions.

Par ailleurs, lorsque plusieurs places sont en sortie de t_x, il se pose le problème du choix de la place à laquelle op_x doit être associée.

Pour résoudre ces deux problèmes, l'algorithme distingue les places situées en sortie de t_x qui servent à décrire l'état des processus séquentiels (c'est l'ensemble appelé P_x) des places situées en sortie de t_x qui servent à leur synchronisation (c'est l'ensemble appelé K_x).

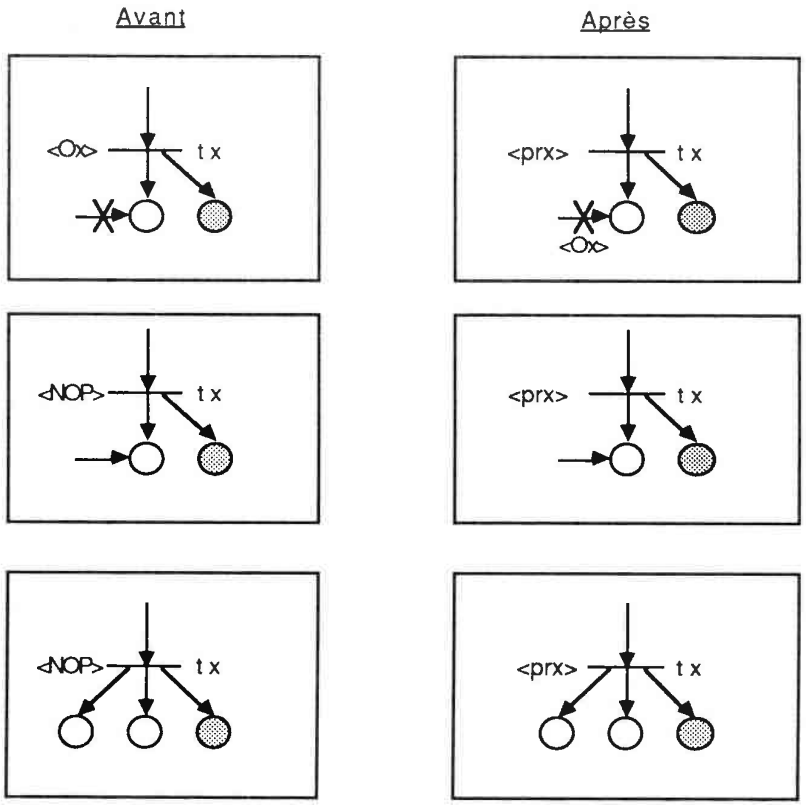
Si P_x contient une seule place qui est telle que t_x est la seule transition en entrée de cette place, ou si op_x correspond à une opération vide, dans les transformations effectuées, il n'est pas nécessaire de modifier la structure du

réseau (cf. figure 6.8).

Sinon, on est en présence d'une situation qui nécessite une transformation du réseau. Cette situation correspond soit au cas où P_x contient une seule place, telle que t_x n'est pas la seule transition en entrée de cette place, alors que $op_x \neq$ NOP, soit au cas où P_x contient plusieurs places, alors que $op_x \neq$ NOP. Dans le premier cas, nous avons vu qu'on ne peut pas associer op_x à la place contenue dans P_x sans remettre en cause les contraintes d'ordre initialement définies. Le second cas pose un problème de choix de place. Dans les deux cas, le réseau est transformé comme indiqué sur la figure 6.9. Cette transformation consiste à ajouter dans le réseau une place p_{xs} qui n'a comme transition d'entrée que la transition t_x, puis en sortie de cette place une nouvelle transition t_{xf} qui n'a comme places de sortie que les places contenues dans Px, qui disparaîssent en sortie de t_x. op_x est alors associée à la nouvelle place p_{xs}. Cette transformation constitue le seul type de transformation que l'algorithme peut être amené à effectuer sur la structure du réseau. On peut constater qu'elle correspond à l'ajout d'une place substituable, et que de ce fait, elle préserve chaque fois l'abstraction sur l'ensemble de transitions du réseau initial. On est ainsi assuré que les contraintes d'ordre initialement définies sur les opérations ne seront pas violées dans le RdPI résultant :

-Les opérations ne sont exécutées que comme la conséquence du tir des transitions auxquelles elles étaient initialement associées.

- Par ailleurs, l'ordre de franchissement de ces transitions reste conforme aux contraintes initiales.



N.B.: représente les places de Kx

désigne que la place n'a aucune transition en entrée

Figure 6.8

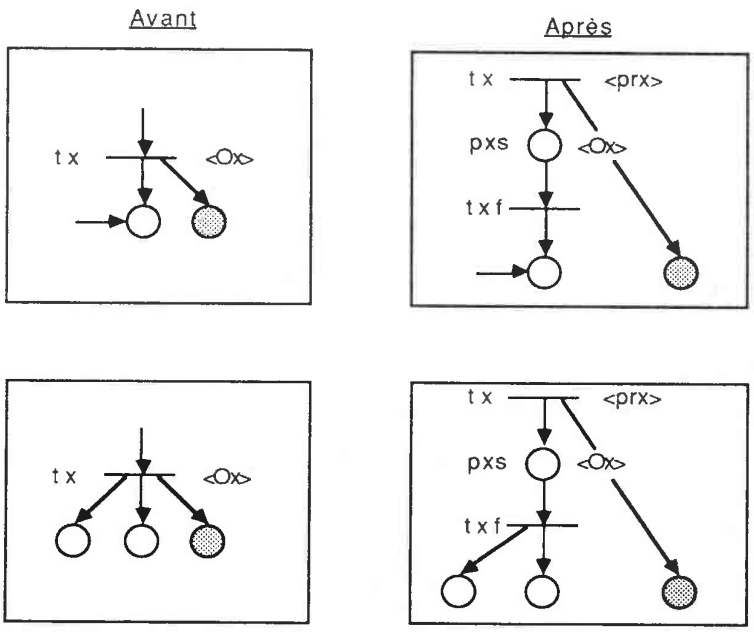


Figure 6.9

6.2.2 LES PROPRIETES DU RESEAU TRANSFORME

Nous abordons à présent une discussion sur les propriétés du RdPI résultant. Comme nous l'avions déjà noté, de façon générale, l'interprétation peut resteindre les séquences de franchissement possible d'un RdP, et remettre en cause sa vivacité. Par contre, il est facile de se convaincre que l'interprétation dans un RdPI n'invalide pas les invariants de places du réseau sous jacent. Par ailleurs, nous avons aussi déjà montré que par rapport à nos règles de fonctionnement, on peut définir une condition suffisante pour que le comportement du réseau interprété soit identique à celui du réseau autonome sous-jacent(c.f. section 2.4.3.2). Si cette condition que nous rappelons est vérifée, les deux réseaux ont bien évidemment les mêmes propriétés :

- Pour tout couple de transitions (t1, t2) dont les tirs doivent être synchronisés aux occurrences d'un même événement, il ne doit pas exister

de marquage accessible dans le réseau autonome qui valide simultanément t1 et t2.

- Pour tout marquage accessible dans le réseau autonome, les transitions validées doivent être synchronisées à des événements indépendants.

A partir d'une identification des événements corrolés, il est facile de vérifier ces deux conditions sur un RdPI construit à l'aide de notre démarche. En s'appuyant sur les invariants de places calculés, on peut s'assurer que deux transitions synchronisées à un même événement, ou à des événements corrolés ne seront jamais validées simultanément. Nous pensons que dans la majorité des cas, cette vérification sera concluante :

- La spécification de tir groupé de transitions sur occurrence d'un même événement ne nous paraît pas être fondamental dans les ateliers de production. Dans tous les cas, comme nous l'avons déjà précisé, cela restraint les possibilités de répartition de l'application.

- Par ailleurs, la synchronisation entre processus séquentiels se conçoit plus naturellement comme une synchronisation logique (liée au marquage des places) plutôt que comme une synchronisation physique (liée à l'occurrence d'événements externes). De ce fait, la spécification de processus séquentiels pouvant accéder à un état où leur évolution dépend de l'occurrence d'événements de l'environnement qui sont corrolés, ne nous paraît pas non plus être une situation courante.

Si la vérification n'est pas concluante et correspond véritablement à une contrainte souhaitée, le spécifieur doit recourir à des analyses plus fines pour garantir la vivacité de son réseau interprêté.

6.3 ILLUSTRATION DE LA DEMARCHE SUR UN EXEMPLE

Dans cette section, nous utilisons l'exemple de l'atelier de bobinage pour

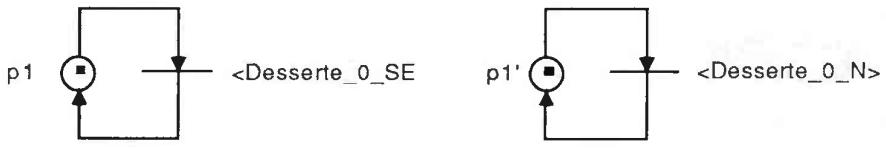
illustrer les différentes étapes de la démarche de conception.

L'étape "Créer-Embryons"

Dans le fonctionnement d'un métier, on peut distinguer deux processus séquentiels : l'un lié à l'activité assurée par le système embarqué, et l'autre lié à l'activité assurée par la navette.

EMB(Desserte-0-SE) instancie le réseau $R0_E$ qui correspond à la spécification initiale du processus séquentiel lié au système embarqué, et EMB(Desserte-0-N) le réseau $R0_E'$ qui correspond à la spécification initiale du processus séquentiel lié à la navette (cf. Figures 6.10 (a) et figure 6.10 (c)).

Outre le caractère vivant de cette spécification initiale, elle est caractérisée par les invariants de places des figures 6.10 (b) et 6.10 (d).



(a) réseau $R0_E$
(b) Invariant : M(p1) = 1
(c) réseau $R0_E'$
(d) Invariant : M(p1')

Figure 6.10

L'étape "Raffiner-Embryons"

Nous abordons d'abord le raffinement du réseau $R0_E$.

1) Une desserte doit se terminer nécessairement par le retour du système embarqué à sa position de dégagement. On peut donc appliquer le raffinement suivant :

SEQ (Desserte-0-SE, (Desserte-1-SE ; Retour-SE))

on obtient le réseau $R1_E$ de la figure 6.11 (a). Par ailleurs, dans le support de semi-flots du réseau $R0_E$, (p1, p2) est substitué à la place p1 , ce qui donne l'invariant de places de la figure 6.11 (b).

2) Pour effectuer une desserte, il faut nécessairement commencer par déclencher le déplacement du système embarqué (Adepse) vers un bobinoir. Par ailleurs, le retour du système embarqué à sa position de dégagement s'effectue à l'aide de deux actions : le déclenchement du déplacement du système embarqué vers sa position de dégagement (Aretse), suivi du déclenchement de l'arrêt du système embarqué quand il atteint sa position de dégagement (Aarrse1). On peut donc raffiner le réseau $R1_E$ par

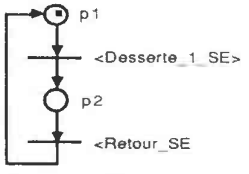
SEQ (Desserte-1-SE, (Adepse ; Desserte-2-SE))
SEQ (Retour-SE, (Aretse ; Aarrse1))

On obtient le réseau, et l'invariant de la figure 6.12.

3) Quand le système embarqué est en déplacement, on peut éventuellement l'arrêter prématurément (Aarrse2) s'il survient une casse de fil. Le réseau $R2_E$ peut donc être raffiné par

ALT(Desserte-2-SE, (Aarrse2 | Desserte-3-SE))

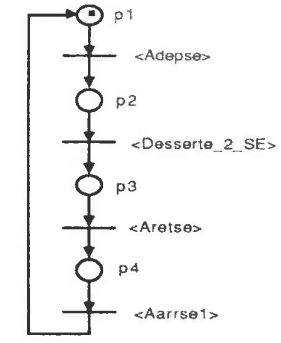
On obtient le réseau, et l'invariant de la figure 6.13.



(a) Réseau $R1_E$

(b) Invariant : M(p1) + M(p2) = 1

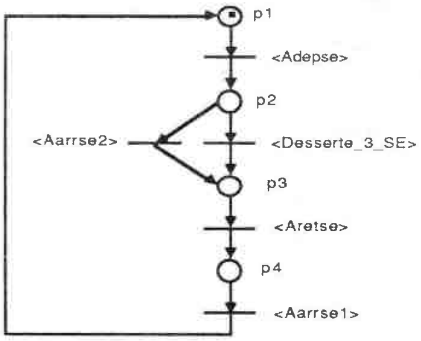
Figure 6.11



(a) Réseau $R2_E$

(b) Invariant :
M(p1)+M(p2)+M(p3)+M(p4) = 1

Figure 6.12



(a) Réseau $R3_E$

(b) Invariant :
M(p1)+M(p2)+M(p3)+M(p4) = 1

Figure 6.13

4) S'il ne survient pas de casse de fil, le système embarqué atteindra à un moment donné le bobinoir vers lequel il est en déplacement. A cet instant, il faudra l'arrêter (Aarrse3). Le réseau $R3_E$ peut donc être raffiné par

SEQ (Desserte-3-SE, (Aarrse3 ; Desserte-4-SE))

on obtient le réseau et l'invariant de la figure 6.14.

5) A son arrivée devant le bobinoir, le système embarqué peut ne rien faire s'il survient entre temps une casse de fil. Le réseau $R4_E$ doit donc être raffiné par :

ALT (Desserte-4-SE, (Anulese | Desserte-5-SE))

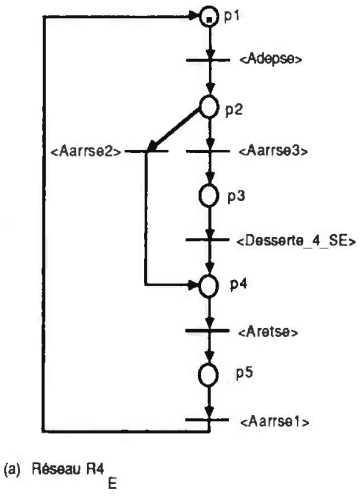
On obtient le réseau et l'invariant de la figure 6.15.

5) Le service du système embarqué sur le bobinoir quand la fin du bobinage intervient s'effectue en deux opérations : la coupe du fil (Acf) suivie de l'échange de la bobine achevée avec une bobine vierge (Aechse). Le réseau $R4_E$

doit donc être raffiné par

SEQ (Desserte-5-SE, (Acf ; Aechse)).

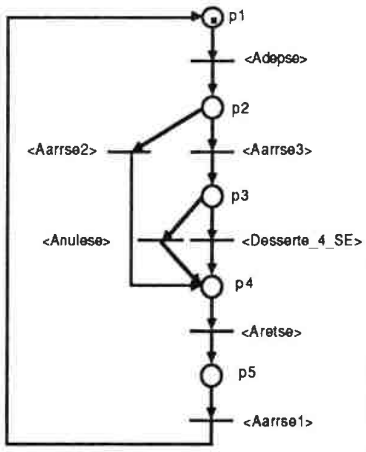
Ce qui donne en définitive le réseau et l'invariant de la figure 6.16. Ce réseau est bien entendu vivant, et l'invariant de places nous assure que le processus raffiné ne peut pas être simultanément dans deux états.



(a) Réseau $R4_E$

(b) Invariant :
M(p1)+M(p2)+M(p3)+M(p4)+M(p5) = 1

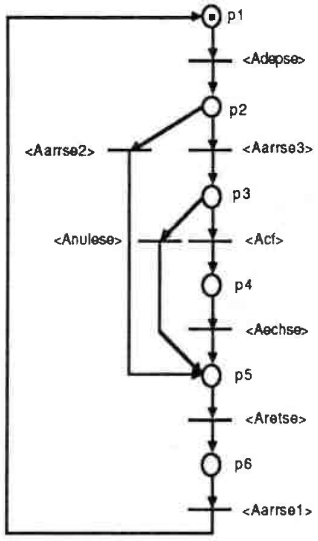
Figure 6.14



(a) Réseau $R5_E$

(b) Invariant :
M(p1)+M(p2)+M(p3)+M(p4)+M(p5) = 1

Figure 6.15

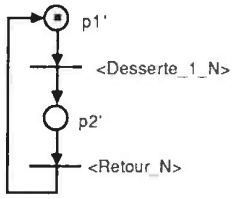


(a) Réseau $R6_E$

(b) Invariant :
M(p1)+M(p2)+M(p3)+M(p4)+M(p5)+M(p6)+M(p7) = 1

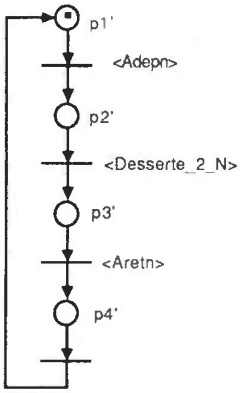
figure 6.16

On peut raffiner le réseau $R0_E$' en procédant de la même manière. La figure 6.17 résume les différentes étapes du raffinement.



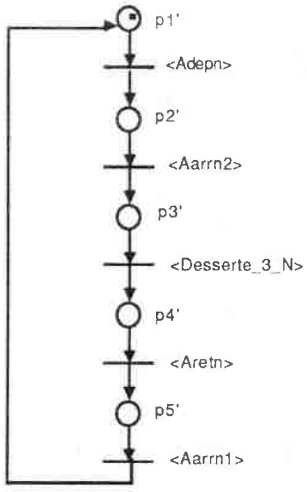
(a) Réseau R1 '$_E$

(b) Invariant : M(p1') = 1



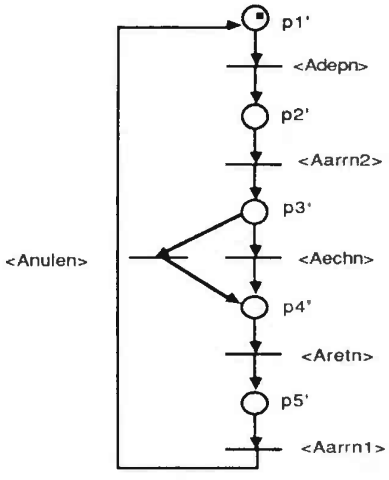
(c) Réseau R2 '$_E$

(d) Invariant :
M(p1')+M(p2')+M(p3')+M(p4') = 1



(e) Réseau R3 '$_E$

(f) Invariant :
M(p1')+M(p2')+M(p3')+M(p4')+M(p5') = 1



(g) Réseau R4 '

(h) Invariant :
M(p1')+M(p2')+M(p3')+M(p4')+M(p5') = 1

Figure 6.17

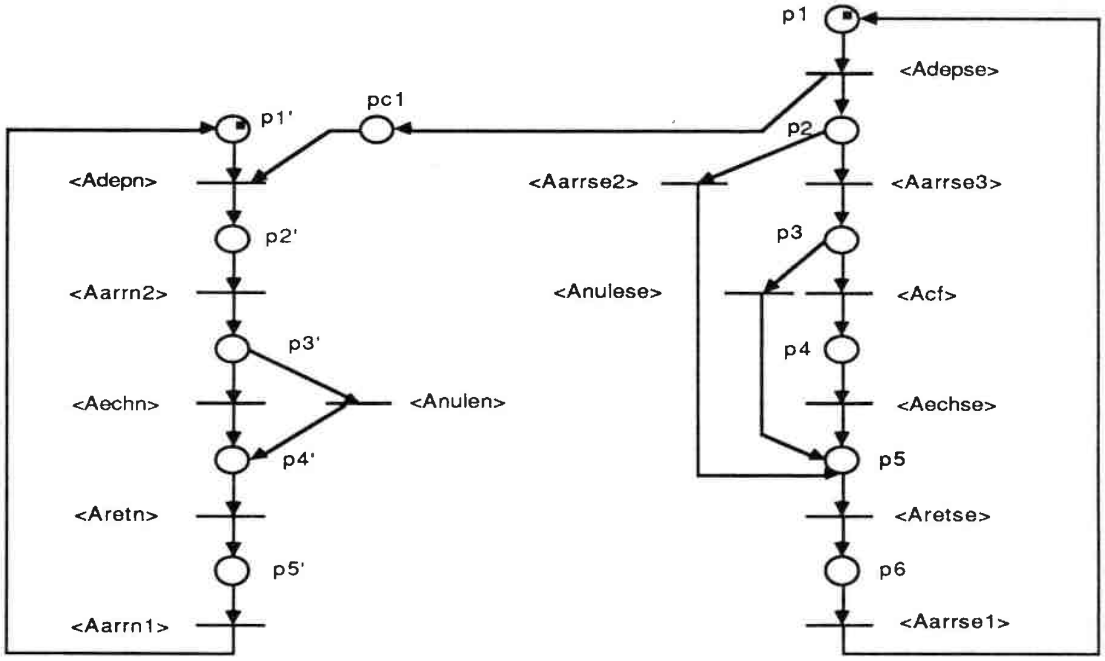
L'étape composer :

Dans cette étape nous définissons les contraintes de synchronisation entre les deux processus raffinés.

1) Le déclenchement du déplacement de la navette vers un bobinoir (Adepn), doit être précédé du déclenchement du déplacement du système embarqué (Adepse). Ces deux opérations doivent donc être synchronisées par :

PRE((Adepse ; Adepn) , 0)

On obtient le réseau et les invariants de la figure 6.16.



(a) Réseau RG1 E

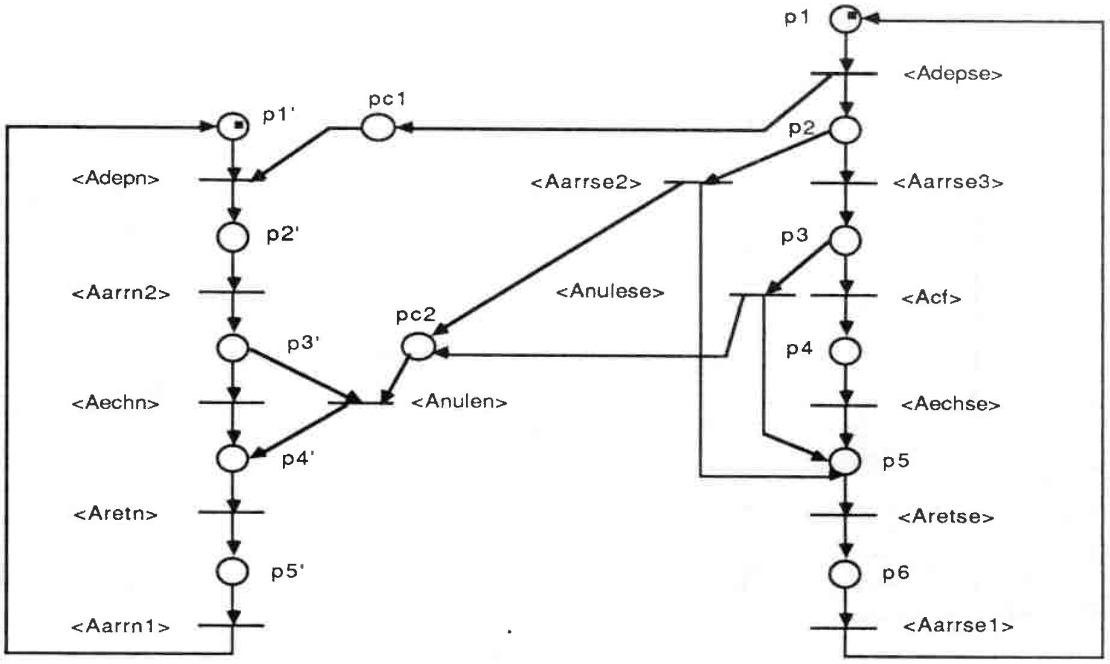
(b) Invariants :
M(p1)+M(p2)+M(p3)+M(p4)+M(p5)+M(p6) = 1
M(p1')+M(p2')+M(p3')+M(p4')+M(p5')+M(p6') = 1

Figure 6.18

2) Dans le même ordre d'idée, l'action de la navette liée à une casse de fil (Anulen) doit être précédée des actions du système embarqué liées également à une casse de fil (Aarrse2, Anulese). Ces opérations doivent donc être synchronisées par :

PRE (((Aarrse2 | Anulese) ; Anulen), 0)

on obtient le réseau et les invariants de la figure 6.19.



(a) Réseau $RG2_E$

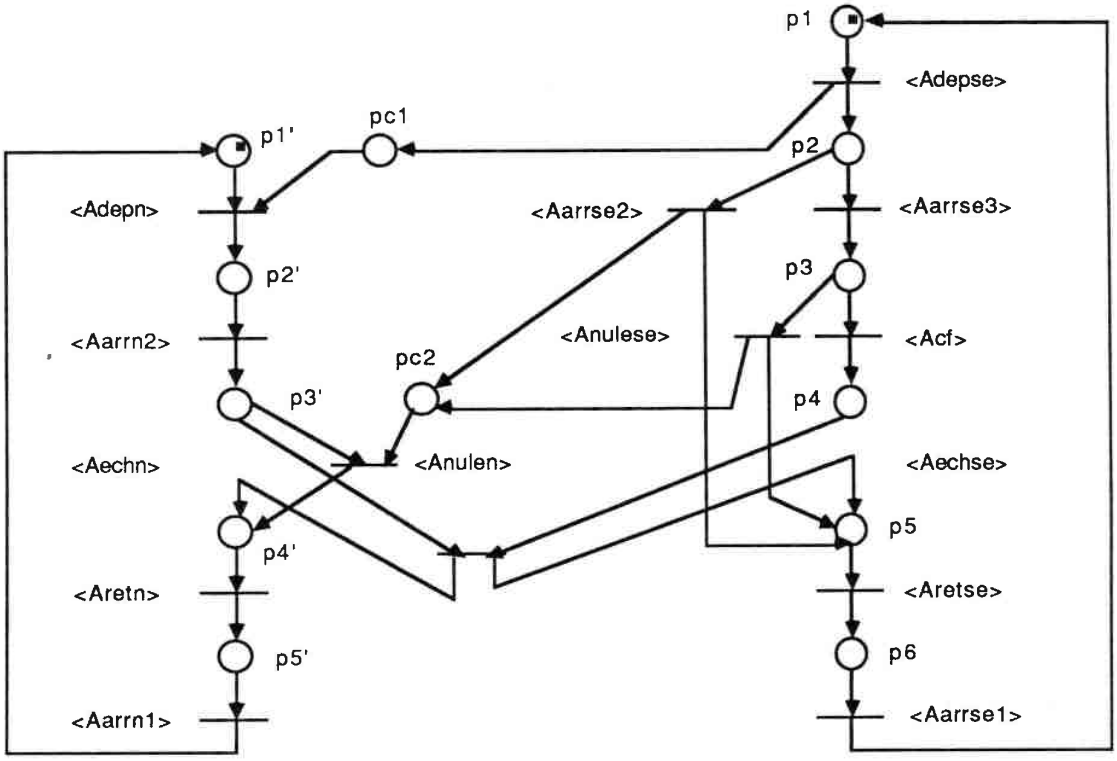
(b) Invariants :
M(p1')+M(p2')+M(p3')+M(p4')+M(p5') = 1
M(p1)+M(p2)+M(p3)+M(p4)+M(p5)+M(p6) = 1

Figure 6.19

3) Enfin, les opérations liées à l'échange de la bobine achevée (Aechse, Aechn) doivent avoir lieu en même temps dans les deux processus séquentiels. Ces deux opérations doivent donc être synchronisées par

SIM (Aechse || Aechn).

On obtient le réseau et les invariants de la figure 6.20.



(a) Réseau $RG3_E$

(b) Invariants :
M(p1')+M(p2')+M(p3')+M(p4')+M(p5') = 1
M(p1)+M(p2)+M(p3)+M(p4)+M(p5)+M(p6) = 1

Figure 6.20

Ce réseau définit la relation d'ordre que nous cherchions à caractériser. De ce fait, l'étape "Raffiner-A-Nouveau" n'est pas nécessaire dans cet exemple.

La prise en compte des synchronisations avec l'environnement

Pour chaque étiquette du réseau global $RG3_E$, nous définissons de la façon suivante l'opération de commande qui lui est associée, et le prédicat que l'état de l'environnement doit vérifier pour que cette opération soit déclenchable.

ETIQUETTE	*OPERATION ASSOCIEE*	*PREDICAT DE DECLENCHEMENT*
Adepse	Adepse := vrai	Ereq [i]
Aarrse3	Aarrse := vrai	↑ Efdepse [i] (τ2)
Aarrse2	Aarrse := vrai	Ecasse [i]
Aarrse1	Aarrse := vrai	↑ Efretse (τ2)
Acf	Acf := vrai	Efbob [i]
Aech	Aech := vrai	toujours vrai
Aretse	Aretse := vrai	toujours vrai
Anulese	NOP	Ecasse [i]
Adepn	Adepn := vrai	toujours vrai
Aarrn2	Aarren := vrai	↑ Efdepn [i] (τ2)
Aarrn1	Aarrn := vrai	↑ Efretn (τ2)
Aretn	Aretn := vrai	toujours vrai
Anulen	NOP	toujours vrai

En appliquant à partir de ces informations l'algorithme de transformation en un RdPI, on obtient le réseau de la figure 2.37 de la section 2.6.2. On remarquera que la transformation liée à l'opération Aech a necessité la modification du réseau par l'ajout d'une place substituable : la transition à laquelle cette opération est associée dans le réseau $RG3_E$ possède en sortie deux places dont aucune n'a été introduite par les primitives PRE et EXC. Par ailleurs, les transformations liées à l'action Aarrse2 à également nécessité une

modification du réseau par l'ajout d'une place substituable : la transition à laquelle l'opération est associée dans le réseau $RG3_E$ a en sortie une place unique non introduite par les primitives PRE et EXC. Cette place est également en sortie d'une autre transition.

L'analyse des propriétés du RdPI résultant

Les invariants de places qu'on obtient après les modifications de l'algorithme de transformation sont définis comme suit :

M (p1) + M (p2) + M (p3) + M (p4) + M (p5) + M (p6) = 1 (1)
M(p4) + M(p9) + M(p10) + M(p11) + M(p12) + M(p13) + M(p14) + M(p15) = 1 (2)

on peut noter que le prédicat Ecasse[i] est le seul événement synchronisé à plusieurs transitions. Pour que ces transitions soient simultanément validées, il faut que les places p10 et p11 soient simultanément marquées, ce qui est contradictoire avec l'invariant (2).

Par ailleurs, les seuls prédicats qui ne sont pas indépendants entre eux sont les prédicats Ereq[i] , et Efbob[i] liés au bobinage: on ne peut pas avoir une fin de bobinage sans avoir eu au préalable une requête. Pour que les transitions auxquelles ces deux prédicats sont associés puissent être simultanément validées, il faut que les places p9 et p11 soient simultanément marquées, ce qui est également contradictoire avec l'invariant (2).

Nous en déduisons alors que l'interprétation ne modifie pas les propriétés du réseau.

6.4 COMPARAISON AVEC D'AUTRES APPROCHES

Dans les principes, la démarche préconisée rejoint celles proposées dans [Berthelot 85 b] et [Guyot 87] . Dans chacune des trois démarches, on peut

distinguer une première étape où l'on décrit le problème à un certain niveau d'abstraction, et une deuxième étape où le réseau obtenu est raffiné exclusivement à l'aide de règles de transformation qui se fondent sur les règles de réduction.

Dans [Berthelot 85 b], la première étape sert à décrire le service que doit assurer un protocole de communication. Aucune aide n'est proposée à priori pour garantir par construction les bonnes propriétés de ce réseau qui constitue en fait le réseau embryon du processus de raffinement.

Dans [Guyot 87] tout comme dans notre cas, la première étape contraint le spécifieur à construire par raffinements successifs un réseau qui appartient à une classe pour laquelle le théorème de Commoner est applicable, ceci afin d'exploiter ce théorème pour garantir la vivacité du réseau. La différence entre les deux approches réside dans le choix de cette classe. Dans [Guyot 87], ce choix porte sur les graphes d'événements. Avec ce type de réseau, on ne peut exprimer ni choix, ni conflit d'accés à une ressource partagée. En fait, il n'est particulièrement bien adapté que pour décrire des systèmes d'ordonnancement de tâches qui coopèrent sans conflit. Dans notre cas, nous avons retenu les graphes simples. Cette classe inclut les graphes d'événements et permet de mieux modéliser la coopération entre des tâches parallèles. De ce fait, elle est bien plus intéressante pour décrire le comportement d'un système de commande.

Par ailleurs, il est important de noter que si nous levons les contraintes d'application D6.6 (2) de la primitive PRE et D6.7 (2) de la primitive SIM qui servent à restreindre dans la première phase le réseau en construction à un graphe simple, nous obtenons un ensemble de primitives qui permettent de construire une classe très large de réseaux. Les invariants que nous pouvons déterminer permettent de garantir sans grands frais les propriétés du système liées à la sûreté de fonctionnement. Si le réseau est borné, sa vivacité peut être décidée par construction du graphe des marquages accessibles, sinon, il faut recourir à la simulation. Ces différentes techniques de validation sont

complémentaires et peuvent être gérées de façon cohérente par un outil d'aide à la conception.

6.5 LES EXTENSIONS POSSIBLE

L'organisation des ateliers de production ne conduit malheureusement pas qu'à des processus séquentiels qui coopèrent par synchronisation directe. Les problèmes de transport des pièces qui sont des problèmes très importants ne relèvent pas par exemple de ce type de modélisation. Nous étudions ici, quelques extensions possibles de la démarche présentée, qui permettent de couvrir une classe plus importante de situations.

6.5.1 LA MODELISATION DE L'ETAT DE L'ENVIRONNEMENT A L'AIDE D'UN AUTOMATE

Pour décrire l'état dans lequel doit être l'environnement pour qu'un ordre de commande puisse être déclenché, nous avions choisi de nous appuyer sur les notions de conditions, d'événements et de politique de consommation des occurrences d'événements. Il est des cas où l'état que doit vérifier l'environnement se décrit plus simplement en caractérisant l'ordre d'occurrence de certains événements. Par exemple, dans l'atelier de bobinage, si nous supposons que du fait des choix technologiques les variables d'entrée Ereq[i], Ecasse[i] et Efbob[i] ne doivent pas être considérées comme des variables binaires non fugitives, mais plutôt comme des variables binaires fugitives, la machine à états de la figure 6.21 permet de caractériser l'état d'un bobinoir en caractérisant l'ordre dans lequel les événements ↑Ereq, ↑Efbob et ↑Ecasse peuvent se produire.

La condition de déclenchement de l'opération "Adepse : = vrai" peut se définir alors comme un test à 1 de la place p_3 de ce réseau. Ainsi, on s'assure que le système embarqué ne démarrera que si une requête a été émise et si entre temps une casse de fil n'a pas eu lieu. En appliquant successivement la règle de

réduction d'anneau [Boussin 78] , (cf. figure 6.22), et la suppression de transition neutre sur ce réseau, on peut le réduire en une place qui est une place implicite qui peut donc être supprimée à son tour. Le test à 1 introduit ne modifie donc pas les propriétés du réseau initial.

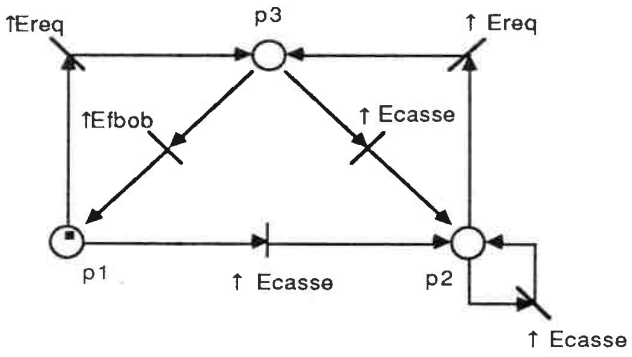


Figure 6.21

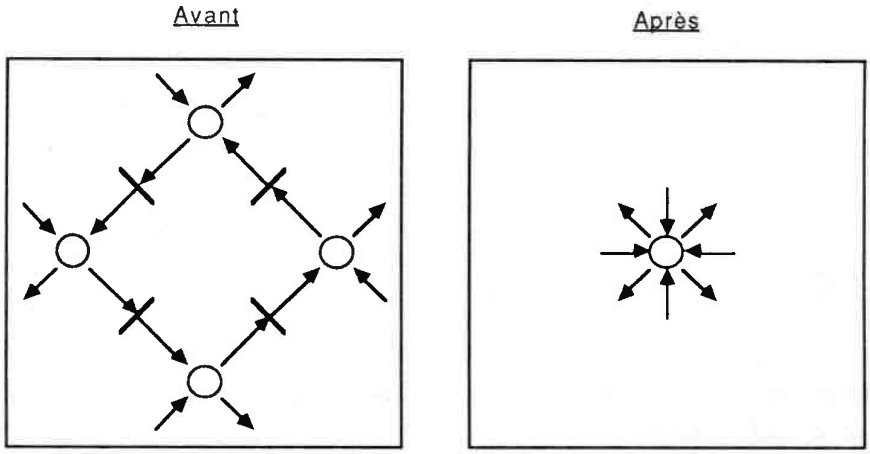


Figure 6.22

On remarquera que dans la spécification de la pompe de Kramer (cf. section 2.6.1), ce mode de définition de la synchronisation des tirs d'une transition à l'évolution de l'environnement a été utilisé. En fait, le réseau

obtenu peut également être construit en utilisant notre démarche.

6.5.2 LA COMPOSITION DE RESEAUX PAR FUSION DE PLACES

Dans [Valette 85 a] , on montre comment construire et valider un réseau de façon ascendante à partir de réseaux plus élémentaires ayant de bonnes propriétés. Cette méthode est particulièrement bien adaptée pour modéliser la commande de chariots filoguidés.

Le réseau de transport est subdivisé en sections. Des capteurs permettent de détecter les fins de courses dans chaque section. Pour éviter les collisions, à tout instant on ne doit avoir au plus qu'un seul chariot présent dans une section. Le réseau de la figure 6.23 décrit le contrôle d'une section.

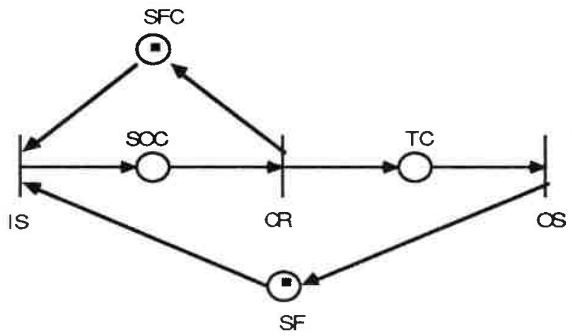


Figure 6.23

Le tir de la transition IS correspond à l'entrée d'un chariot dans la section, le tir de la transition CR à la détection de la fin de course dans la section, et le tir de la transition OS à la sortie du chariot de la section.

Par ailleurs, le marquage de la place SOC signifie qu'un chariot est en déplacement dans la section, le marquage de la place SF que la section est libre, et le marquage de la place SFC qu'aucun chariot n'est en déplacement dans la section.

Ce réseau est vivant et admet les invariants de places suivants :

$$M(SOC) + M(TC) + M(SF) = 1$$
$$M(SOC) + M(SFC) = 1.$$

Le réseau global modélisant la commande du réseau de transport s'obtient en composant les réseaux élémentaires correspondant à la commande des différentes sections. Par exemple, la commande d'un croisement est décrit par le réseau de la figure 6.24, qui a été obtenu en fusionnant les places SFC.

On montre que lorsqu'on compose deux réseaux de cette façon, les invariants de places qui ne contiennent pas de place concernée par la composition sont tous préservés.

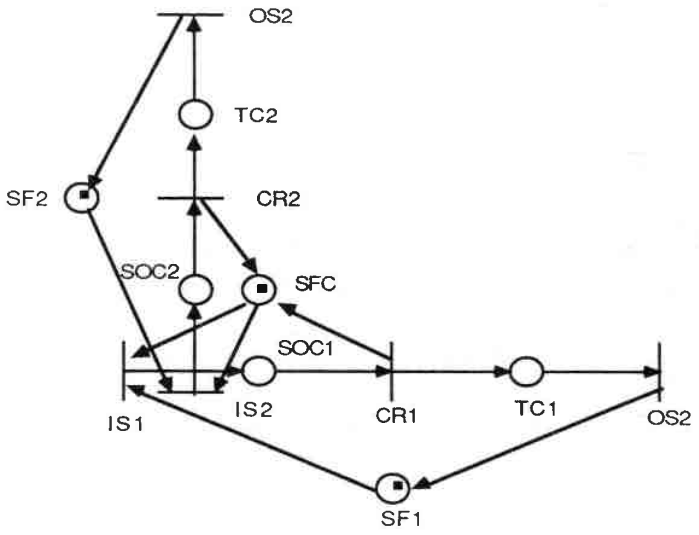


Figure 6.24

Par ailleurs, lorsque l'on compose deux réseaux, si deux invariants de places sont tels qu'une place de l'un des invariants est fusionné avec une autre place de même poids de l'autre invariant, et toutes les autres places des deux

invariants ne sont pas concernés par la fusion, alors la concaténation des deux invariants est un invariant du réseau résultant.

Ces deux propriétés permettent de calculer les invariants de places suivants pour le réseau de la figure 6.24 :

$$M(SOC1) + M(SFC) + M(SOC2) = 1$$
$$M(SOC1) + M(TC1) + M(SF1) = 1$$
$$M(SOC2) + M(TC2) + M(SF2) = 1.$$

La première équation permet de déduire qu'après composition, le comportement décrit garantit qu'on ne peut avoir au plus qu'un chariot dans une section.

Le réseau de la figure 6.23, et ses invariants peuvent être construits par raffinements successifs à l'aide de nos primitives comme indiqué dans la figure 6.25.

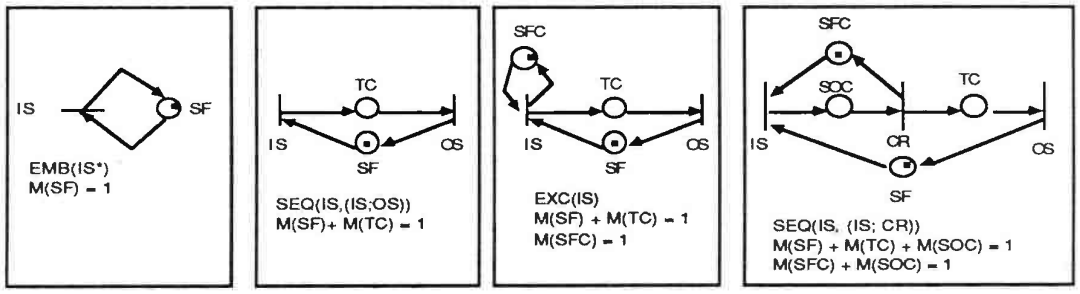


Figure 6.25

En incluant dans notre démarche une primitive de composition par fusion de places telle que nous venons de le voir, on voit bien qu'on se donne un moyen supplémentaire pour construire rigoureusement d'autres types de réseaux, tels ceux décrivant la commande de chariots filoguidés.

6.5.3 COMPOSITION PAR FUSION DE TRANSITIONS ET SUBSTITUTION DE SOUS-RESEAUX

Mentionnons enfin l'existence de deux autres théorèmes importants pour la synthèse de réseaux par composition présentés dans [André 81]. Ces deux théorèmes concernent la composition de réseaux par fusion de transitions et la substitution de sous-réseau. Ces deux opérations sont particulièrement intéressantes lorsque les réseaux concernés ont même abstraction sur leurs transitions frontières. Dans ce cas, leur comportement est préservé.

On peut par exemple, interconnecter la spécification d'un processus séquentiel producteur de pièces, et la spécification d'un processus séquentiel consommateur de pièces, en substituant dans un réseau décrivant les contraintes de synchronisation de ces deux processus, un sous-réseau par un autre décrivant un système de convoyage (cf. figure 6.26).

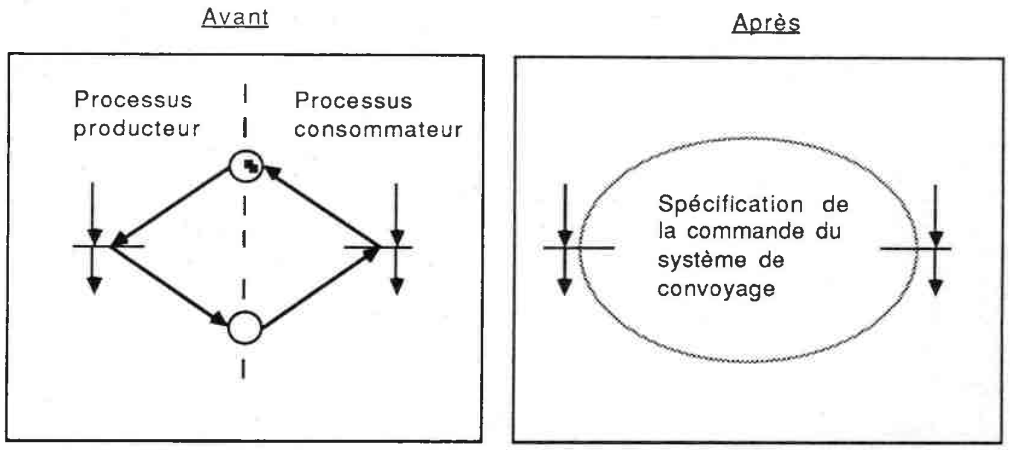


Figure 6.26

Dans un environnement d'aide à la conception, il peut alors être intéressant de constuire une bibliothèque de réseaux embryons qui décrivent différents types de files FIFO que l'on valide préalablement. Ces embryons peuvent ensuite être raffinés pour décrire la commande de systèmes de

convoyage spécifiques, et interconnectés de cette façon à la spécification de processus séquentiels.

Chapitre 7

AIDE A LA CONCEPTION DU COMPORTEMENT INTERNE

PLAN

7.0 INTRODUCTION

Dans le chapitre 8, nous avons présenté une démarche méthodique qui permet de constuire à l'aide d'un RdPI la spécification du comportement externe d'un système de commande. Dans cette démarche, l'une de nos préoccupations essentielle a été de garantir par construction les bonnes propriétés des réseaux que le spécifieur construit, de manière à lui éviter ou à lui simplifier les étapes délicates de validation.

Dans ce chapitre, nous abordons avec le même état d'esprit la phase de conception du comportement interne. Par rapport à notre démarche globale, cette phase concerne le passage du niveau conceptuel au niveau organisationnel. Son but est donc de déterminer l'architecture interne qui facilitera le mieux, ou qui rendra possible l'implémentation ultérieure du système de commande sur un réseau de processeurs.

Elle conduit pour l'essentiel à regrouper dans un même module, en fonction de différentes contraintes, (contraintes de temps, contraintes géographiques, contraintes de robustesse, etc ...), les prises en compte d'événements externes (i.e les entrées), et les déclenchements d'opérations de commande (i.e les sorties), qu'il est important de ne pas dissocier, et à définir le comportement des différents modules ainsi constitués.

La difficulté majeure de cette phase de conception réside dans la définition du comportement des entités modulaires. En effet, par définition, ces entités ne peuvent partager de données communes. De ce fait, les décisions prises dans un module ne peuvent tenir compte que d'un état partiel du système de commande. Pour que chaque module puisse disposer d'informations suffisantes pour assurer les mêmes interactions avec l'environnement que celles définies par la spécification du comportement externe, il est donc indispensable de modifier cette spécification pour prendre en compte en plus la coopération inter-modulaire.

Notre objectif est de décharger le concepteur de cette tâche, et de lui éviter par la même occasion la nécessité de vérifier la cohérence entre le comportement externe et le comportement interne de son système. Pour cela, nous nous appuyons sur trois approches de décomposition d'un réseau en sous-réseaux. Ces trois approches offrent différentes possibilités de regroupement des entrées/sorties d'un système de commande dans des sous-réseaux qui communiquent par échange de messages pour assurer le même comportement que le réseau initial. Ce sont :

- l'approche de décomposition par partitionnement
- l'approche de décomposition par dédoublement
- et l'approche de décomposition par substitution de blocs bien formés.

Les deux premières approches permettent de concevoir une architecture où le contrôle est décentralisé entre les différents modules, et la troisième approche de hiérarchiser le contrôle pour distinguer par exemple un niveau coordination et un niveau commande locale. Pour chacune de ces approches, nous proposons des règles de décomposition qui permettent de reconstruire de façon systématique les différents sous-réseaux, lorsque les choix de regroupement des entrées/sorties sont connus. De façon générale, ces règles conduisent à une duplication plus ou moins importante d'information sur l'état du système dans les différents sous-réseaux, et à la mise en place d'un protocole de coopération par échanges de messages pour la mise à jour de cette information. La complexité des modifications qui peuvent ainsi être introduites est en rapport avec les liens temporels qui doivent exister entre les entrées/sorties réparties dans les différents modules. Plus ces liens seront forts, plus les modifications introduites seront complexes.

Ainsi, mis ensemble, les primitives de raffinement et les règles de décomposition permettent à un concepteur de construire des systèmes complexes et sûrs de fonctionnement, sans passer nécessairement par des étapes de validation.

Dans ce qui suit, nous présentons successivement les règles de décomposition pour le partionnement (section 7.1), le dédoublement (section 7.2), et la substitution de sous-réseaux (section 7.3). Ensuite, nous aborderons la prise en compte des problèmes de présentation des sous-réseaux résultants (section 7.4). Un algorithme de décomposition fondé sur ces différents aspects sera alors présenté (section 7.5). Nous terminerons le chapitre en illustrant la démarche sur des exemples (section 7.6), et en la comparant à une approche similaire proposée dans [Valette 82] [Courvoisier 83], (section 7.7).

Pour simplifier l'exposé, pour chacune des règles que nous présenterons, nous considérerons que la décomposition conduit à deux sous-réseaux. Le recours aux liaisons divergentes et convergentes (c.f. sections 2.5.2 et 2.5.3) permet de généraliser ces règles, et de réaliser des décompositions en autant de sous-réseaux que nécessaire, sans introduire de complication supplémentaire.

7.1 L'APPROCHE DE DECOMPOSITION PAR PARTITIONNEMENT

C'est l'approche qui convient le mieux lorsque les contraintes de structuration permettent de garder le plus possible groupé les entrées/sorties logiquement dépendantes. Elle conduit le concepteur à partitionner le réseau qui décrit le comportement externe de son système en différents sous-réseaux par rapport à des **transitions frontières**. Ce partitionnement précise d'une part les transitions qui doivent être entièrement incluses dans un sous-réseau, et d'autre part, pour chaque transition frontière, la répartition dans les différents sous-réseaux des places en entrée et des places en sortie, et l'affectation du prédicat associé.

Dans ce qui suit, nous passons en revue différents schémas de décomposition possibles d'une transition frontière. Nous verrons que pour certains schémas de décomposition, il est en pratique impossible de mettre en place un protocole de synchronisation qui préserve le comportement externe initialement spécifié, sans une vue globale de l'état du système. De ce fait, ces schémas constituent un guide pour le concepteur. Ils indiquent les contraintes à

respecter dans un réseau pour rendre possibles certains choix d'organisation.

Pour faciliter la lecture, nous présenterons chaque fois la formalisation des règles de décomposition de façon textuelle et graphique. La formalisation graphique s'appuie sur les conventions suivantes :

1) Les choix de décomposition au niveau d'une transition frontière sont exprimés à l'aide d'un trait vertical. Nous supposerons que les places qui apparaîssent à gauche de ce trait sont affectées au sous-réseau R_I' et les places qui apparaîssent à droite au sous-réseau R_I''. Par ailleurs, si le prédicat (noté r_i) associé à la transition frontière apparaît à gauche du trait, nous considérerons qu'il est affecté au sous-réseau R_I', et s'il apparaît à droite qu'il est affecté au sous-réseau R_I''.

2) Les places en entrée auxquelles est associé un arc pendant indiquent la présence de places en entrée de plusieurs transitions. Les places en entrée auxquelles est associé un arc pendant marqué d'une croix indiquent l'absence de places partagées par plusieurs transitions. Et enfin les places en entrée auxquelles aucun arc pendant n'est associé indiquent la présence de places partagées par plusieurs transitions ou non.

7.1.1 LE CAS DES TRANSITIONS NON FRONTIERES

Ce cas correspond à la règle la plus triviale. Elle consiste tout simplement à inclure entièrement dans un sous-réseau une transition avec ses places en entrée, ses places en sortie et le prédicat qui lui est associé.

Règle RD01

Définition D.7.1 :

Soit R_I un RdPI à décomposer par partitionnement en deux réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' = < P'', T'' ; Pre'', Post'' >$.

Les transformations correspondant au choix de décomposition qui consiste à inclure entièrement dans un des deux sous-réseaux, disons R_I', toute transition $t_x \in T$ avec ses places d'entrée, ses places de sortie, et son prédicat associé, sont définies comme suit :

(1) $T' \leftarrow T' + \{ t_x \}$ (*La transition* t_x *est incluse dans* T')

(2) $P' \leftarrow P' + (\cdot t_x + t_x \cdot)$

(*Les places en entrée et en sortie de la transition* t_x *sont incluses dans* P')

(3) $\forall p \in P' : Pre' (p, t_x) = Pre (p, t_x)$ si $p \in P$ 0 sinon.

(*La transition* t_x *conserve les mêmes places en entrée dans* R *et* R').

(4) $\forall p \in P' : Post' (p, t_x) = Post (p, t_x)$ si $p \in P$ 0 sinon.

(*La transition* t_x *conserve les mêmes places en sortie dans* R *et* R')

(5) $\Psi' (t_x) = \Psi (t_x)$

(*La transition* t_x *conserve la même interprétation dans* R_I *et* R_I'

(6) $\forall p \in (\cdot t_x + t_x \cdot) : \Phi' (p) = \Phi (p)$

(*Les places en entrée et en sortie de la transition* t_x *conservent la*

même interprétation dans R_I *et* R_I'). ⊗

La figure 7.1 schématise graphiquement cette règle.

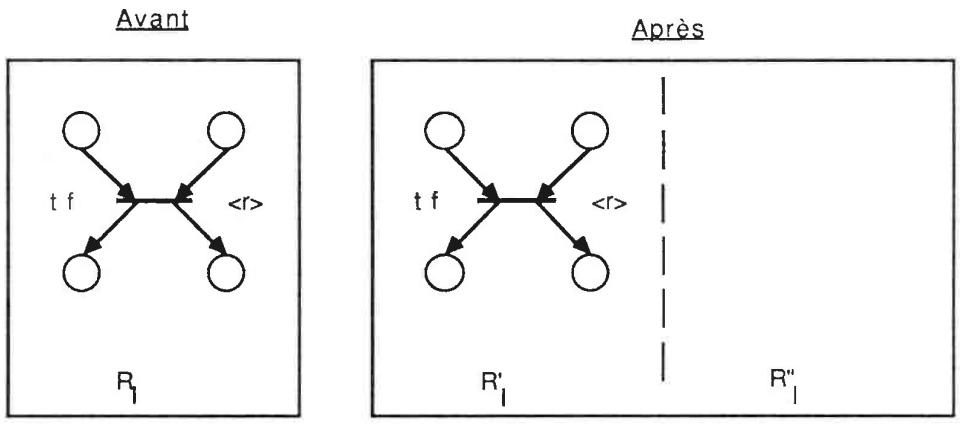


Figure 7.1

7.1.2 LES TRANSITIONS FRONTIERES NON EN CONFLIT STRUCTUREL AVEC D'AUTRES TRANSITIONS FRONTIERES

Nous regroupons dans cette section la présentation des schémas de décomposition des transitions frontières qui ne sont pas en conflit structurel avec d'autres transitions frontières.

a) La règle RD02

Cette règle concerne les transitions frontières étiquetées par le prédicat pr_0 (i.e. le prédicat toujours vrai).

Définition D 7.2 :

Soit R_I un RdPI à décomposer par partitionnement en deux réseaux R_I' et

R_I" :

R_I = < R ; V ; OP ; PR ; Φ ; Ψ > avec R = < P, T ; Pre, Post >

R_I' = < R' ; V' ; OP' ; PR' ; Φ' ; Ψ' > avec R' = < P', T' ; Pre', Post' >

R_I" = < R" ; V" ; OP" ; PR" ; Φ" ; Ψ" > avec R" = < P", T" ; Pre", Post" >.

Pour toute transition frontière $t_f \in T$ telle que :

(1) $\Psi(t_f) = pr_0$ (*Le prédicat associé à* t_f *est le prédicat toujours vrai*)

(2) $^{\cdot}t_f = E'_f + E"_f$, $E'_f \cap E"_f = \varnothing$

(E'_f et $E"_f$ *définissent un partitionnement des places en entrée de* t_f *en deux sous-ensembles disjoints*).

(3) $t_f^{\cdot} = S'_f + S"_f$, $S'_f \cap S"_f = \varnothing$

(S'_f et S_f" *définissent un partitionnement des places en sortie de* t_f *en deux sous-ensembles disjoints*).

(4) $E_f' = \varnothing$ ou $E_f" = \varnothing \Rightarrow S_f' \neq \varnothing$ et $S_f" \neq \varnothing$

(E_f' *et* S_f' *ne sont pas simultanément vides et* $E_f"$ *et* $S_f"$ *non plus*)

(5) $\forall p \in {}^{\cdot}t_f, \forall t \in T : t \neq t_f$ et $Pre(p, t) > 0 \Rightarrow$ t n'est pas une transition frontière

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

Les transformations correspondant au choix de décomposition qui consiste à affecter les places contenues dans E_f' et S_f' au sous-réseau R_I' et les places contenues dans $E_f"$ et $S_f"$ au sous-réseau R_I" sont définies comme suit :

Dans R_I'

(6) T' <- T' + { $\$t_x$ } (*Dans* T' *on inclut une nouvelle transition* $\$t_x$)

(7) P' <- P' + E_f' + S_f' (*Dans* P' *on inclut les places contenues dans* E_f' *et* S_f')

(8) $\forall\ p \in P' : Pre'(p, \$t_x) = Pre(p, t_f)$ si $p \in E_f'$ et 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en entrée les places contenues dans* E_f').

(9) $\forall\ p \in P' : Post'(p, \$t_x) = Post(p, t_f)$ si $p \in S_f'$ et 0 sinon.

(*La nouvelle transition* $\$t_x$ *reçoit en sortie les places contenues dans* S'_f).

Dans R_I''

(10) $T'' \leftarrow T'' + \{ \underline{\$t_x} \}$ (*Dans* T" *on inclut une nouvelle transition* $\underline{\$t_x}$)

(11) $P'' \leftarrow P'' + E_f'' + S_f''$ (*Dans* P" *on inclut les places contenues dans* E_f'' *et* S_f'')

(12) $\forall\ p \in P'' : Pre(p, \underline{\$t_x}) = Pre(p, t_f)$ si $p \in E_f''$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en entrée les places contenues dans* E_f'')

(13) $\forall\ p \in P'' : Post''(p, \underline{\$t_x}) = Post(p, t_f)$ si $p \in S_f''$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en sortie les places contenues dans* S_f'').

Dans R_I' *et* R_I''

(14) $\Psi'(\$t_x) = ?M_x$, $\Psi''(\underline{\$t_x}) = !\ M_x$ (*Un rendez-vous est défini entre* $\$t_x$ *et* $\underline{\$t_x}$)

(15) $\forall\ p \in E_f' + S_f' : \Phi'(p) = \Phi(p)$

(*Les places contenues dans* E'_f *et* S'_f *conservent leur interprétation dans* R_I')

(16) $\forall\ p \in E_f'' + S_f'' : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* E_f'' *et* S_f'' *conservent leur interprétation dans* R_I'').

La formalisation graphique de cette règle est décrite par la figure 7.2.

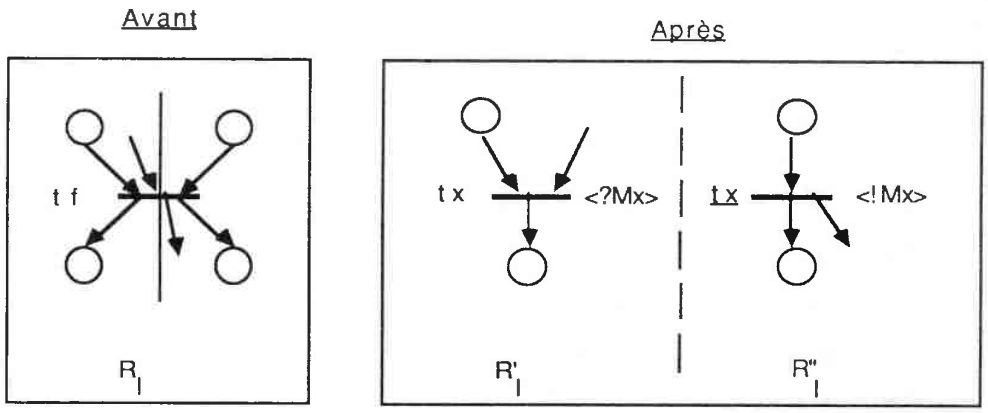


Figure 7.2

(Validation en annexe C, chapitre C.1)

b) La règle RD03

Cette règle concerne les transitions frontières étiquetées par un prédicat différent de pr_0 et n'ayant en entrée que des places non partagées.

Définition D7.3 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et $R_I"$:

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I" = < R" ; V' ; OP" ; PR" ; \Phi" ; \Psi" >$ avec $R' = < P", T" ; Pre" ; Post" >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) $\Psi (t_f) \neq pr_0$

(*Le prédicat associé à* t_f *est différent du prédicat toujours vrai*)

(2) $\cdot t_f = E_f' + E_f''$, $E_f' \cap E_f'' = \varnothing$, $|E_f'| > 0$, $|E_f''| > 0$

(E_f' *et* E_f'' *définissent un partitionnement des places en entrée de* t_f *en deux sous-ensembles disjoints*)

(3) $t_f \cdot = S_f' + S_f''$, $S_f' \cap S_f'' = \varnothing$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de* t_f *en deux sous-ensembles disjoints*)

(4) $\forall p \in E_f' + E_f'' : p\cdot = \{ t_f \}$

(*La seule transition en sortie des places contenues dans* E_f' *et* E_f'' *est* t_f).

Les transformations correspondant en choix de décomposition qui consiste à associer les places contenues dans E_f' et S_f' au sous-réseau R_I', les places contenues dans E_f'' et S_f'' aux sous-réseaux R_I'', et le prédicat défini par Ψ (t_f) à l'un des deux sous-réseaux (disons R_I') sont définies comme suit :

Dans R_I'

(5) $T' \leftarrow T' + \{ \$t_x, \$t_y \}$

(*Dans* T' *on inclut deux nouvelles transitions* $\$t_x$ *et* $\$t_y$)

(6) $P' \leftarrow P' + E_f' + S_f' + \{ \$p_y \}$

(*Dans* P' *on inclut les places contenues dans* E_f', S_f' *et une nouvelle place* $\$p_y$)

(7) $\forall p \in P' : Pre'(p, \$t_x) = Pre(p, t_f)$ si $p \in E_f'$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en entrée les places contenues dans* E_f')

(8) $\forall p \in P' : Post(p, \$t_x) = 1$ si $p = \$p_y$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en sortie la nouvelle place* $\$p_y$)

(9) $\forall p \in P' : Pre'(p, \$t_y) = 1$ si $p = \$p_y$ 0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en entrée la nouvelle place* $\$p_y$)

(10) $\forall\ p \in P' : Post'(p, \$t_y) = Post(p, t_f)$ si $p \in S_f'$ 0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en sortie les places contenues dans* S_f').

Dans R_I''

(11) $T'' \leftarrow T'' + \{ \underline{\$t_x} + \underline{\$t_y} \}$

(*Dans T" on inclut deux nouvelles transitions* $\underline{\$t_x}$ *et* $\underline{\$t_y}$)

(12) $P'' \leftarrow P'' + E_f'' + S_f'' + \{ \underline{\$p_y} \}$

(*Dans P" on inclut les places contenues dans* E_f'', S_f'' et une nouvelle place $\underline{\$p_y}$)

(13) $\forall\ p \in P'' : Pre''(p, \underline{\$t_x}) = Pre(p, t_f)$ si $p \in E_f''$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en entrée les places contenues dans* E_f'')

(14) $\forall\ p \in P'' :$ $Post''(p, \underline{\$t_x}) =$ 1 si $p = \underline{\$p_y}$ et 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en sortie la nouvelle place* $\underline{\$p_y}$)

(15) $\forall\ p \in P'' : Pre''(p, \$t_y) = 1$ si $p = \$p_y$ et 0 sinon

(*La nouvelle transition* $\underline{\$t_y}$ *reçoit en entrée la nouvelle place* $\underline{\$p_y}$)

(16) $\forall\ p \in P'' : Post''(p, \underline{\$t_y}) = Post(p, t_f)$ si $p \in S_f''$ et 0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en sortie les places contenues dans* S_f'').

Dans R_I' *et* R_I''

(17) $\Psi'(\$t_x) = ?M_x$, $\Psi''(\$t_x) = !\ M_x$

(*On définit une synchronisation par rendez-vous entre* $\$t_x$ *et* $\underline{\$t_x}$)

(18) $\Psi'(\$t_y) = \Psi(t_f) \rightarrow ?\ M_y$, $\Psi''(\underline{\$t_y}) = !\ M_y$

(*On associe le prédicat définit par* $\Psi(t_f)$ *à la transition* $\$t_y$ *et on définit*

une synchronisation par rendez-vous entre $\$t_y$ *et* $\underline{\$t_y}$).

(19) $\forall\ p \in E_f' + S_f' : \Phi'(p) = \Phi(p)$

(*Les places contenues dans* E_f' et S_f' *conservent la même interprétation dans* R_I *et* R_I')

(20) $\forall\ p \in (E_f'' + S_f'') : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* E_f'' *et* S_f'' *conservent la même interprétation dans* R_I *et* R_I'').

(21) $\Phi'(\$p_y) = \Phi''(\underline{\$p_y}) = NOP.\ \otimes$

Le schéma de décomposition et les règles de transformation de la règle RD03 sont formalisés graphiquement dans la figure 7.3.

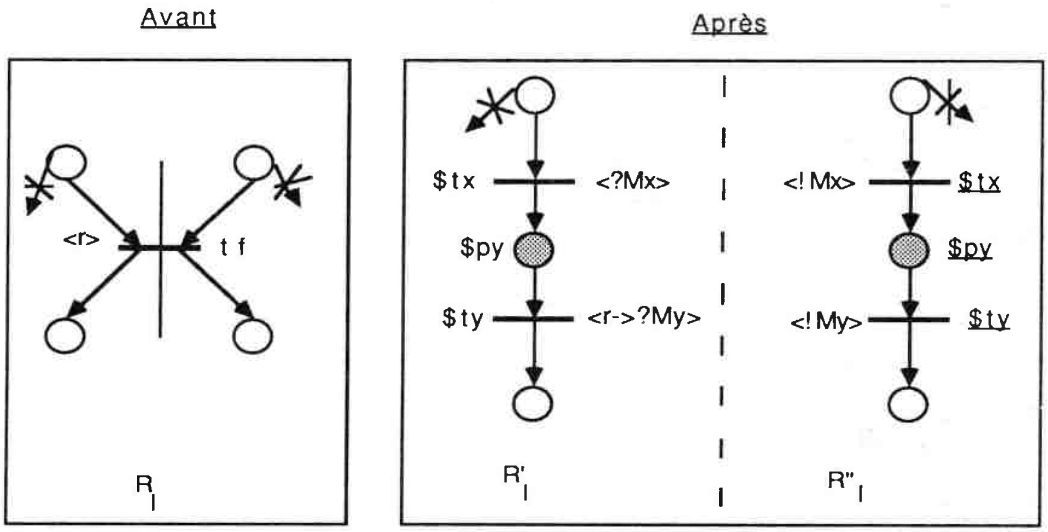


Figure 7.3

Le premier rendez-vous permet au sous-réseau R_I' de s'assurer que la précondition définie par les places d'entrée de la transition t_f est devenue vraie.

Ces places n'étant pas partagées, cette information sera encore vraie lorsque le prédicat r deviendra vrai et lorsque le deuxième rendez-vous aura lieu.

(Validation en annexe C, chapitre C.2)

c) La règle RD04

La règle RD04 se distingue de la règle RD03 par le fait que le sous-ensemble de places affecté à l'un des deux sous-réseaux (disons $R_{I'}$) contient des places partagées.

Définition 7.4 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I''.

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I'' = < R'' ; V' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R' = < P'', T'' ; Pre'' ; Post'' >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) $\Psi (t_f) \neq pr_0$

(*Le prédicat associé à* t_f *est différent du prédicat toujours vrai*).

(2) $t_f = E_f' + E_f''$, $E_f' \cap E_f'' = \varnothing$, $|E_f'| \geq 0$, $|E_f''| > 0$

(E_f' *et* E_f'' *définissent un partitionnement des places en entrée de la transition* t_f *en deux sous-ensembles disjoints*).

(3) $t_f' = S_f' + S_f''$, $S_f' \cap S_f'' = \varnothing$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de la transition* t_f *en deux sous-ensembles disjoints*).

(4) $\exists p \in E_f'$, $\exists t \in T$ telles que : Pre (p, t) > 0 et $t \neq t_f$

(*Dans* E_f' *il existe au moins une place partagée*)

(5) $\forall p \in E_f''$: $p^{\cdot} = \{ t_f \}$

(*La seule transition en sortie des places contenues dans* E_f'' *est* t_f).

(6) $\forall p \in {}^{\cdot}t_f$, $\forall t \in T : t \neq t_f$ et Pre (p, t) > 0 $\Rightarrow$ t n'est pas une transition frontière

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

les transformations correspondant au choix de décomposition qui consiste à affecter les places contenues dans E_f' et S_j' au sous-réseau R_I', les places contenues dans E_f'' et S_f'' au sous-réseau R_I'', et le prédicat défini par $\Psi(t_f)$ au sous-réseau R_I' sont définies comme suit :

Dans R_I'

(7) T' <- T' + { $\$t_x$, $\$t_y$ }

(*Dans* T' *on inclut deux nouvelles transitions* $\$t_x$ *et* $\$t_y$)

(8) P' <- P' + E_f' + S_f' + { $\$p_x$, $\$p_y$ }

(*Dans* P' *on inclut les places contenues dans* E_f', S_f' *et deux nouvelles places* $\$p_x$ *et* $\$p_y$)

(9) M_0' ($\$p_x$) = 1 (*On marque initialement la place* $\$p_x$ *à* 1)

(10) $\forall p \in P'$: Pre' (p, $\$t_x$) = 1 si p = $\$p_x$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en entrée la nouvelle place* $\$p_x$)

(11) $\forall p \in P'$: Post' (p, $\$t_x$) = 1 si p = $\$p_y$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en sortie la nouvelle place* $\$p_y$)

(12) $\forall p \in P'$: Pre' (p, $\$t_y$) = Pre (p, t_f) si $p \in E_f'$

1 si $p = \$p_y$

0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en entrée les places contenues dans* E_f' *et la nouvelle place* $\$p_y$)

(13) $\forall p \in P' : Post'(p, \$t_y)$: $Post(p, t_f)$ si $p \in S_f'$

1 si $p = \$p_x$

0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en sortie les places contenues dans* S_f' *et la nouvelle place* $\$p_x$)

Dans R_I"

(14) $T" \leftarrow T" + \{ \underline{\$t_x}, \underline{\$t_y} \}$

(*Dans T" on inclut deux nouvelles transitions* $\underline{\$t_x}$ et $\underline{\$t_y}$)

(15) $P" \leftarrow P" + E_f" + S_f" + \{ \underline{\$p_y} \}$

(*Dans P" on inclut les places contenues dans* $E_f"$, $S_f"$ *et une nouvelle place* $\underline{\$p_y}$)

(16) $\forall p \in P" : Pre"(p, \underline{\$t_x}) = Pre(p, t_f)$ si $p \in E_f"$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en entrée les places contenues dans* $E_f"$)

(17) $\forall p \in P" : Post"(p, \underline{\$t_x}) = 1$ si $p = \underline{\$p_y}$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en sortie la nouvelle place* $\underline{\$p_y}$)

(18) $\forall p \in P" : Pre"(p, \underline{\$t_y}) =$ 1 si $p = \underline{\$p_y}$ 0 sinon

(*La nouvelle transition* $\underline{\$t_y}$ *reçoit en entrée la nouvelle place* $\underline{\$p_y}$)

(19) $\forall p \in P" : Post"(p, \underline{\$t_y}) = Post(p, t_f)$ si $p \in S_f"$ 0 sinon

(*La nouvelle transition* $\underline{\$t_y}$ *reçoit en sortie les places contenues dans* $S_f"$).

Dans R_I' *et* R_I''

(20) $\Psi'(\$t_x) = ?M_x$, $\Psi''(\underline{\$t_x}) = !\,M_x$

(*On définit une synchronisation par rendez-vous entre* $\$t_x$ *et* $\underline{\$t_x}$)

(21) $\Psi'(\$t_y) = \Psi(t_f) \rightarrow ?\,M_y$, $\Psi''(\underline{\$t_y}) = !\,M_x$

(*On associe le prédicat défini par* $\Psi(t_f)$ *à la transition* $\$t_y$, *et on définit une synchronisation par rendez-vous entre* $\$t_y$ *et* $\underline{\$t_y}$)

(22) $\forall\, p \in E_f' + S_f' : \Phi'(p) = \Phi(p)$

(*Les places contenues dans* E_f' *et* S_f' *conservent la même interprétation dans* R_I *et* R_I')

(23) $\forall\, p \in E_f'' + S_f'' : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* E_f'' *et* S_f'' *conservent la même interprétation dans* R_I' *et* R_I'')

(24) $\Phi'(\$p_x) = \Phi'(\$p_y) = \Phi''(\underline{\$p_y}) = NOP$

(*Aucune opération n'est associée aux nouvelles places*). ⊗

La figure 7.4 donne la formalisation graphique de la règle RD04.

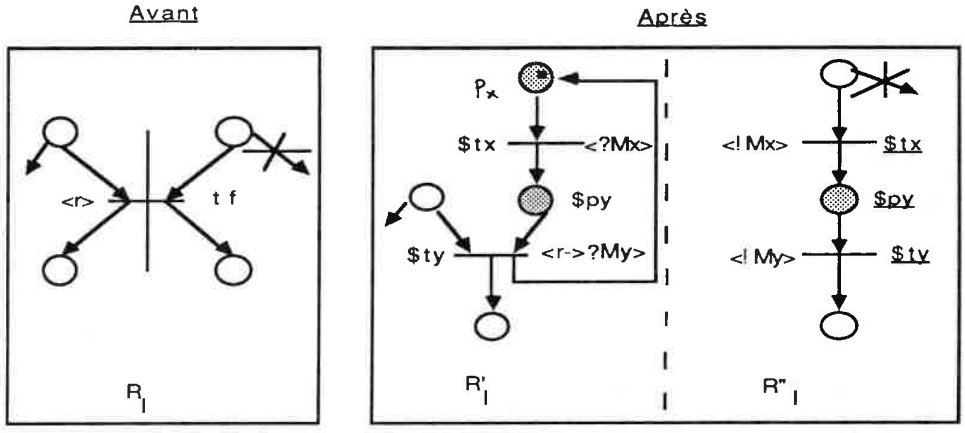


Figure 7.4

La synchronisation du tir des transitions $\$t_x$ et $\underline{\$t_x}$ permet au sous-réseau R_I' de s'assurer que la précondition définie par les places contenues dans E_f'' est devenue vraie. Ces places n'étant pas partagées, cette information sera encore vraie lorsque la transition $\$t_y$ deviendra validée et franchissable.

(Validation en annexe C, chapitre C.3)

d) La règle RD05

Le schéma de décomposition correspondant à la règle RD05 est caractérisé par deux points essentiels :

1°) Le prédicat associé à la transition frontière doit être un événement dont la politique de prise en compte des occurrences est indépendante du temps. C'est le cas par exemple de la politique définie par le type de communication égalité où la contrainte imposée est essentiellement une contrainte d'ordre ; dès lors qu'une occurrence est produite, elle peut être prise en compte plus ou moins vite, pourvu qu'au moment de sa prise en compte, elle corresponde à l'occurrence la plus ancienne.

2°) Par ailleurs, ce prédicat doit être affecté avec un sous-ensemble de places ne contenant pas de places partagées à l'un des deux sous-réseaux (disons R"), et les autres places à l'autre sous-réseau (disons R'). Les places affectées à R' peuvent être ou ne pas être partagées.

Définition D 7.5 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I'' = < R'' ; V' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R' = < P'', T'' ; Pre'' ; Post'' >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) Ψ (t) est un événement dont la politique de prise en compte des occurrences est indépendante du temps

(2) $\cdot t_f = E_f' + E_f''$, $E_f' \cap E_f'' = \varnothing$, $|E_f'| > 0$, $|Ef''| \geq 0$

(E_f' *et* E_f'' *définissent un partitionnement des places en entrée de la transition* t_f *en deux sous-ensembles disjoints*)

(3) $t_f \cdot = S_f' + S_f''$, $S_f' \cap S_f'' = \varnothing$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de la transition* t_f *en deux sous-ensembles disjoints*)

(4) $\forall p \in E_f'' : p \cdot = \{ t_f \}$

(*La seule transition en sortie des places contenues dans* E_f'' *est* t_f)

(5) $\forall p \in \cdot t_f , \forall t \in T : t \neq t_f$ et $Pre (p, t) > 0 \Rightarrow$ t n'est pas une transition frontière.

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

les transformations correspondant au choix de décomposition qui consiste à affecter les places contenues dans E_f' et S_f' au sous-réseau R_I', les places contenues dans E_f'' et S_f'' au sous-réseau R_I'' et le prédicat défini par Ψ (t_f) au sous-réseau R_I'' sont définies comme suit :

Dans R_I'

(6) $T' <- T' + \{ \$t_x \}$ (*On inclut dans* T' *une nouvelle transition* $\$t_x$)

(7) $P' <- P' + E_f' + S_f'$ (*On inclut dans* P' *les places contenues dans* E_f' *et* S_f')

(8) $\forall p \in P' : Pre'(p, \$t_x) = Pre(p, t_f)$ si $p \in E_f'$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en entrée les places contenues dans* E_f')

(9) $\forall p \in P' : Post'(p, \$t_x) = Post(p, t_f)$ si $p \in S_f'$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en sortie les places contenues dans* S_f').

Dans R_I''

(10) $T'' \leftarrow T'' + \{ \underline{\$t_x} \}$ (*Dans* T' *on inclut une nouvelle transition* $\underline{\$t_x}$)

(11) $P'' \leftarrow P'' + E_f'' + S_f''$

(*Dans* P'' *on inclut les places contenues dans* E_f'' *et* S_f'')

(12) $\forall p \in P'' : Pre''(p, \underline{\$t_x}) = Pre(p, t_f)$ si $p \in E_f''$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en entrée les places contenues dans* E_f'')

(13) $\forall p \in P'' : Post''(p, \underline{\$t_x}) = Post(p, t_f)$ si $p \in S_f''$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en sortie les places contenues dans* S_f'').

Dans R_I' *et* R_I''

(14) $\Psi'(\$t_x) = ? M_x'$, $\Psi''(\underline{\$t_x}) = \Psi(t_f) \rightarrow ! M_x$

(*On associe le prédicat défini par* $\Psi(t_f)$ *à la transition* $\underline{\$t_x}$, *et on définit une synchronisation par rendez-vous entre* $\$t_x$ *et* $\underline{\$t_x}$)

(15) $\forall p \in E_f' + S_f' : \Phi'(p) = \Phi(p)$

(*Les places contenues dans* E_f' *et* S_f' *conservent la même interprétation dans* R_I *et* R_I')

(16) $\forall p \in E_f'' + S_f'' : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* E_f'' *et* S_f'' *conservent la même interprétation dans* R_I *et* R_I''). ⊗

La figure 7.5 schématise graphiquement le schéma de décomposition et les

règles de transformation de la règle RD05.

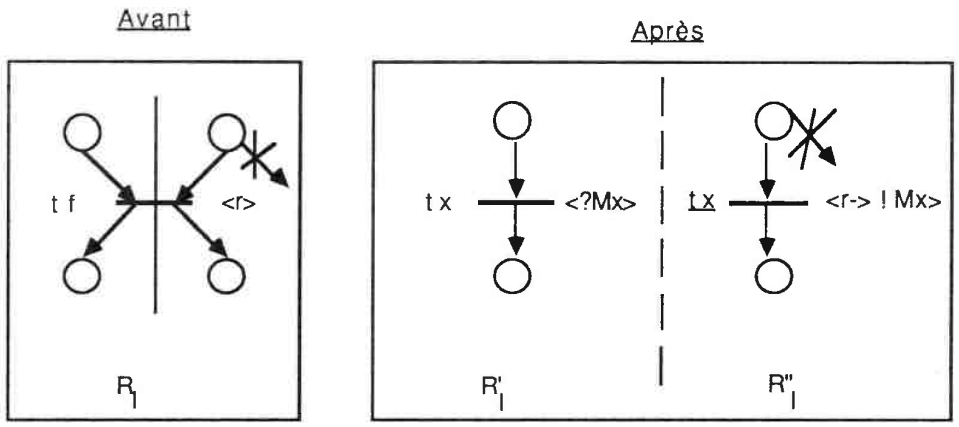


Figure 7.5

Dans le réseau initial, la précondition pour que les occurrences de r soient prises en compte est que toutes les places en entrée de t_f soient suffisamment marquées. On notera que la décomposition affaiblit cette précondition en la restreignant aux seules places contenues dans E_f". Les occurrences de r pourront de ce fait, après leur production, être prises en compte plus tôt. Leur politique de prise en compte étant indépendante du temps, on peut considérer que cela ne modifiera pas la logique du comportement externe.

Par ailleurs, on notera que, pour un choix de décomposition donné, lorsque les places contenues dans E_f' ne sont pas partagées, ce choix satisfait également le schéma de décomposition de la règle RD03. De ce fait, ces deux schémas de décomposition ont une intersection non vide. Lorsqu'une transition frontière satisfait les deux schémas, on appliquera de préférence la règle RD05 qui conduit à une solution plus simple.

(Validation en annexe C, chapitre C.4)

e) La règle RD06

Le schéma de décomposition de la règle RD06 se distingue du schéma de décomposition de la règle RD05 d'une part par le fait que le prédicat associé à la transition frontière n'est pas un événement dont la prise en compte des occurrences est indépendante du temps, et d'autre part par le fait qu'une au moins des places affectées au réseau R_I' est partagée.

Définition D 7.6 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I'' = < R'' ; V' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R' = < P'', T'' ; Pre'' ; Post'' >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) Ψ (t) n'est pas un événement dont la politique de prise en compte des occurrences est indépendante du temps

(2) $\cdot t_f = E_f' + E_f''$, $E_f' \cap E_f'' = \varnothing$, $|E_f'| > 0$, $|E_f''| \geq 0$

(E_f' *et* E_f'' *définissent un partitionnement des places en entrée de la transition* t_f *en deux sous-ensembles disjoints*).

(3) $t_f' = S_f' + S_f''$, $S_f' \cap S_f'' = \varnothing$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de la transition* t_f *en deux sous-ensembles disjoints*)

(4) $\exists p \in E_f'$, $\exists t \in T$ telle que : $Pre (p, t) > 0$ et $t \neq t_f$

(*Dans* E_f' *il existe au moins une place partagée*)

(5) $\forall p \in E_f'' : p^{\cdot} = \{ t_f \}$

(*La seule transition en sortie des places contenues dans* E_f'' *est* t_f)

(6) $\forall p \in {}^{\cdot}t_f, \forall t \in T : t \neq t_f$ et $Pre(p, t) > 0 \Rightarrow$ t n'est pas une transition frontière

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

le choix de décomposition qui consiste à affecter les places contenues dans E_f' et S_f' au sous-réseau R', les places contenues dans E_f'' et S_f'' au sous-réseau R" et le prédicat défini par Ψ (t) au sous-réseau R" est impossible à réaliser. ⊗

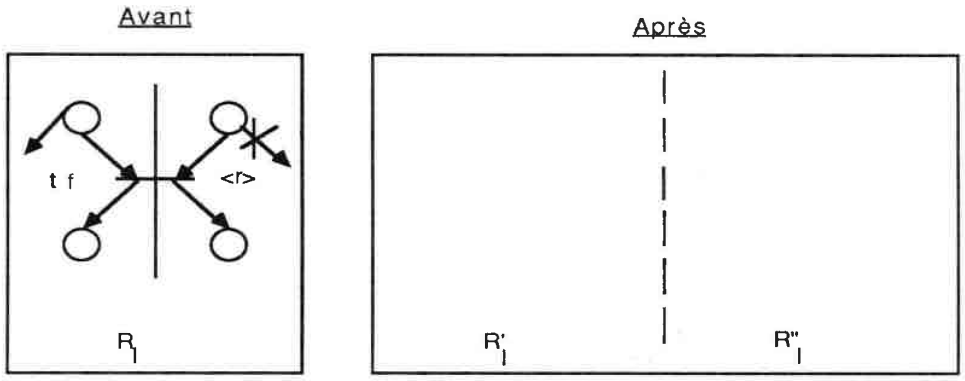


Figure 7.6

En effet, Ψ (t) ne représentant plus comme dans la règle RD05 un événement dont la politique de prise en compte des occurrences est indépendante du temps, on ne peut plus affaiblir la précondition pour sa vérification, sans remettre en cause le comportement initialement défini. Le sous-réseau R_I'' auquel ce prédicat est affecté doit donc, avant toute vérification, s'assurer que la précondition intialement définie est satisfaite. Pour effectuer cette première vérification, le réseau R_I'' aura besoin de connaître à un instant

donné le marquage des places contenues dans E_f'. En pratique, cette information sera impossible à obtenir. Entre une notification du marquage des places contenues dans E_f' par le sous-réseau R_I', et la réception de cette notification par le sous-réseau R_I'', le marquage notifié peut évoluer, puisque les places contenues dans E_f' sont aussi en entrée d'autres transitions pouvant être franchies de façon indépendante.

f) La règle RD07

Cette règle caractérise également un schéma de décomposition impossible à réaliser. Ce schéma correspond au cas où les sous-ensembles de places affectés aux deux sous-réseaux contiennent chacun au moins une place partagée, le prédicat associé à la transition frontière étant affecté à l'un des deux sous-réseaux.

Définition D 7.7:

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I'' = < R'' ; V' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R' = < P'', T'' ; Pre'' ; Post'' >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) $\Psi (t_f) \neq pr_0$

(Le prédicat associé à t_f est différent du prédicat toujours vrai)

(2) $\cdot t_f = E_f' + E_f''$, $E_f' \cap E_f'' = \varnothing$, $|E_f'| > 0$, $|E_f''| > 0$

(E_f' et E_f'' définissent un partitionnement des places en entrée de la

transition t_f *en deux sous-ensembles disjoints*)

(3) $t_f^{\cdot} = S_f' + S_f''$, $S_f' \cap S_f'' = \varnothing$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de la transition* t_f *en deux sous-ensembles disjoints*)

(4) $\exists\, p \in E_f'$, $\exists\, t \in T$ telles que : Pre (p, t) > 0 et $t \neq t_f$

(*Dans* E_f' *il existe au moins une place partagée*)

(5) $\exists\, p \in E_f''$, $\exists\, t \in T$ telles que : Pre (p, t) > 0 et $t \neq t_f$

(*Dans* E_f'' *il existe au moins une place partagée*).

(6) $\forall\, p \in {}^{\cdot}t_f$, $\forall\, t \in T : t \neq t_f$ et Pre (p, t) > 0 ⇒ t n'est pas une transition frontière.

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

le choix de décomposition qui consiste à affecter les places contenues dans E_f' et S_f' au sous-réseau R_I', les places contenues dans E_f'' et S_f'' au sous-réseau R_I'' et le prédicat défini par Ψ (t_f) à l'un des deux sous-réseaux (disons R_I'') est impossible à réaliser.

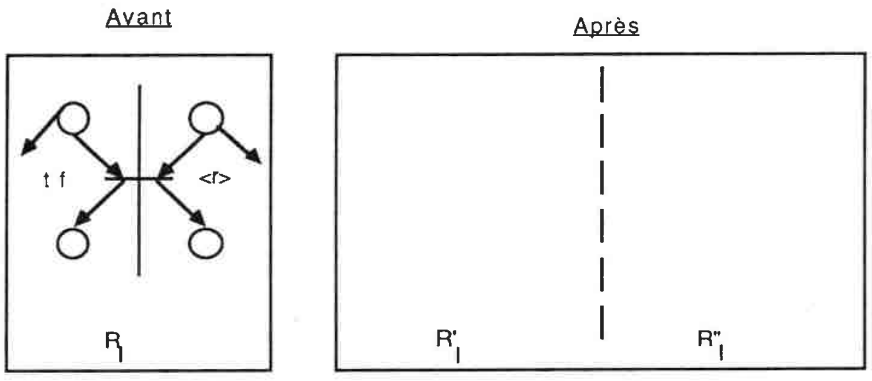


Figure 7.7

Les raisons de cette impossibilité sont les mêmes que celles évoquées pour la règle RD06 : avant de vérifier le prédicat Ψ (t_f) qui lui est associé, le réseau R_I" ne peut connaître avec exactitude le marquage des places affectées au sous-réseau R_I'.

g) La règle RD08

Dans plusieurs des règles précédentes (RD03, RD04, RD05), nous avons supposé que les places en entrée de la transition frontière étaient partitionnées en sous-ensembles, qui ne pouvaient pas toujours être vides. Lorsque ces sous-ensembles sont vides, les protocoles de synchronisation définis restent encore valables, mais peuvent aussi être définis plus simplement en appliquant la règle RD08.

Définition D 7.8 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I" :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre' , Post' >$

$R_I" = < R" ; V' ; OP" ; PR" ; \Phi" ; \Psi" >$ avec $R' = < P", T" ; Pre" ; Post" >$.

Pour toute transition frontière $t_f \in T$ telle que :

(1) $\Psi (t_f) \neq pr_0$

(*Le prédicat associé à t_f est différent du prédicat toujours vrai*)

(2) $E_f = {}^{\cdot}t_f$, $|E_f| \geq 0$

(E_f *désigne l'ensemble des places en entrée de la transition* t_f)

(3) $t_f^{\cdot} = S_f' + S_f"$, $S_f' \cap S_f" = \varnothing$, $|S_f'| > 0$, $|S_f"| > 0$

(S_f' *et* S_f'' *définissent un partitionnement des places en sortie de la transition* t_f *en deux sous-ensembles disjoints*)

(4) $\forall p \in {}^{\cdot}t_f$, $\forall t \in T : t \neq tf$ et $Pre(p, t) > 0 \Rightarrow$ t n'est pas une transition frontière.

(*Aucune des places d'entrée de la transition* t_f *n'est en entrée d'une autre transition frontière*)

les transformations correspondant au choix de décomposition qui consiste à associer les places contenues dans E_f et S_f' en sous-réseau R_I', les places contenues dans S_f'' au sous-réseau R_I'', et le prédicat défini par Ψ (t_f) au réseau R_I'' sont définies comme suit :

Dans R_I'

(5) $T' \leftarrow T' + \{ \$t_x \}$ (*On inclut dans T' une nouvelle transition* $\$t_x$)

(6) $P' \leftarrow P' + E_f + S_f'$ (*On inclut dans P' les places contenues dans* E_f *et* S_f')

(7) $\forall p \in P' : Pre'(p, \$t_x) = Pre(p, t_f)$ si $p \in E_f$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en entrée les places contenues dans* E_f)

(8) $\forall p \in P' : Post'(p, \$t_x) = Post(p, t_f)$ si $p \in S_f'$ 0 sinon

(*La nouvelle transition* $\$t_x$ *reçoit en sortie les places contenues dans* S_f').

Dans R_I''

(9) $T'' \leftarrow T'' + \{ \underline{\$t_x} \}$ (*On inclut dans T" une nouvelle transition* $\underline{\$t_x}$)

(10) $P'' \leftarrow P'' + S_f''$ (*On inclut dans P" les places contenues dans* S_f'')

(11) $\forall p \in P'' : Pre''(p , \underline{\$t_x}) = 0$

(*La nouvelle transition* $\underline{\$t_x}$ *ne reçoit aucune place en entrée*)

(12) $\forall p \in P'' : Post''(p, \underline{\$t_x}) = Post(p, t_f)$ si $p \in S_f''$ 0 sinon

(*La nouvelle transition* $\underline{\$t_x}$ *reçoit en sortie les places contenues dans* S_f").

Dans R_I' *et* R_I"

(13) $\Psi'(\$t_x) = \Psi(t_f) \rightarrow ? M_x$, $\Psi''(\underline{\$ tx}) = ! M_x$

(*On associe le prédicat défini par* $\Psi(t_f)$ *à la transition* $\$t_x$ *et on définit une synchronisation par rendez-vous entre* $\$t_x$ *et* $\underline{\$t_x}$)

(14) $\forall p \in E_f + S_f' : \Phi'(p) = \Phi(p)$

(*Les places contenues dans* E_f *et* S'_f *conservent la même interprétation dans* R_I *et* R_I')

(15) $\forall p \in S_f'' : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* S_f" *conservent la même interprétation dans* R_I *et* R_I"). ⊗

La figure 7.8 formalise graphiquement la règle RD08.

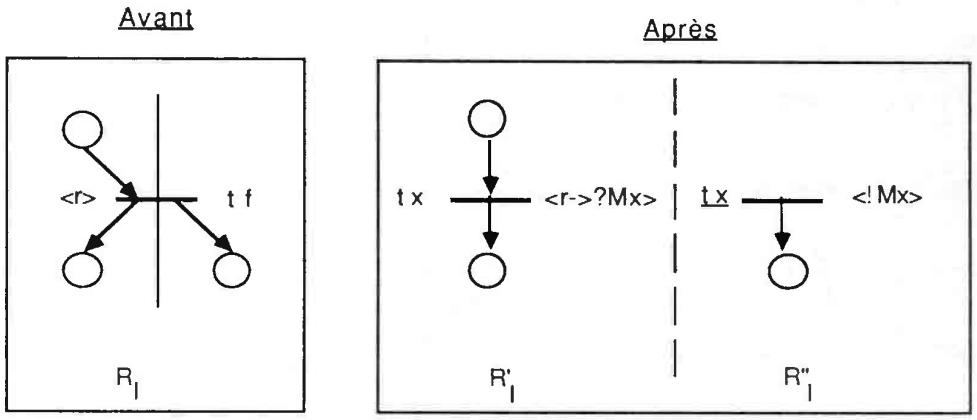


Figure 7.8

(Validation en annexe B, chapitre C.5)

7.1.3 TRANSITIONS FRONTIERES EN CONFLIT STRUCTUREL

Nous abordons à présent l'étude des schémas de décomposition où plusieurs transitions frontières sont en conflit structurel. Ces schémas peuvent correspondre par exemple à des situations où le concepteur cherche à encapsuler séparément les tâches utilisatrices d'une ressource et la tâche d'allocation de cette ressource. Pour ce type de schémas, les liens très forts qui existent entre les opérations modélisées conduisent à des protocoles de synchronisation plus complexes. Pour cela, nous restreignons notre étude aux cas où toutes les places communes des transitions en conflit sont regroupées dans un sous-réseau, et les autres places en entrée de ces transitions dans un autre sous-réseau.

a) La règle RD09

Dans le schéma de décomposition correspondant à cette règle, les places communes et les prédicats associés aux transitions frontières en conflit structurel ne sont pas affectés au même sous-réseau.

Définition D 7.9 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Pour tout sous-ensemble $T_f = \{ t_{f1}, ..., t_{fn} \} \subset T$ de transitions frontières tel que :

(1) $E_c = \{ p_j \mid p_j^{\cdot} = T_f \} \neq \varnothing$

(E_c *désigne l'ensemble des places communes aux transitions contenues dans* T_f)

(2) $\forall\, t_{fi} \in T_f : {}^{\cdot}t_{fi} = E'_{fi} + E_c \;,\; E'_{fi} \cap E_c = \varnothing \;,\; |E'_{fi}| \geq 0$

(E'_{fi} *et* E_c *définissent un partitionnement des places en entrée de* t_{fi} *en deux sous-ensembles disjoints*)

(3) $\forall\, t_{fi} \in T_f, \forall\, p \in E'_{fi} : p^{\cdot} \cap (T_f - \{ t_{fi} \}) = \varnothing$

(*Pour toute transition* $t_{fi} \in T_f$, *les places contenues dans* E'_{fi} *ne sont en entrée d'aucune autre transition de* T_f)

(4) $\forall\, t_{fi} \in T_f : t_{fi}^{\cdot} = S'_{fi} + S''_{fi} \;,\; S'_{fi} \cap S''_{fi} = \varnothing$

(S'_{fi} *et* S''_{fi} *définissent un partitionnement des places en sortie de* t_{fi} *en deux sous-ensembles disjoints*).

(5) $\forall\, t_{fi}, t_{fj} \in T_f, \forall\, p \in E_c : \mathrm{Pre}(p, t_{fi}) = \mathrm{Pre}(p, t_{fj})$

(*Pour chaque place* $p \in E_c$, *le nombre de marques consommables par chaque transition* $t_{fi} \in T_f$ *est le même*)

les transformations correspondant au choix de décomposition qui consiste pour chaque transition $t_{fi} \in T_f$ à associer les places contenues dans E'_{fi} et S'_{fi} au sous-réseau R_I', les places contenues dans E_c et S_{fi}'' au sous-réseau R_I'', et le prédicat éventuellement défini par $\Psi(t_{fi})$ au sous-réseau R_I' sont définies comme suit (f1 et f2 sont deux applications qui associent respectivement une nouvelle transition et une nouvelle place à une transition de T_f):

Dans R_I'

(6) $T' \leftarrow T' + T_x + \{ \$t_y \}$

où $T_x = \{ \$t_{xi} \mid f1(\$t_{xi}) \in T_f \}$

(*Dans* T' *on inclut une nouvelle transition* $\$t_y$, *et pour chaque transition frontière* $t_{fi} \in T_f$ *une nouvelle transition* $\$t_{xi}$)

(7) $P' \leftarrow P' + (\Sigma_{tfi \in Tf} E'_{fi} + S'_{fi}) + \{ \$p_{y1} , \$p_{y2} \}$

(*Dans* P' *on inclut deux nouvelles places* $\$p_{y1}$ *et* $\$p_{y2}$, et *pour chaque transition frontière* $t_{fi} \in T_f$ *les places contenues dans* E'_{fi} *et* S'_{fi})

(8) $M'(\$p_{y1}) = 1$ (*On marque la place* $\$p_{y1}$ *à* 1)

(9) $\forall\ t_{xi} \in T_x$ telle que $f1(\$t_{xi}) = t_{fi}$, $\forall\ p \in P'$:

$Pre'(p , \$t_{xi}) = Pre(p, t_{fi})$ si $p \in E'_{fi}$

1 si $p = \$p_{y2}$

0 sinon

(*Chaque transition* $\$t_{xi} \in T_x$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en entrée les places contenues dans* E'_{fi} *et la nouvelle place* $\$p_{y2}$)

(10) $\forall\ \$t_{xi} \in T_x$ telle que $f1(\$t_{xi}) = t_{fi}$, $\forall\ p \in P'$:

$Post'(p , \$t_{xi}) = Post(p, t_{fi})$ si $p \in S'_{fi}$

1 si $p = \$p_{y1}$

0 sinon

(*Chaque transition* $\$t_{xi} \in T_x$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en sortie les places contenues dans* S'_{fi} *et la nouvelle place* $\$p_{y1}$)

(11) $\forall\ p \in P' : Pre'(p , \$t_y) = 1$ si $p = \$p_{y1}$ 0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en entrée la nouvelle place* $\$p_{y1}$).

(12) $\forall\ p \in P' : Post'(p , \$t_y) = 1$ si $p = \$p_{y2}$ 0 sinon

(*La nouvelle transition* $\$t_y$ *reçoit en sortie la nouvelle place* $\$p_{y2}$).

Dans R_I''

(13) $T'' \leftarrow T'' + T_x + \{ \underline{\$t_y} \}$

où $T_{\underline{x}} = \{ \underline{\$t_{xi}} \mid f1(\underline{\$t_{xi}}) \in T_f \}$

(Dans T" on inclut une nouvelle transition $\underline{\$t_y}$*, et pour chaque transition frontière* $t_{fi} \in T_f$ *une nouvelle transition* $\underline{\$t_{xi}}$*).*

(14) $P" \leftarrow P" + E_c + (\Sigma_{tfi \in Tf} S"_{fi}) + \{ \underline{\$p_{y2}} \}$

(Dans P" on inclut une nouvelle place $\underline{\$p_{y2}}$*, les places communes contenues dans* E_c*, et pour chaque transition frontière* $t_{fi} \in T_f$*, les places contenues dans* $S"_{fi}$*).*

(15) $\forall \underline{\$t_{xi}} \in T_{\underline{x}}$ telle que $f1(\underline{\$t_{xi}}) = t_{fi}$, $\forall\ p \in P"$:

$Pre"(p, \underline{\$t_{xi}}) =$ 1 si $p = \underline{\$p_{y2}}$ 0 sinon

(Chaque transition $\underline{\$t_{xi}} \in T_{\underline{x}}$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en entrée la place* $\underline{\$p_{y2}}$*)*

(16) $\forall\ \underline{\$t_{xi}} \in T_{\underline{x}}$ telle $f1(\underline{\$t_{xi}}) = t_{fi}$, $\forall\ p \in P"$:

$Post"(p, \underline{\$t_{xi}}) = Post(p, t_{fi})$ si $p \in S_{fi}"$ 0 sinon

(Chaque transition $\underline{\$t_{xi}} \in T_{\underline{x}}$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en sortie les places contenues dans* $S"_{fi}$*)*

(17) $\forall\ t_{fi} \in T_f$, $\forall\ p \in P"$:

$Pre"(p, \underline{\$t_y}) = Pre(p, t_{fi})$ si $p \in E_c$ 0 sinon

(La nouvelle transition $\underline{\$t_y}$ *reçoit en entrée les places communes contenues dans* E_c*)*

(18) $\forall\ p \in P"$: $Post"(p, \underline{\$t_y}) = 1$ si $p = \$p_{y2}$ et 0 sinon

(La nouvelle transition $\underline{\$t_y}$ *reçoit en sortie la nouvelle place* $\underline{\$p_{y2}}$ *).*

Dans R_I' *et* $R_I"$

(19) $\forall\ \$t_{xi} \in T_x$, $\forall\ \underline{\$T_{xi}} \in T_{\underline{x}}$ telle que $f1(\$t_{xi}) = f1(\underline{\$t_{xi}}) = t_{fi}$:

$\Psi'(\$t_{xi}) = \Psi\ (t_{fi}) \rightarrow !\ M_{xi}$, $\Psi"(\underline{\$t_{xi}}) = ?\ M_{xi}$

(*Pour chaque couple de transitions* $\$t_{xi} \in T_x$ *et* $\underline{\$t_{xi}} \in T_{\underline{x}}$ *associées à la même transition frontière* $t_{fi} \in T_f$, *on définit une synchronisation par rendez-vous et on associe le prédicat défini par* $\Psi(t_{fi})$ *à* $\$t_{xi}$)

(20) $\Psi'(\$t_y) = !\, M_y$, $\Psi''(\underline{\$t_y}) = ?\, M_y$

(*Une synchronisation par rendez-vous est définie entre les nouvelles transitions* $\$t_y$ *et* $\underline{\$t_y}$).

(21) $\forall\, p \in E_c + (\Sigma_{tfi \in Tf}\, S''_{fi}) : \Phi''(p) = \Phi(p)$

(*Les places contenues dans* E_c *et dans les* S''_{fi} *conservent la même interprétation dans* R_I *et* R_I'').

(22) $\forall\, p \in \Sigma_{tfi \in Tf}\, (E'_{fi} + S'_{fi}) : \Phi'(p) = \Phi(p)$

(*Les places contenues dans les* E'_{fi} *et* S'_{fi} *conservent la même interprétation dans* R_I *et* R_I').

(23) $\Phi''(\underline{\$p_{y2}}) = NOP$

(24) $\Phi'(\$p_{y1}) = \Phi'(\$p_{y2}) = NOP$

La figure 7.9 schématise graphiquement le schéma de décomposition et les règles de transformation de la règle RD07.

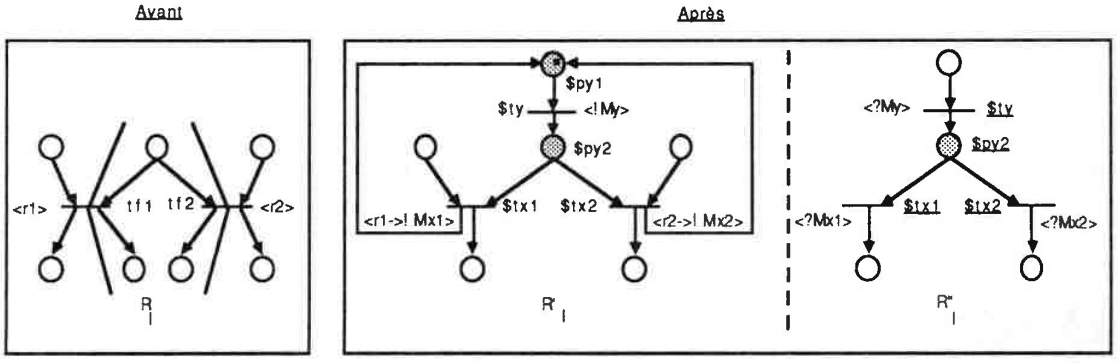


Figure 7.9

La réalisation du rendez-vous défini entre t_y et $\underline{t_y}$ signifie pour le sous-réseau R_I' que la place commune affectée au sous-réseau R_I'' est marquée. Comme cette place n'est partagée que par les transistions en conflit, cette information restera vraie jusqu'à ce que R_I' résolve le conflit.

(Validation en annexe C, chapitre C.6)

b) La règle RD10

Le schéma de décomposition de la règle RD10 se distingue de celui de la règle RD09 par le fait que le sous-réseau d'affectation des places communes et des prédicats associés aux transitions frontières en conflit structurel est le même. En plus, dans ce schéma de décomposition, les autres places en entrée des transitions frontières ne doivent pas être partagées.

Définition D 7.10 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Pour tout sous-ensemble $T_f = \{ t_{f1}, ..., t_{fn} \} \subset T$ de transitions frontières tel que :

(1) $E_c = \{ p_j \mid p_j^{\cdot} = T_f \} \neq \varnothing$

(E_c *désigne l'ensemble des places communes aux transitions contenues dans* T_f)

(2) $\forall\ t_{fi} \in T_f : {}^{\cdot}t_{fi} = E'_{fi} + E_c\ ,\ E'_{fi} \cap E_c = \varnothing\ ,\ |E'_{fi}| \geq 0$

(E'_{fi} *et* E_c *définissent un partitionnement des places en entrée de* t_{fi} *en deux sous-ensembles disjoints*)

(3) $\forall\ t_{fi} \in T_f, \forall\ p \in E'_{fi} : p^{\cdot} = \{\ t_{fi}\ \}$

(*Pour toute transition* $t_{fi} \in T_f$, *les places contenues dans* E'_{fi} *ne sont pas partagées*)

(4) $\forall\ t_{fi} \in T_f : t_{fi}^{\cdot} = S'_{fi} + S''_{fi}\ ,\ S'_{fi} \cap S''_{fi} = \varnothing$

(S'_{fi} *et* S''_{fi} *définissent un partitionnement des places en sortie de* t_{fi} *en deux sous-ensembles disjoints*).

(5) $\forall\ t_{fi}, t_{fj} \in T_f, \forall\ p \in E_c$: Pre (p, t_{fi}) = Pre(p, t_{fj})

(*Pour chaque place* $p \in E_c$, *le nombre de marques consommables par chaque transition* $t_{fi} \in T_f$ *est le même*)

les transformations correspondant au choix de décomposition qui consiste pour chaque $t_{fi} \in T_f$ à affecter les places contenues dans E'_{fi} et S'_{fi} au sous-réseau R_I', les places contenues dans E_c et S''_{fi} au sous-réseau R_I'', et le prédicat éventuellement défini par $\Psi\ (t_{fi})$ au sous-réseau R_I'' sont définies comme suit (ici également f1 et f2 sont deux applications qui associent respectivement une nouvelle transition et une nouvelle place à une transition de T_f) :

Dans R_I'

(6) $T' \leftarrow T' + T_x + T_y + \{\ \$t_z\ \}$

où $T_x = \{\ \$t_{xi}\ |\ f1(\$t_{xi}) \in T_f\ \}$

$T_y = \{\ \$t_{yi}\ |\ f1(\$t_{yi}) = t_{fi} \in T_f\ ,\ E'_{fi} \neq \varnothing\ \}$

(*Dans* T' *on inclut une nouvelle transition* $\$t_z$, *et pour chaque transition frontière* $t_{fi} \in T_f$ *une nouvelle transition* $\$t_{xi}$, *et si* E'_{fi} *n'est pas vide, une deuxième nouvelle transition* $\$t_{yi}$)

(7) $P' \leftarrow P' + \Sigma_{tfi \in Tf} (E'_{fi} + S'_{fi}) + P_y + \{ \$p_{z1} , \$p_{z2} \}$

où $P_y = \{ \$p_{yi} \mid f2(\$P_{yi}) = t_{fi} \in T_f , E'_{fi} \neq \varnothing \}$

(Dans P' on inclut deux nouvelles places $\$p_{z1}$ et $\$p_{z2}$, et pour chaque transition $t_{fi} \in T_f$ les places contenues dans E'_{fi} et S'_{fi} puis une nouvelle place $\$p_{yi}$ si E'_{fi} n'est pas vide).

(8) $M'(\$p_{z1}) = 1$

(On marque initialement à 1 la nouvelle place $\$p_{z1}$)

(9) $\forall \$t_{xi} \in T_x$ telle que $f1(\$t_{xi}) = t_{fi}$, $\forall p \in P'$:

$Pre'(p, \$t_{xi}) =$ 1 si $p = \$p_{z2}$

1 si $p \in P_y$ et $f2(p) = t_{fi}$

0 sinon

(Chaque transition $\$t_{xi} \in T_x$ associée à une transition $t_{fi} \in T_f$ reçoit en entrée la nouvelle place $\$p_{z2}$ et la place $\$p_{yi} \in P_y$ associée à cette même transition)

(10) $\forall \$t_{xi} \in T_x$ telle que $f1(\$t_{xi}) = t_{fi}$, $\forall p \in P'$:

$Post'(p , \$t_{xi}) =$ $Post(p, t_{fi})$ si $p \in S'_{fi}$

1 si $p = \$p_{z1}$

0 sinon

(Chaque transition $\$t_{xi} \in T_x$ associée à une transition frontière $t_{fi} \in T_f$ reçoit en sortie les places contenues dans S'_{fi} et la nouvelle place $\$p_{z1}$)

(11) $\forall \$t_{yi} \in T_y$ telle que $f1(\$t_{yi}) = t_{fi}$, $\forall p \in P'$:

$Pre'(p, \$t_{yi}) = Pre(p, t_{fi})$ si $p \in E'_{fi}$ 0 sinon

(Chaque transition $\$t_{yi} \in T_y$ associée à une transition $t_{fi} \in T_f$ reçoit en entrée les places contenues dans E'_{fi})

(12) $\forall \$t_{yi} \in T_y$ telle que $f1(\$t_{yi}) = t_{fi}$, $\forall p \in P'$:

Post' (p, $t_{yi}) = 1 si (p ∈ P_y et f2(p) = t_{fi}) 0 sinon

(*Chaque transition* $t_{yi} ∈ T_y$ *associée à une transition* $t_{fi} ∈ T_f$ *reçoit en sortie la place* $p_{yi} ∈ P_y$ *associée à cette même transition*)

(13) ∀ p ∈ P' : Pre' (p, $t_z) = 1 si p = $p_{z1} et 0 sinon

(*La transition* t_z *reçoit en entrée la nouvelle place* p_{z1})

(14) ∀ $t_{zj} ∈ T_z telle que f3($t_{zj}) = p_j , ∀ p ∈ P' :

Post' (p , $t_z) = 1 si p = $p_{z2} et 0 sinon

(*La transition* t_z *reçoit en sortie la place* p_{z2}).

Dans R_I"

(15) T" <- T" + T_x + T_y + { t_z }

où T_x = { t_{xi} | f1($t_{xi}) ∈ T_f }

T_y = { t_{yi} | f1 ($t_{yi}) = $t_{fi} ∈ T_f , $E'_{fi} ≠ ∅ }

(*Dans* T" *on inclut une nouvelle transition* t_z, *et pour chaque transition frontière* $t_{fi} ∈ T_f$ *une nouvelle transition* t_{xi}, *et si* E'_{fi} *n'est pas vide, une deuxième nouvelle transition* t_{yi})

(16) P" <- P" + E_c + ($\Sigma_{tfi∈Tf}$ $S"_{fi}$) + P_y + P_w + { p_{z2} }

où P_y = { p_{yi} | f2($p_{yi}) = $t_{fi} ∈ T_f , $E'_{fi} ≠ ∅ }

P_w = { p_{wi} | f2($p_{wi}) = $t_{fi} ∈ T_f , $E'_{fi} ≠ ∅ }

(*Dans* P" *on inclut une nouvelle place* p_{z2}, *les places communes contenues dans* E_c, *et pour chaque transition frontière* $t_{fi} ∈ T_f$ *les places contenues dans* $S"_{fi}$ *puis deux nouvelles places* p_{wi} *et* p_{yi} *si* E'_{fi} *n'est pas vide*)

(17) ∀ p ∈ P_w : M"(p) = 1

(*Les places contenues dans* P_w *sont initialement marquées à* 1)

(18) $\forall \underline{\$t_{xi}} \in T_x$ telle que $f1(\underline{\$t_{xi}}) = t_{fi}$, $\forall p \in P''$:

$Pre''(p, \underline{\$t_{xi}}) =$ 1 si $p = \underline{\$p_{z2}}$

1 si $p \in P_y$ et $f2(p) = t_{fi}$

0 sinon

(*Chaque transition* $\underline{\$t_{xi}} \in T_x$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en entrée la nouvelle place* $\underline{\$p_{z2}}$ *et la place* $\underline{\$p_{yi}} \in P_y$ *associée à cette même transition*)

(19) $\forall \underline{\$t_{xi}} \in T_x$ telle $f1(\underline{\$t_{xi}}) = t_{fi}$, $\forall p \in P''$:

$Post''(p, \underline{\$t_{xi}}) =$ $Post(p, t_{fi})$ si $p \in S''_{fi}$

1 si $p \in P_w$ et $f2(p) = t_{fi}$

0 sinon

(*Chaque transition* $\underline{\$t_{xi}} \in T_x$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en sortie les places contenues dans* S''_{fi} *et la place* $\$p_{wi} \in P_w$ *associée à cette même transition*)

(20) $\forall \underline{\$t_{yi}} \in T_y$ telle que $f1(\underline{\$t_{yi}}) = t_{fi}$, $\forall p \in P''$:

$Pre''(p, \underline{\$t_{yi}}) =$ 1 si ($p \in P_w$ et $f2(p) = t_{fi}$) 0 sinon

(*Chaque transition* $\underline{\$t_{yi}} \in T_y$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en entrée la place* $\$p_{wi} \in P_w$ *associée à cette même transition*)

(21) $\forall \underline{\$t_{yi}} \in T_y$ telle que $f1(\underline{\$t_{yi}}) = t_{fi}$, $\forall p \in P''$:

$Post''(p, \underline{\$t_{yi}}) =$ 1 si ($p \in P_y$ et $f2(p) = t_{fi}$) 0 sinon

(*Chaque transition* $\underline{\$t_{yi}} \in T_y$ *associée à une transition frontière* $t_{fi} \in T_f$ *reçoit en sortie la place* $\underline{\$p_{yi}} \in P_y$ *associée à cette même transition*)

(22) $\forall t_{fi} \in T_f$, $\forall p \in P''$:

$Pre''(p, \underline{\$t_z}) = Pre(p, t_{fi})$ si $p \in E_c$ 0 sinon

(*La transition* $\underline{\$t_z}$ *reçoit en entrée les places communes contenues dans* E_c)

(23) $\forall p \in P''$: Post'' $(p, \underline{\$t_z}) = 1$ si $p = \$p_{z2}$ et 0 sinon

(La transition $\underline{\$t_z}$ reçoit en sortie la nouvelle place $\underline{\$p_{z2}}$).

Dans RI' et RI''

(24) $\forall \$t_{xi} \in T_x$, $\forall \underline{\$T_{xi}} \in T_x$ telle que $f1(\$t_{xi}) = f1(\underline{\$t_{xi}}) = t_{fi}$:

$\Psi''(\underline{\$t_{xi}}) = \Psi(t_{fi}) \rightarrow ! M_{xi}$, $\Psi'(\$t_{xi}) = ? M_{xi}$

(Pour chaque couple de transitions $\$t_{xi} \in T_x$ et $\underline{\$t_{xi}} \in T_x$ associées à la même transition frontière $t_{fi} \in T_f$, on définit une synchronisation par rendez-vous et on associe le prédicat défini par $\Psi(t_{fi})$ à $\underline{\$t_{xi}}$)

(25) $\forall \$t_{yi} \in T_y$, $\forall \underline{\$t_{yi}} \in T_y$ telles que $f1(\$t_{yi}) = f1(\underline{\$t_{yi}}) = t_{fi}$:

$\Psi'(\$t_{yi}) = ? M_{yi}$, $\Psi''(\underline{\$t_{yi}}) = ! M_{yi}$

(Pour chaque couple de transitions $\$t_{yi} \in T_y$ et $\underline{\$t_{yi}} \in T_y$ associées à la même transition frontière $t_{fi} \in T_f$, on définit une synchronisation par rendez-vous)

(26) $\Psi'(\$t_z) = ? M_z$, $\Psi''(\underline{\$t_z}) = ! M_z$

(On définit une synchronisation par rendez-vous entre les nouvelles transitions $\$t_z$ et $\underline{\$t_z}$).

(27) $\forall p \in E_c + (\Sigma_{tfi \in Tf} S''_{fi}) : \Phi''(p) = \Phi(p)$

(Les places contenues dans E_c et dans les S''_{fi} conservent la même interprétation dans R_I et R_I'').

(28) $\forall p \in \Sigma_{tfi \in Tf} (E'_{fi} + S'_{fi}) : \Phi'(p) = \Phi(p)$

(Les places contenues dans les E'_{fi} et S'_{fi} conservent la même interprétation dans R_I et R_I').

(29) $\forall p \in P_{\underline{y}} + P_w + \{ \$p_{\underline{z2}} \} : \Phi''(p) = NOP$

(30) $\forall p \in P_y + \{ \$p_{z1} + \$p_{z2} \} : \Phi'(p) = NOP.$ ⊗

Le schéma de décomposition et les règles de transformation de la règle RD10 sont schématisés graphiquement par la figure 7.10.

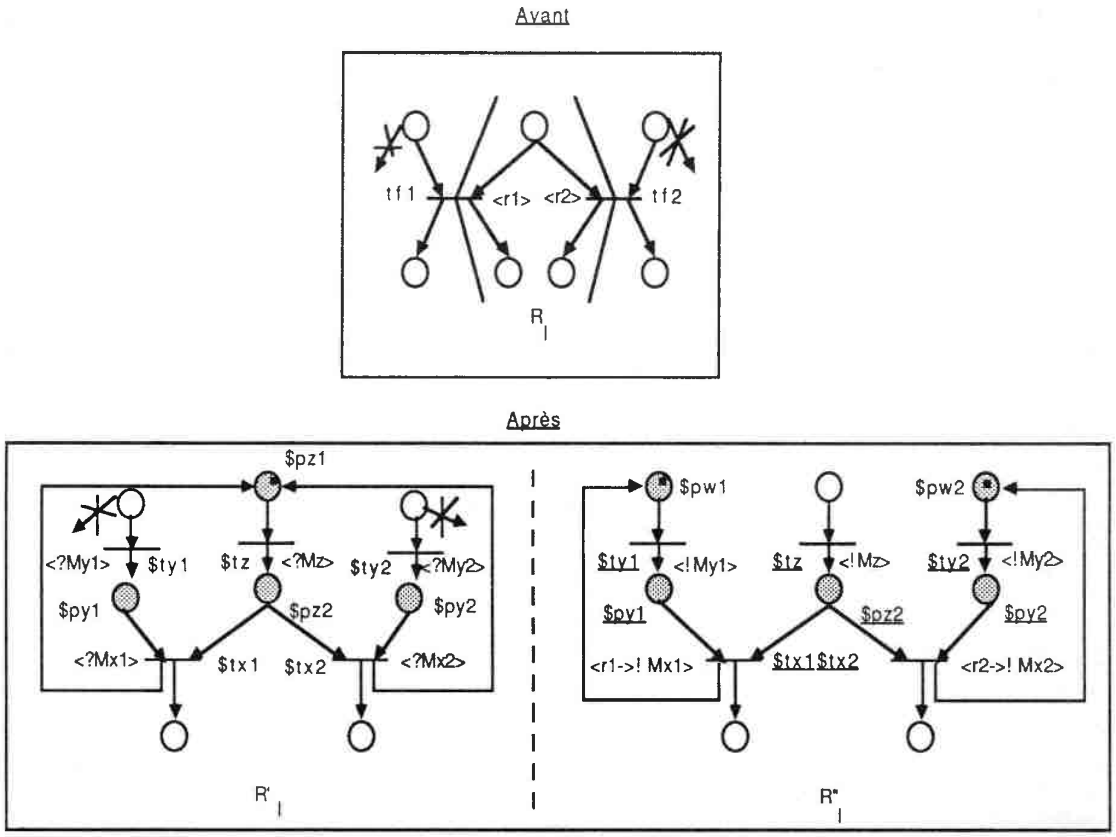


Figure 7.10

(Validation en annexe C, chapitre C.7)

c) La règle RD11

Le schéma de décomposition de la règle RD11 se distingue du schéma de décomposition de la règle RD10 par le fait que les places non communes sont partagées.

Définition 7.11 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Pour tout sous-ensemble $T_f = \{ t_{f1}, ..., t_{fn} \} \subset T$ de transitions frontières tel que :

(1) $E_c = \{ p_j \mid p_j^{\cdot} = T_f \} \neq \varnothing$

(E_c *désigne l'ensemble des places communes aux transitions contenues dans* T_f)

(2) $\forall t_{fi} \in T_f : {}^{\cdot}t_{fi} = E'_{fi} + E_c$, $E'_{fi} \cap E_c = \varnothing$, $|E'_{fi}| \geq 0$

(E'_{fi} *et* E_c *définissent un partitionnement des places en entrée de* t_{fi} *en deux sous-ensembles disjoints*)

(3) $\forall t_{fi} \in T_f, \forall p \in E'_{fi} : p^{\cdot} \cap (Tf - \{ t_{fi} \}) = \varnothing$

(*Pour toute transition* $t_{fi} \in T_f$, *les places contenues dans* E'_{fi} *ne sont en entrée d'aucune autre transition de* T_f)

(4) $\exists t_{fi} \in T_f, \exists p \in E'_{fi} : |p^{\cdot}| > 1$

(*Il existe une transition* $t_{fi} \in T_f$, *telle que* E'_{fi} *contient au moins une place partagée*)

(5) $\forall t_{fi} \in T_f : t_{fi}^{\cdot} = S'_{fi} + S''_{fi}$, $S'_{fi} \cap S''_{fi} = \varnothing$

(S'_{fi} *et* S''_{fi} *définissent un partitionnement des places en sortie de* t_{fi} *en deux sous-ensembles disjoints*).

le choix de décomposition qui consiste pour chaque $t_{fi} \in T_f$ à affecter les places contenues dans E'_{fi} et S'_{fi} au sous-réseau R_I', les places contenues dans E_c et S''_{fi} au sous-réseau R_I'', et le prédicat éventuellement défini par $\Psi(t_{fi})$ au réseau R_I'' est impossible à réaliser.

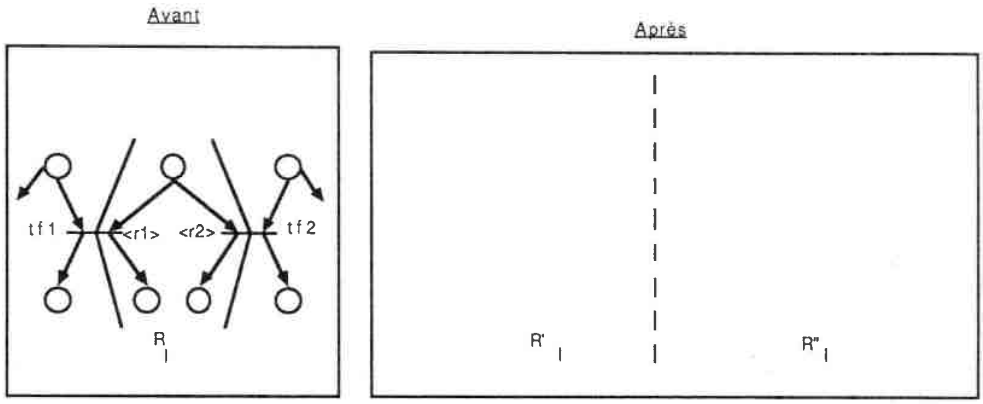


Figure 7.11

Avant de vérifier les prédicats qui lui sont associés, le sous-réseau R_I'' ne peut connaître avec exactitude le marquage des places partagées affectées au sous-réseau R_I'.

d) La règle RD12

Le schéma de décomposition de la règle RD12 se distingue des schémas de décomposition des règles RD09 et RD10 essentiellement par le fait que les prédicats associés aux transitions frontières en conflit structurel ne sont pas globalement affectés à un seul sous-réseau, mais répartis entre les deux sous-réseaux.

Définition 7.12 :

Soit R_I un RdPI à décomposer par partitionnement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Pour tout sous-ensemble $T_f = \{ t_{f1}, ..., t_{fn} \} \subset T$ de transitions frontières tel que :

(1) $E_c = \{ p_j \mid p_j^{\cdot} = T_f \} \neq \varnothing$

(E_c *désigne l'ensemble des places communes aux transitions contenues dans* T_f)

(2) $\forall t_{fi} \in T_f : {}^{\cdot}t_{fi} = E'_{fi} + E_c$, $E'_{fi} \cap E_c = \varnothing$, $|E'_{fi}| \geq 0$

(E'_{fi} *et* E_c *définissent un partitionnement des places en entrée de* t_{fi} *en deux sous-ensembles disjoints*)

(3) $\forall t_{fi} \in t_f, \forall p \in E'_{fi} : p^{\cdot} \cap (T_f - \{ t_{fi} \}) = \varnothing$

(*Pour toute transition* $t_{fi} \in T_f$, *les places contenues dans* E'_{fi} *ne sont en entrée d'aucune autre transition frontière*)

(4) $\forall t_{fi} \in T_f : t_{fi}^{\cdot} = S'_{fi} + S''_{fi}$

(S'_{fi} *et* S''_{fi} *définissent un partionnement des places en sortie de* t_{fi} *en deux sous-ensembles disjoints*).

le choix de décomposition qui consiste pour chaque transition $t_{fi} \in T_f$ à affecter les places contenues dans E'_{fi} et S'_{fi} au sous-réseau R_I', les places contenues dans E_c et S''_{fi} au sous-réseau R_I'', puis à répartir les prédicats associés à ces transitions entre R_I' et R_I'' est impossible à réaliser.

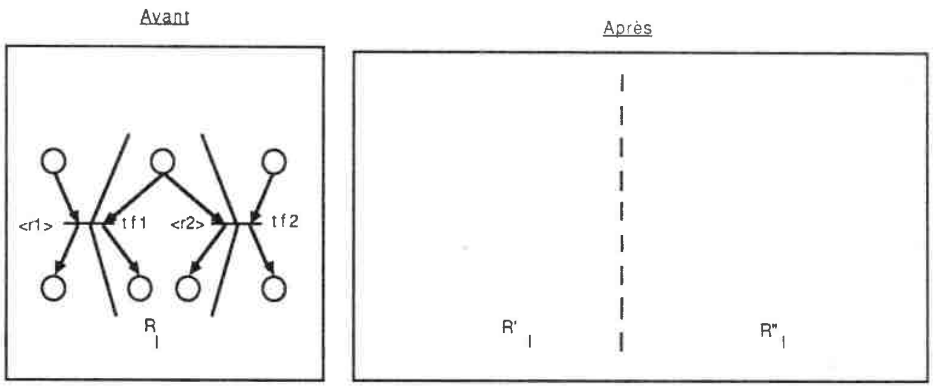


Figure 7.12

Les transitions frontières concernées étant en conflit structurel, la prise en compte de l'occurrence d'un événement peut invalider la précondition pour la prise en compte des occurrences des autres événements associés aux autres transitions. Si ces événements sont répartis entre les deux sous-réseaux, cette contrainte ne peut plus être garantie sans risque d'interblocage. On ne peut en effet éviter aux deux sous-réseaux de prendre en compte au même instant des occurrences d'événements qui s'invalident mutuellement.

7.2 L'APPROCHE DE DECOMPOSITION PAR DEDOUBLEMENT

La procédure de décomposition par dédoublement permet de répartir des opérations d'entrée/sortie entre deux sous-réseaux ayant même structure de contrôle que le réseau initial, en fonction des choix de regroupement décidés par le concepteur. Par rapport à la procédure de décomposition par partitionnement, cette procédure peut conduire à de meilleures solutions lorsque les contraintes de structuration imposent la répartition d'opérations d'entrée/sortie trés fortement liées. Elle repose sur deux règles : la règle RD13 qui correspond à un schéma de décomposition réalisabble, et la règle RD14 qui

caractérise un schéma de décomposition irréalisable.

a) La règle RD13

Le schéma de décomposition de cette règle se distingue par le fait que lorsque plusieurs transitions apparaîssent en sortie d'une place, les prédicats associés à ces transitions sont tous affectés au même sous-réseau.

Définition D 7.13 :

Soit R_I un RdPI à décomposer par dédoublement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Les règles de transformation correspondant au choix de regroupement des entrées/sorties caractérisé de la façon suivante :

(1) $\forall\ p_j \in P,\ \forall\ t_1, t_2 \in p_j^{\cdot}$:

$t_1 \neq t_2\ ,\ \Psi\ (t_1) \neq pr_0\ ,\ \Psi\ (t_2) \neq pr_0$

$\Rightarrow$ affectation de $\Psi\ (t_1)$ = affectation de $\Psi\ (t_2)$

(*Les prédicats associés à des transitions en sortie d'une même place sont affectés au même sous-réseau*).

(2) $\forall\ p_j \in P$: $\Phi\ (p_j)$ est affecté à l'un des deux sous-réseaux

sont définies comme suit :

(3) $T' <- T\ ,\ T'' <- T$

$P' <- P\ ,\ P'' <- P$

Pre' = Pre" = Pre , Post' = Post" = Post

(R *est dupliqué en deux sous-réseaux* R' *et* R")

(4) $\forall\ t_i \in T$ telle que $\Psi(t_i) \neq pr_0$:

$\Psi'(t_i) = \Psi(t_i)$ -> ! M_i si $\Psi(t_i)$ est affecté à R'

? M_i sinon

$\Psi"(t_i) = \Psi(t_i)$ -> !M_i si $\Psi(t_i)$ est affecté à R"

? M_i sinon

$\forall\ t_i \in T$ telle que $\Psi(t_i) = pr_0$: $\Psi'(t_i) = !\ M_i$, $\Psi"(t_i) = ?\ M_i$

(*Le prédicat associé à chaque transition* $t_i \in T$ *est associé à la transition de même nom dans le sous-réseau auquel il est affecté, et une synchronisation par rendez-vous est définie entre les transitions de même nom dans* R_I' *et* $R_I"$)

(5) $\forall\ p_j \in P$: $\Phi'(p_j) =$ $\Phi(p_j)$ si $\Phi(p_j)$ est affecté à R'

NOP sinon

$\Phi"(p_j) =$ $\Phi(p_j)$ si $\Phi(p_j)$ est affecté à R"

NOP sinon

(*L'opération associée à chaque place* $p_j \in P$ *est associée à la place de même nom dans le sous-réseau auquel elle est affectée*). ⊗

(Validation en annexe C chapitre C.8)

Exemple

Comme le montre cet exemple, les transformations de la règle RD13 conduisent les deux sous-réseaux R_I' et $R_I"$ à évoluer de façon synchrone, en franchissant dans le même ordre les mêmes transitions.

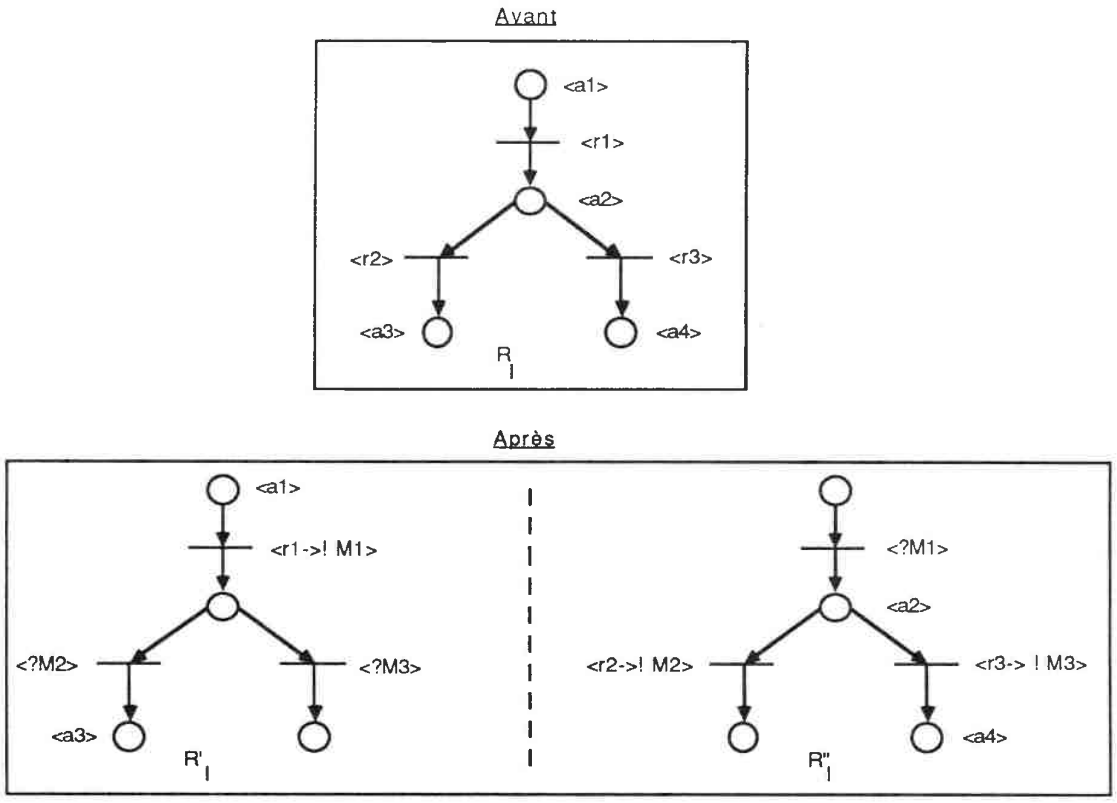


Figure 7.13

b) La règle RD14

Le schéma de décomposition de cette règle se distingue du schéma de décomposition de la règle précédente par le fait que les prédicats associés à des transitions en sortie d'une place sont répartis entre les deux sous-réseaux.

Définition D 7.14:

Soit R_I un RdPI à décomposer par dédoublement en deux sous-réseaux R_I' et R_I'' :

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$.

Les choix de regroupement des entrées/sorties caractérisés comme suit sont impossibles à réaliser :

$\exists\ p_j \in P$, $\exists\ t_1 , t_2 \in p_j^{\cdot}$ tels que :

$t_1 \neq t_2$, $\Psi\ (t_1) \neq pr_0$, $\Psi\ (t_2) \neq pr_0$,

affectation de $\Psi\ (t_1) \neq$ affectation de $\Psi\ (t_2)$. $\otimes$

La situation ainsi caractérisée est en effet équivalente à celle déjà rencontrée pour la règle RD12. Les deux sous-réseaux ne peuvent coopérer pour franchir les transitions en conflit sans risque d'interblocage.

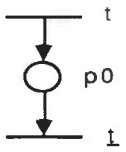
7.3 L'APPROCHE DE DECOMPOSITION PAR SUBSTITUTION DE SOUS-RESEAU BIEN FORME

Cette procédure de décomposition correspond à une forme particulière de décomposition par partitionnement par rapport à deux transitions frontières qui doivent nécessairement être la transition initiale et la transition finale d'un bloc bien formé [Valette 76]. La décomposition s'effectue alors en effectuant les deux opérations suivantes :

1) On substitue dans le réseau initial le bloc bien formé par un réseau-transition [André 81] (cf. figure 7.14) pour obtenir le premier sous-réseau.

2) Ensuite, on transforme le bloc bien formé en un réseau bien formé par l'ajout d'une place repos pour obtenir le second sous-réseau.

Le but de cette procédure de décomposition est de permettre ainsi une hiérarchisation du contrôle en distinguant un niveau coordination et un niveau commande locale. Elle repose sur une seule règle de décomposition, la règle RD15, qui peut être composée avec les règles RD04 et RD08 pour former de nouvelles règles.



NB: Un réseau-transition est constitué de deux transitions t et $\underline{t}$ interconnectées par une seule place p_0 :

Pre (p_0, t) = Post $(p_0, \underline{t})$ = 0

Pre $(p_0, \underline{t})$ = Post (p_0, t) = 1

Figure 7.14

a) La règle RD15

Elle est caractérisée par le fait que dans le réseau à décomposer, la transition initiale et la transistion finale du bloc bien formé sont toutes les deux étiquetées par le prédicat toujours vrai.

Définition 7.15 :

Soit R_I un RdPI à décomposer en deux sous-réseaux R_I' et R_I'', par substitution du bloc bien formé R_{Ib} ayant t_{ini} pour transition initiale et t_{fin} pour transition finale:

$R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$

$R_I' = < R' ; V' ; OP' ; PR' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$

$R_I'' = < R'' ; V'' ; OP'' ; PR'' ; \Phi'' ; \Psi'' >$ avec $R'' < P'', T'' ; Pre'', Post'' >$

$R_{Ib} = < R_b ; V_b ; OP_b ; PR_b ; \Phi_b ; \Psi_b >$ avec $R_b = < P_b, T_b ; Pre_b, Post_b >$.

Si

(1) $\Psi_b (t_{ini}) = \Psi_b (t_{fin}) = pr_0$

(*La transition initiale et la transition finale sont étiquetées par le prédicat toujours vrai*)

(2) la transition t_{ini} est t-sensibilisée simultanément avec elle même.

les règles de transformation correspondant au choix de décomposition qui consiste à affecter le bloc bien formé R_{Ib} au sous-réseau R_I'' et le reste du réseau R_I au sous-réseau R_I' sont définis comme suit :

Dans R_I'

(3) $T' <- T - T_b + \{ t_{ini}, t_{fin} \}$

(*Dans T' on inclut toutes les transitions de T qui n'appartiennent pas au bloc bien formé, ainsi que les transitions* t_{ini} *et* t_{fin})

(4) $P' <- P - P_b + \{ p_I \}$

(*Dans P' on inclut toutes les places de P qui n'appartiennent pas au bloc bien formé, et une nouvelle place* p_I)

(5) $\forall t \in T' - \{ t_{fin} \}$, $\forall p \in P'$: $Pre' (p, t) = Pre (p, t)$ si $p \neq p_I$ 0 sinon

(*A l'exception de* t_{fin}, *les transitions conservent les mêmes places en entrée dans* R_I *et* R_I')

(6) $\forall p \in P'$: $Pre' (p, t_{fin}) = 1$ si $p = p_I$ 0 sinon

(*La transition* t_{fin} *reçoit en entrée la place* p_I)

(7) $\forall t \in T' - \{ t_{ini} \}$, $\forall p \in P$: $Post' (p, t) = Post (p, t)$ si $p \neq p_I$ 0 sinon

(*A l'exception de* t_{ini} , *les transitions conservent les mêmes places en sortie dans* R_I *et* R_I')

(8) $\forall p \in P' : Post'(p, t_{ini}) = 1$ si $p = p_I$ 0 sinon

(*La transition* t_{ini} *reçoit en sortie la place* p_I).

Dans R_I''

(9) $T'' \leftarrow T_b$

(*Dans T" on inclut toutes les transitions du bloc bien formé*)

(10) $P'' \leftarrow P_b + \{ \underline{p_I} \}$

(*Dans P" on inclut toutes les places du bloc bien formé, et une nouvelle place* $\underline{p_I}$)

(11) $M'(\underline{p_I}) = 1$ (*On marque la nouvelle place* $\underline{p_I}$ *à* 1)

(12) $\forall t \in T'' - \{ t_{ini} \}, \forall p \in P'' : Pre''(p, t) = Pre(p, t)$ si $p \neq \underline{p_I}$ 0 sinon

(*A l'exception de* t_{ini}, *les transitions conservent les mêmes places en entrée dans* R_I *et* R_I'')

(13) $\forall p \in P'' : Pre''(p, t_{ini}) = 1$ si $p = \underline{p_I}$ 0 sinon

(*La transition* t_{ini} *reçoit en entrée* $\underline{p_I}$)

(14) $\forall t \in T'' - \{ t_{fin} \}, \forall p \in P'' : Post''(p, t) = Post(p, t)$ si $p \neq \underline{p_I}$ 0 sinon

(*A l'exception de* t_{fin}, *les transitions conservent les mêmes places en sortie dans* R_I *et* R_I'').

(15) $\forall p \in P'' : Post''(p, t_{fin}) = 1$ si $p = p_I$ 0 sinon

(*La transition* t_{fin} *reçoit en sortie* p_I)

Dans R_I' *et* R_I''

(16) $\Psi'(t_{ini}) = ? M_i$, $\Psi''(t_{ini}) = ! M_i$

(*On définit une synchronisation par rendez-vous entre* t_{ini} *dans* RI' *et*

t_{ini} *dans* RI")

(17) $\Psi'(t_{fin}) = ?\ M_j$, $\Psi''(t_{fin}) = !\ M_j$

(*On définit une synchronisation par rendez-vous entre* t_{fin} *dans* R_I' *et* t_{fin} *dans* R_I'')

(18) $\forall\ t \in T' - \{ t_{ini}, t_{fin} \} : \Psi'(t) = \Psi(t)$

(*Les transitions autres que* t_{ini} *et* t_{fin} *conservent la même interprétation dans* R_I *et* R_I')

(19) $\forall\ p \in P' - \{ p_I \} : \Phi'(p) = \Phi(p)$

(*Les places autres que* p_I *conservent la même interprétation dans* R_I *et dans* R_I')

(20) $\forall\ p \in P'' - \{ \underline{p_I} \} : \Phi''(p) = \Phi(p)$

(*Les places autres que* $\underline{p_I}$ *conservent la même interprétation dans* R_I *et dans* R_I''). ⊗

La figure 7.15 donne la formulation graphique de la règle RD15.

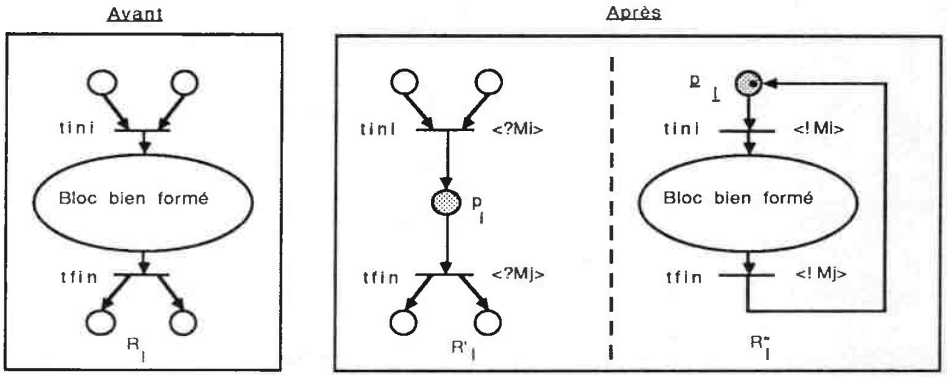


Figure 7.15

(Validation en annexe C chapitre C.9)

b) Composition de la règle RD15 avec les règles RD04 et RD08

Nous donnons à titre indicatif dans les figures 7.16 et 7.17 la formalisation graphique des compositions possibles de la règle RD15 avec les règles RD04 et RD08.

Règle RD15 + règle RD08 :

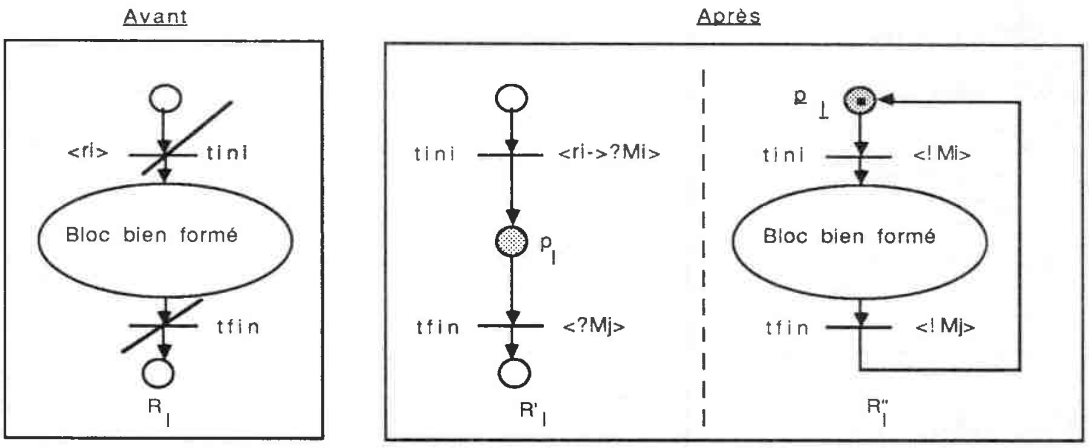


Figure 7.16

Règle RD15 + règle RD04 :

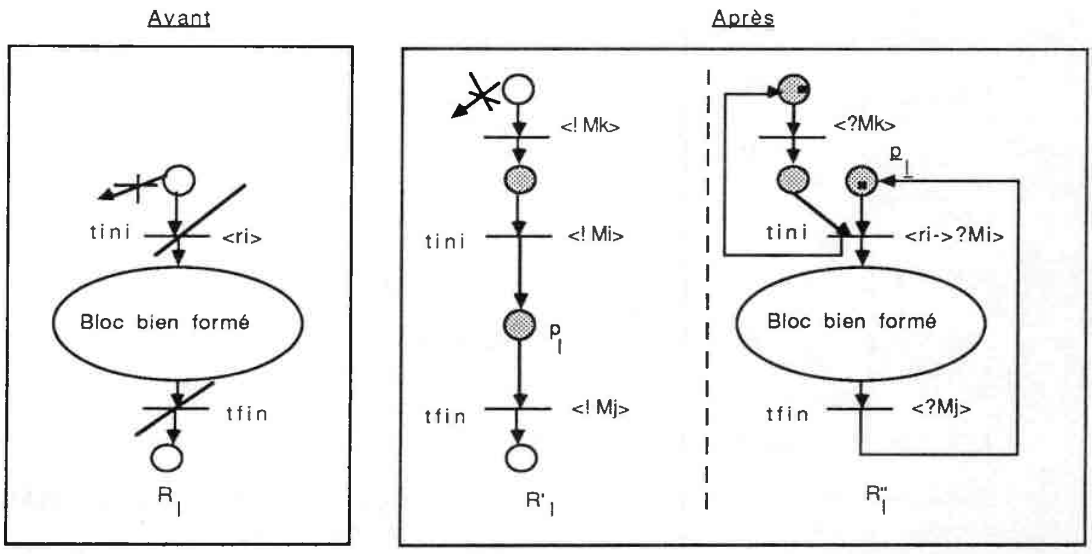


Figure 7.17

7.4 LA PRISE EN COMPTE DES PROBLEMES DE PRESENTATION

Le but des règles de transformation que nous abordons dans cette section n'est pas d'offrir des possibilités additionnelles de décomposition, mais plutôt d'assurer une bonne présentation des sous-réseaux que l'on peut synthétiser en utilisant les règles précédentes.

7.4.1 L'AJOUT DE PLACES IMPLICITES

La règle d'ajout de places implicites permet, lorsque les choix de regroupement du concepteur sont connus, d'effectuer préalablement des transformations sur le réseau à décomposer, de manière à préserver dans certaines situations la connexité des sous-réseaux qui résulteront de la décomposition.

Si $R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$ est un RdPI à décomposer par partitionnement, cette règle concerne toute transition frontière $t_f \in T$ ayant toutes ses places en sortie affectées à un seul sous-réseau (disons R_I'). En plus, pour cette transition t_f, toute transition qui apparaît en entrée ou en sortie de ses places de sortie doit être aussi une transition frontière :

(1) $\forall p \in t_f^{\cdot} : t \in {}^{\cdot}p \Rightarrow$ t est une transition frontière.

(2) $\forall p \in t_f^{\cdot} : t \in p^{\cdot} \Rightarrow$ t est une transition frontière.

Les transformations qu'on effectue pour une transition frontière t_f de ce type consistent alors à ajouter pour chaque place p_j sortie de cette transition, une place $\underline{p_j}$ qui reçoit en entrée et en sortie les mêmes transitions que p_j, et qu'on affecte au deuxième sous-réseau R_I'' :

(3) $\forall\ p_j \in t_{f'}$:

- $P \leftarrow P + \{ \underline{p_j} \}$
- $\forall\ t_i \in T$: $Pre(\underline{p_j}, t_i) = Pre(p_j, t_i)$, $Post(\underline{p_j}, t_i) = Post(p_j, t_i)$

Chaque place $\underline{p_j}$ aura de ce fait toujours le même marquage que la place p_j qui lui correspond. Les places qu'on ajoute de cette façon sont donc des places implicites qui ne modifient pas le comportement décrit par le réseau initial. Comme le montre l'exemple de la figure 7.18, lorsque toutes les places en sortie d'une transition frontière sont affectées au même sous-réseau, leur introduction permet de préserver la connexité de l'autre sous-réseau.

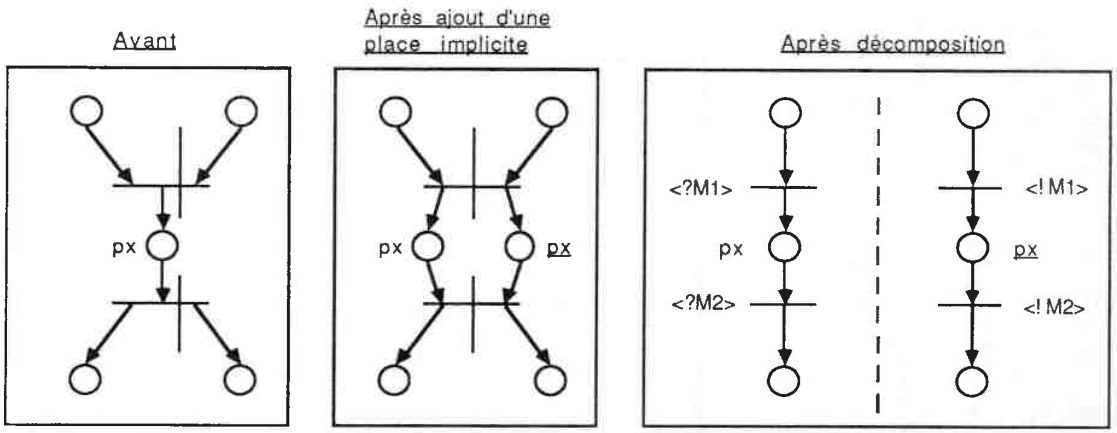


Figure 7.18

7.5.2 TRANSFORMATION DES COMMUNICATIONS ASYNCHRONES EXPRIMEES A L'AIDE D'UN RENDEZ-VOUS

Cette règle de transformation vise également à assurer la connexité des sous-réseaux issus de l'application des procédures de décomposition. Cependant, à l'inverse de la règle précédente, elle n'intervient qu'après décomposition. Elle consiste tout simplement à traduire par des primitives de communication

asynchrone, des communications asynchrones exprimées à l'aide de rendez-vous. Les figures 7.19 à 7.24 formalisent les différentes transformations possibles. Ces transformations se déduisent directement de la sémantique que nous avons définie pour les opérations de communication synchrone et asynchrone lorsqu'elles sont associées à des transitions (c.f. sections 2.5.2 et 2.5.3).

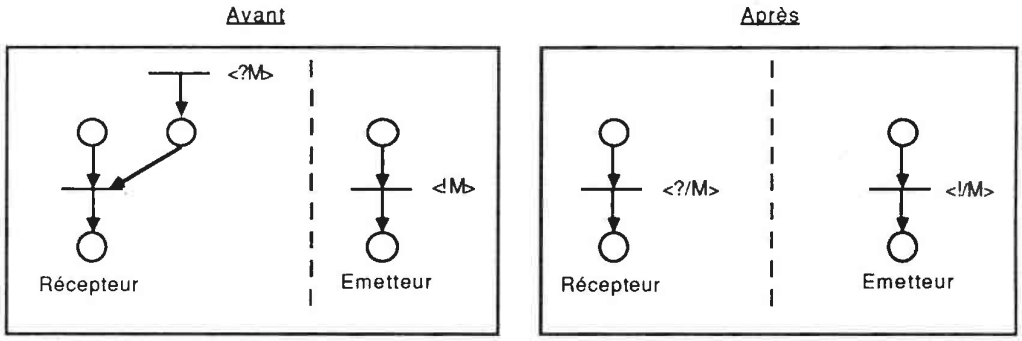


Figure 7.19

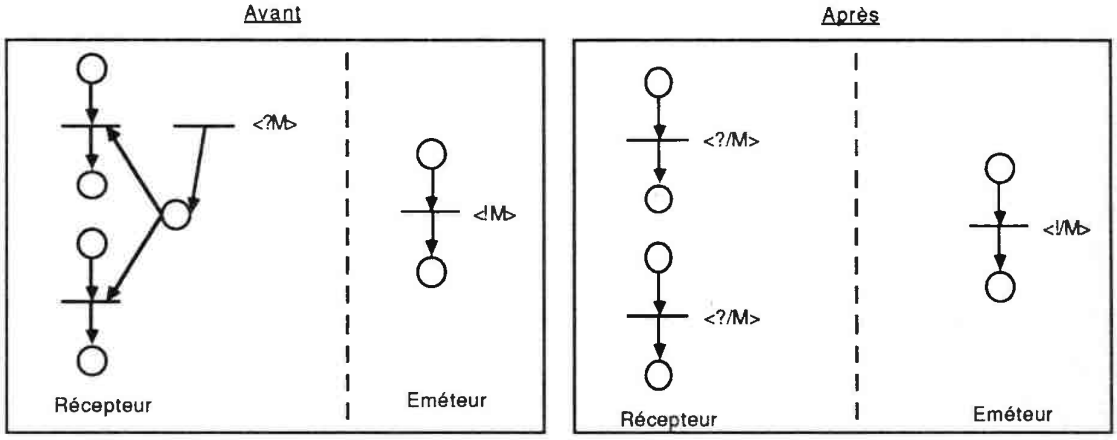


Figure 7.20

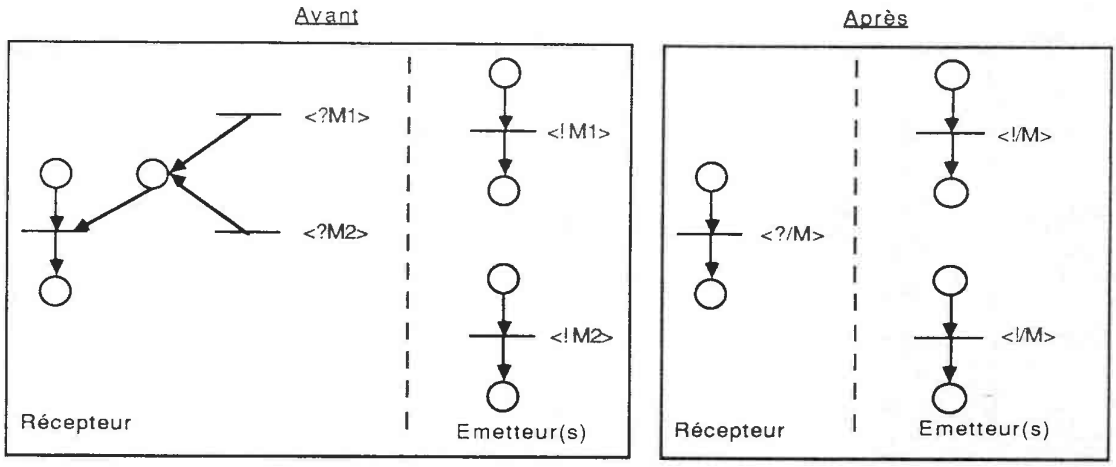


Figure 7.21

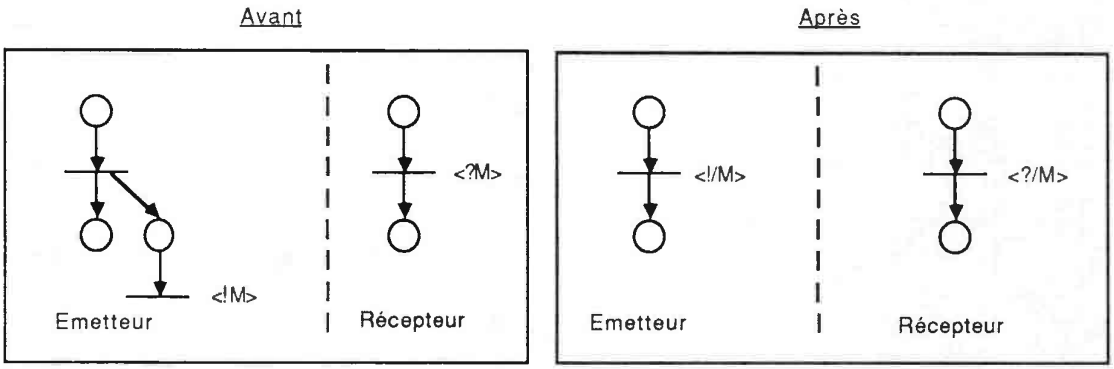


Figure 7.22

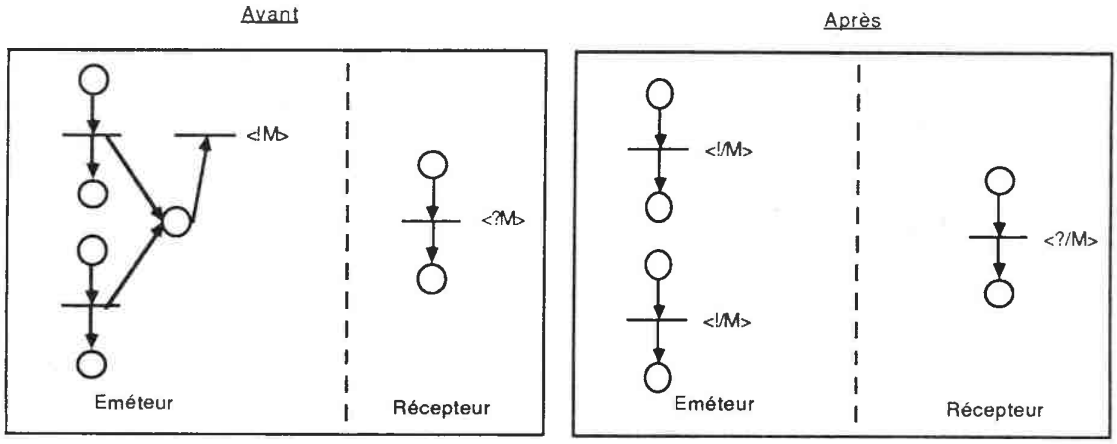


Figure 7.23

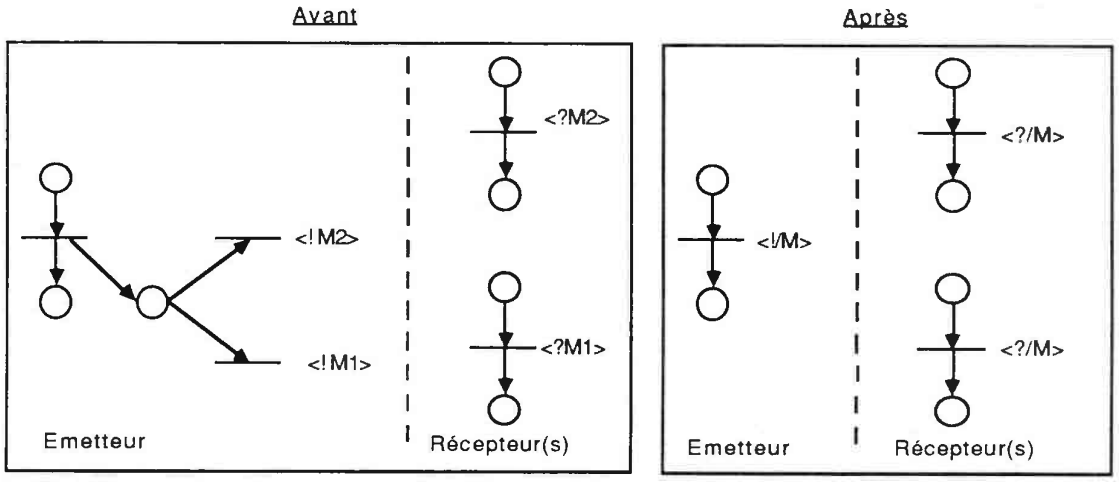


Figure 7.24

7.4.3 REDUCTION DES SOUS-RESEAUX

La dernière règle de transformation que nous abordons concerne la réduction des sous-réseaux issus de l'application de la procédure de

décomposition par dédoublement. Elle consiste à appliquer la règle de réduction par substitution de place sur des places substituables

-étiquetées par NOP
-en sortie d'une seule transition
-et en entrée d'une seule transition étiquetée uniquement que par une opération de communication par rendez-vous, et n'ayant en sortie qu'une place unique, non partagée et étiquetée par NOP.

La figure 7.25 illustre cette règle de transformation. En supprimant des sous-réseaux les places substituables et leurs transitions de sortie qui ne sont liées à aucune interaction avec l'environnement, elle permet de réduire les points de communication entre sous-réseaux. En procédant par fusion des sous-réseaux, il est facile de se convaincre à partir de la figure 7.25 que cette règle de transformation ne modifie pas le comportement externe initialement spécifié.

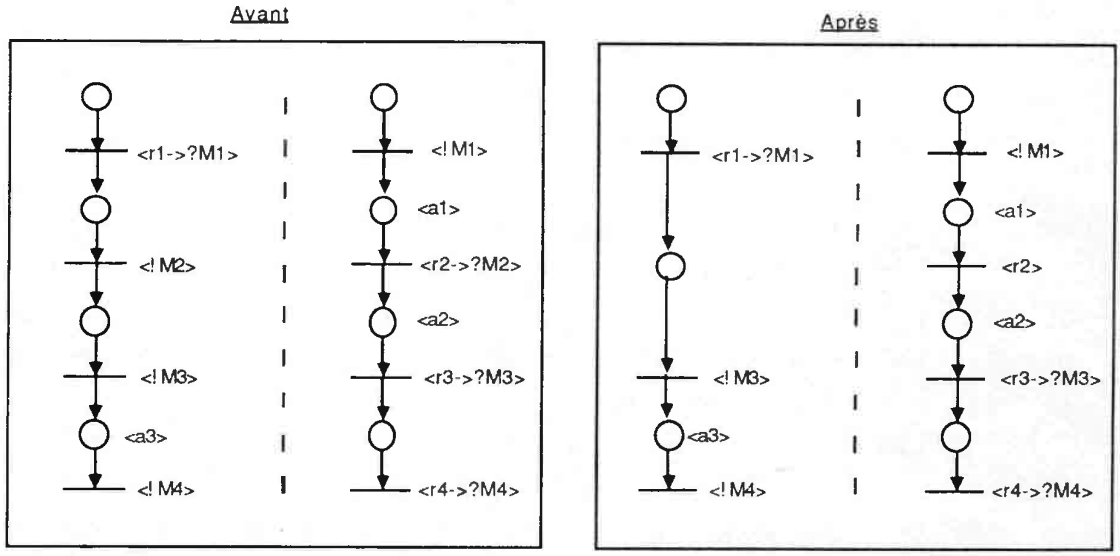


Figure 7.25

7.5 L'ALGORITHME DE DECOMPOSITION

Cet algorithme de décomposition illustre assez bien la démarche de conception préconisée.

```
Non_fin_de_décomposition := vrai ;
tantque     Non_fin_de_décomposition     faire
    début
        Lancer    Sélection_du_sous_réseau_à_décomposer ;
        Lancer    Choix_de regroupement_des_entrées_sorties ;
        Lancer     Vérification_de_choix_de_regroupement
        Si   OK   alors
          début
              Lancer Selection_de_la_procédure_de_décomposition ;
              Si     procédure_de_décomposition_par_partitionnement
              alors  Lancer     ajout_de_places_implicites;
              Lancer décomposition;
              si  décomposition_impossible   alors
                afficher_message
              sinon      sauvegarder_sous_réseaux_résultant;
           fin
        Sinon   afficher_message :
        fin_décomposition ?
    fin;
Lancer   transformation_des_communication_asynchrones_exprimées_
         par_rendez-vous ;
Lancer    Réduction_des_sous_réseaux.
```

N.B.: La vérification sur les choix de regroupement du concepteur porte essentiellement sur la localité des variables calculées et des variables d'entrée. Cette vérification s'impose du fait de l'impossibilité matérielle de réaliser des communications par variables partagées dans une architecture répartie. Par ailleurs, du fait des aléas dans les délais de transmission, si on autorise

l'association d'une variable d'entrée à plusieurs sites, il sera ensuite impossible de garantir qu'à tout instant la valeur de cette variable sera la même sur chacun des sites. Dans certaines situations, il sera donc impossible de faire évoluer un système en respectant les contraintes spécifiées. De ce fait, toute variable calculée ou toute variable d'entrée utilisée dans un sous-réseau doit nécessairement être locale à ce sous-réseau.

7.6 EXEMPLES

Un prototype mettant en oeuvre les règles de décomposition par partitionnement a été développé sur IBM PC en Turbo Pascal. Dans ce prototype, nous prenons tout d'abord en compte les choix de décomposition du concepteur. Ensuite, nous procédons par "pattern matching" pour déterminer pour chaque transition frontière le schéma de décomposition qui lui correspond. Si aucun des schémas identifiés n'est irréalisable, nous appliquons alors les règles de transformation correspondantes.

Les trois premiers exemples que nous présentons sont des exemples de décomposition réalisés à l'aide de ce prototype. Les réseaux utilisés n'ont aucune signification particulière. Ces exemples sont des cas extrêmes qui ont l'avantage de bien mettre en évidence tout l'intérêt de la distinction d'un niveau conceptuel et d'un niveau organisationnel. Les sous-réseaux qu'on obtient après décomposition sont aussi complexes que les réseau initiaux. Ce qui montre que la complexité algorithmique d'un système de commande réparti est bien le produit de deux complexités : celle liée au parallélisme de situation, et celle liée à la coopération qu'induit la répartition d'opérations ayant des liens de dépendance temporelle.

Dans le quatrième exemple, nous abordons selon la démarche préconisée la structuration interne du système de commande de l'atelier de bobinage.

7.6.1 EXEMPLE 1

Dans cet exemple, la décomposition a été effectuée en appliquant sur la transition frontière t_1 la règle RD03.

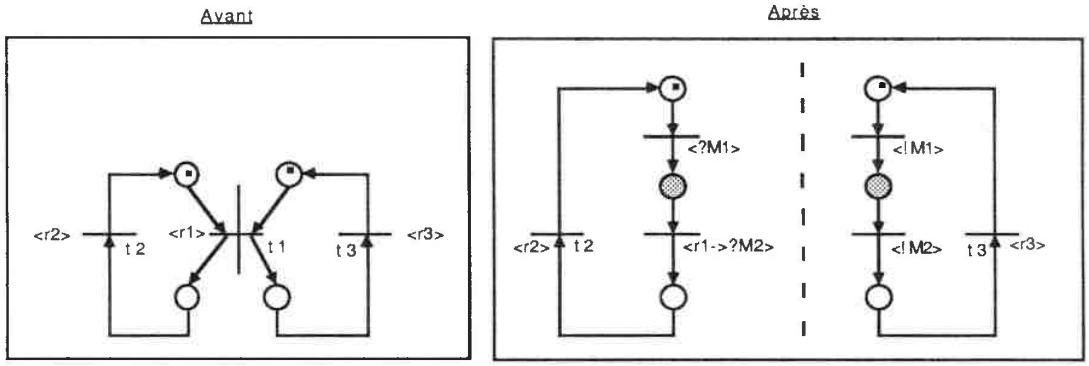


Figure 7.26

7.6.2 EXEMPLE 2

Dans ce deuxième exemple, la règle de décomposition appliquée à la transition frontière t_2 est la règle RD04.

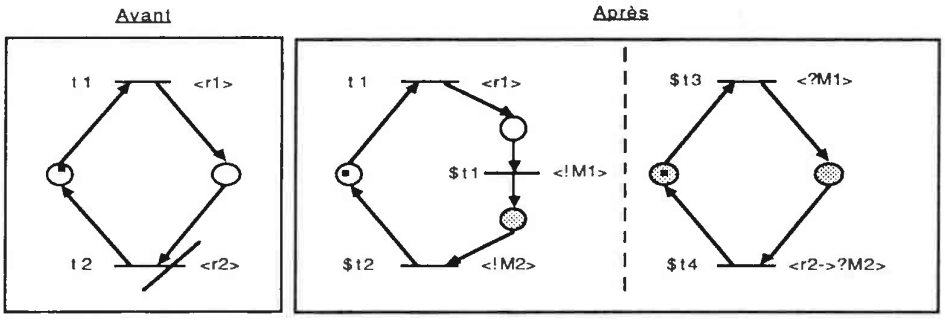
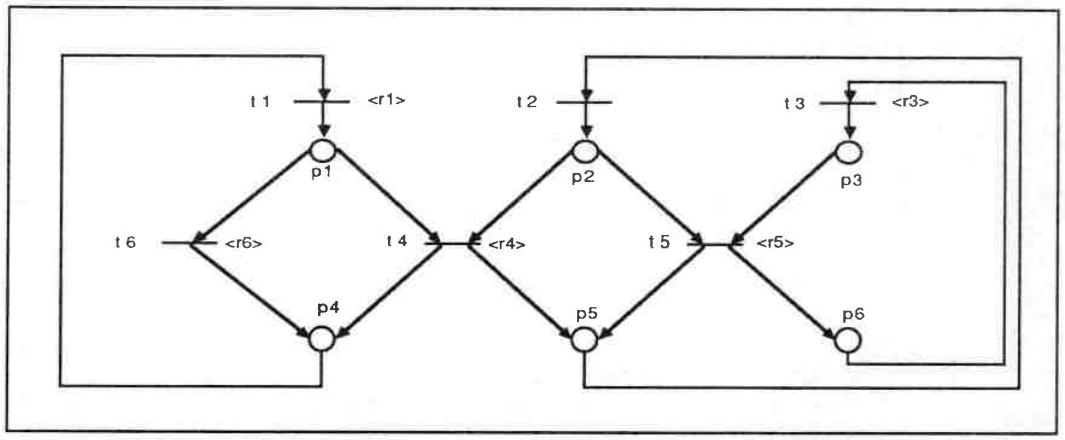


Figure 7.27

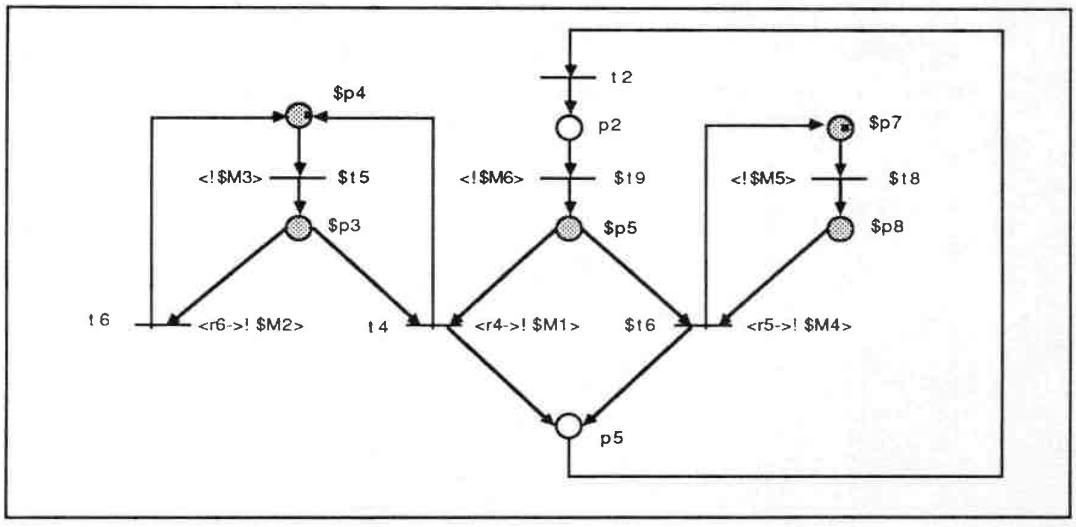
7.6.3 EXEMPLE 3

Dans cet exemple; la décomposition regroupe dans le premier sous-réseau les places p2 et p5, ainsi que les prédicats r4, r5, et r6, et dans le deuxième sous-réseau les places p1, p3, p4, et p6, ainsi que les prédicats r1 et r3. La règle RD09 a été appliquée aux transitions frontières t_4 et t_6, et la règle RD10 aux transitions frontières t_4 et t_5.

Avant



Après



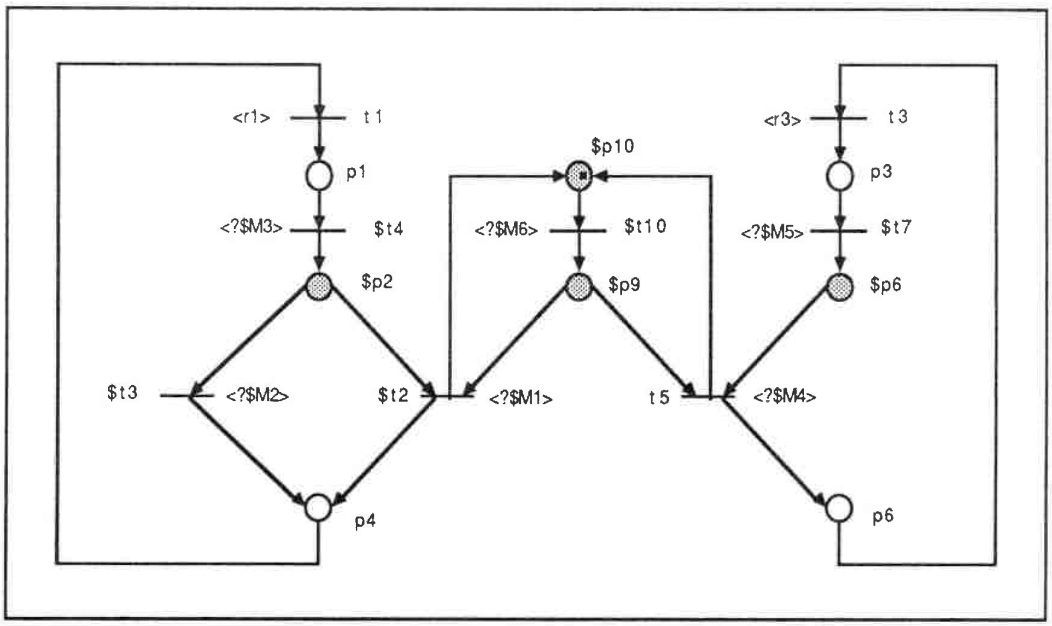


Figure 7.28

Ce qui suit donne la description textuelle du réseau initial, et la description textuelle des sous-réseaux résultant de la décomposition. La description des sous-réseaux a été entièrement synthétisée par le prototype à partir de la description initiale et des choix de décomposition. On remarquera tout particulièrement que la partie environnement d'un sous-réseau ne comporte strictement que les informations propres à ce sous-réseau.

Réseau initial

```
rdpi exemple4;

  rdp

    noeuds p1,p2,p3,p4,p5,p6 : place;
           t1,t2,t3,t4,t5,t6 : transition;


    arcs    entree t1 = p4;
            sortie t1 = p1;
            entree t2 = p5;
            sortie t2 = p2;
            entree t3 = p6;
            sortie t3 = p3;
            entree t4 = p1,p2;
            sortie t4 = p4,p5;
            entree t5 = p2,p3;
            sortie t5 = p5,p6;
            entree t6 = p1;
            sortie t6 = p4;

  frdp;

  environnement

    variables

            entree e1,e2,e7 : entier;
                   e3,e4,e5,e6 : booleen;

            sortie s1,s5,s6 : booleen;
                   s2,s3,s4 : entier;

            calculees c1 : entier;
                      c2 : booleen;

            temporisation tt1,tt2 : booleen;

    conditions

            cond1 = e7 <= 20 ;
            cond2 = ( e1 + 2 ) > s2 ;
            cond3 = (( e1 = 2 ) et tt1) ou e3 ;
            cond4 = e3 ou (cond2 et ( e1 = 0));

    evenements

            evt1 = \^ e6 ;
            evt2 = \! ((e1 = 1) ou  (e2 = 2)) ;
            evt3 = \^ e3 ;
            evt4 = \^ cond1 et \^ cond2 ;
            evt5 = evt1 et evt2 ;
```

```
    actions

            act1 = s2 := 1 ;
            act2 = s3 := ( e1 * 2 ) + ( e2 / 5 ) ;
            act3 = si e1 = 1 alors
                     s1 := e2 <> 10 ;
                     act1 ;
                     s4 := 0
                   fsi ;
            act4 = debut
                     armertemporisation(tt1,120) ;
                     s2 := 5 ;
                     s5 := 7 ;
                     si e2 = 0 alors s2 := 1 ; s3 := 1 fsi ;
                     s1 := faux
                   fin ;
            act5 = desarmertemporisation(tt1) ;
            act6 = s6 := vrai ;

  fenvironnement ;

  interpretation

    receptivite t1 = sur occurrence de evt1 si cond1;
    receptivite t3 = si cond1 ;
    receptivite t4 = sur occurrence de evt2 si cond2 ;
    receptivite t5 = sur occurrence de \! e4 si (e1 + 2) > s2 ;
    receptivite t6 = sur consommation rafraichie de occurrence de evt3 ;
    action p2 = [act1 , act2 , act3] ;
    action p5 = [debut
                   armertemporisation(tt2,10) ;
                   e4 := (2 * e1) / e2 ;
                   si cond2 alors s2 := 1 ; act4 fsi
                 fin] * ;
    action p4 = [act6] ;
    action p6 = [debut e5 := vrai fin] ;

  finterpretation ;

frdpi exemple4.
```

Premier sous-réseau

```
rdp1 EXEMPLE4 ;

  rdp

    noeuds P2, P5, $P3, $P4 ( 1 ), $P5, $P7 ( 1 ), $P8 : place ;
           T2, T4, T6, $T5, $T6, $T8, $T9 : transition ;

    arcs   entree T2 = P5 ;
           sortie T2 = P2 ;
           entree T4 = $P3, $P5 ;
           sortie T4 = P5, $P4 ;
           entree T6 = $P3 ;
           sortie T6 = $P4 ;
           entree $T5 = $P4 ;
           sortie $T5 = $P3 ;
           entree $T6 = $P5, $P8 ;
           sortie $T6 = P5, $P7 ;
           entree $T8 = $P7 ;
           sortie $T8 = $P8 ;
           entree $T9 = P2 ;
           sortie $T9 = $P5 ;

  frdp ;

  environnement

    variables

      entree E1, E2 : entier ;
             E3, E4 : booleen ;

      sortie S2, S3, S4 : entier ;
             S1 : booleen ;

      temporisation TT2, TT1 : booleen ;

    conditions

      COND2 = E1 + 2 > S2 ;

    evenements

      EVT2 = \! ((E1 = 1) ou (E2 = 2)) ;
      EVT3 = \^ E3 ;
```

```
  actions

    ACT1 = S2 := 1 ;
    ACT2 = S3 := E1 * E + E2 / 5 ;
    ACT3 = si E1 > 1 alors
             S1 := E2 <> 10 ;
             ACT1 ;
             S4 := 0
           fsi ;
    ACT4 = debut
             armertemporisation (TT1 , 120) ;
             S2 := 5 ;
             S3 := 7 ;
             si E2 = 0 alors
               S2 := 1 ;
               S3 := 1
             fsi ;
             S1 := faux
           fin ;

fenvironnement ;

interpretation

  receptivite T4 = sur occurrence de EVT2
                   si COND2
                   avec $C1 ! $M1 ;
  receptivite T6 = sur consommation rafraichie de occurrence de EVT3
                   avec $C2 ! $M2 ;
  receptivite $T5 = avec $C3 ! $M3 ;
  receptivite $T6 = sur occurrence de \! E4 si E1 + 2 > S2
                    avec $C4 ! $M4 ;
  receptivite $T8 = avec $C5 ! $M5 ;
  receptivite $T9 = avec $C6 ! $M6 ;
  action P2 = [ACT1 , ACT2 , ACT3] ;
  action P5 = [debut
                armertemporisation (TT2 , 10) ;
                S4 := 2 * E1 / E2 ;
                si COND2 alors
                  S2 := 1 ;
                  ACT4
                fsi
              fin]* ;

  finterpretation ;

frdpi EXEMPLE4.
```

Second sous-réseau

```
rdpl $SS_R2 ;

  rdp

    noeuds P1, P3, P4, P6, $P2, $P6, $P9, $P10 ( 1 ) : place ;
           T1, T3, T5, $T2, $T3, $T4, $T7, $T10 : transition ;

    arcs    entree T1 = P4 ;
            sortie T1 = P1 ;
            entree T3 = P6 ;
            sortie T3 = P3 ;
            entree T5 = $P6, $P9 ;
            sortie T5 = P6, $P10 ;
            entree $T2 = $P2, $P9 ;
            sortie $T2 = P4, $P10 ;
            entree $T3 = $P2 ;
            sortie $T3 = P4 ;
            entree $T4 = P1 ;
            sortie $T4 = $P2 ;
            entree $T7 = P3 ;
            sortie $T7 = $P6 ;
            entree $T10 = $P10 ;
            sortie $T10 = $P9 ;

  frdp ;

  environnement

    variables

      entree E7 : entier ;
             E6 : booleen ;

      sortie S6, S5 : booleen ;

    conditions

      COND1 = E7 <= 20 ;

    evenements

      EVT1 = \^ E6 ;

    actions

      ACT6 = S6 := vrai  ;

  fenvironnement ;
```

```
  interpretation

    receptivite I1 = sur occurrence de EV11
                     si COND1 ;
    receptivite I3 = si COND1 ;
    receptivite I5 = avec $C4 ? $M4 ;
    receptivite $T2 =  avec $C1 ? $M1 ;
    receptivite $T3 =  avec $C2 ? $M2 ;
    receptivite $T4 =  avec $C3 ? $M3 ;
    receptivite $T7 =  avec $C5 ? $M5 ;
    receptivite $T10 =  avec $C6 ? $M6 ;
    action P4 = [ACT6] ;
    action P6 = [S5 := vrai ] ;

  finterpretation ;

frdpi $S5_R2.
```

7.6.4 EXEMPLE 4

Nous abordons à présent la structuration du système de commande de l'atelier de bobinage (c.f. section 2.6.2 fig. 2.37), l'objectif poursuivi étant de regrouper les événements liés au métier (Ereq, Efbob, Ecasse), les événements et les opérations de commande liés au système embarqué (Efdepse, Efretse, Adepse, Aretse, Aarrse, Acf, Aech) et les événements, et les opérations de commande liés à la navette (Efdepn, Efretn, Adepn, Aretn, Aarrn).

L'architecture souhaitée s'obtient en effectuant deux décompositions successives. La première décomposition est une décomposition par partitionnement, par rapport aux transitions frontières t_4, t_5, t_8, t_{10} et t_{12} (cf. figure 7.29). Elle s'obtient en ajoutant une place implicite en sortie de la transition t_4 et en entrée de la transition t_5, puis en appliquant ensuite la règle RD08 aux transitions t_8, t_{10} et t_{12}, et la règle RD02 aux transitions t_4 et t_5. Elle permet de regrouper sur un sous-réseau les événements et les opérations de commande liés à la navette (figure 7.29 (a)), et sur un second sous-réseau les événements et les opérations de commande liés au métier et au système embarqué (figure 7.29 (b)). Le sous-réseau de la figure 7.29 (b) doit donc a son tour être décomposé par dédoublement pour regrouper sur des sous-réseaux distincts, les événements liés au métier d'une part, et les événements et opérations de commande liés au système embarqué de l'autre. Ce dédoublement n'est toutefois pas possible à réaliser : les événements Ecasse et Efdepse qui sont associés à des transitions en conflit structurel doivent être répartis. Ce problème peut être résolu en modifiant la spécification initiale de manière à rendre plus strict l'ordre défini pour la prise en compte des occurrences de ces deux événements. La figure 7.30 rend compte de cette modification, et la figure 7.31 de la décomposition que l'on obtient ensuite. En appliquant alors sur les sous-réseaux résultants d'une part la règle de transformation des communications asynchrones exprimées à l'aide de rendez-vous, et d'autre part, la règle de réduction des sous-réseaux, on obtient la spécification finale de la figure 2.38 (c.f. section 2.6.2).

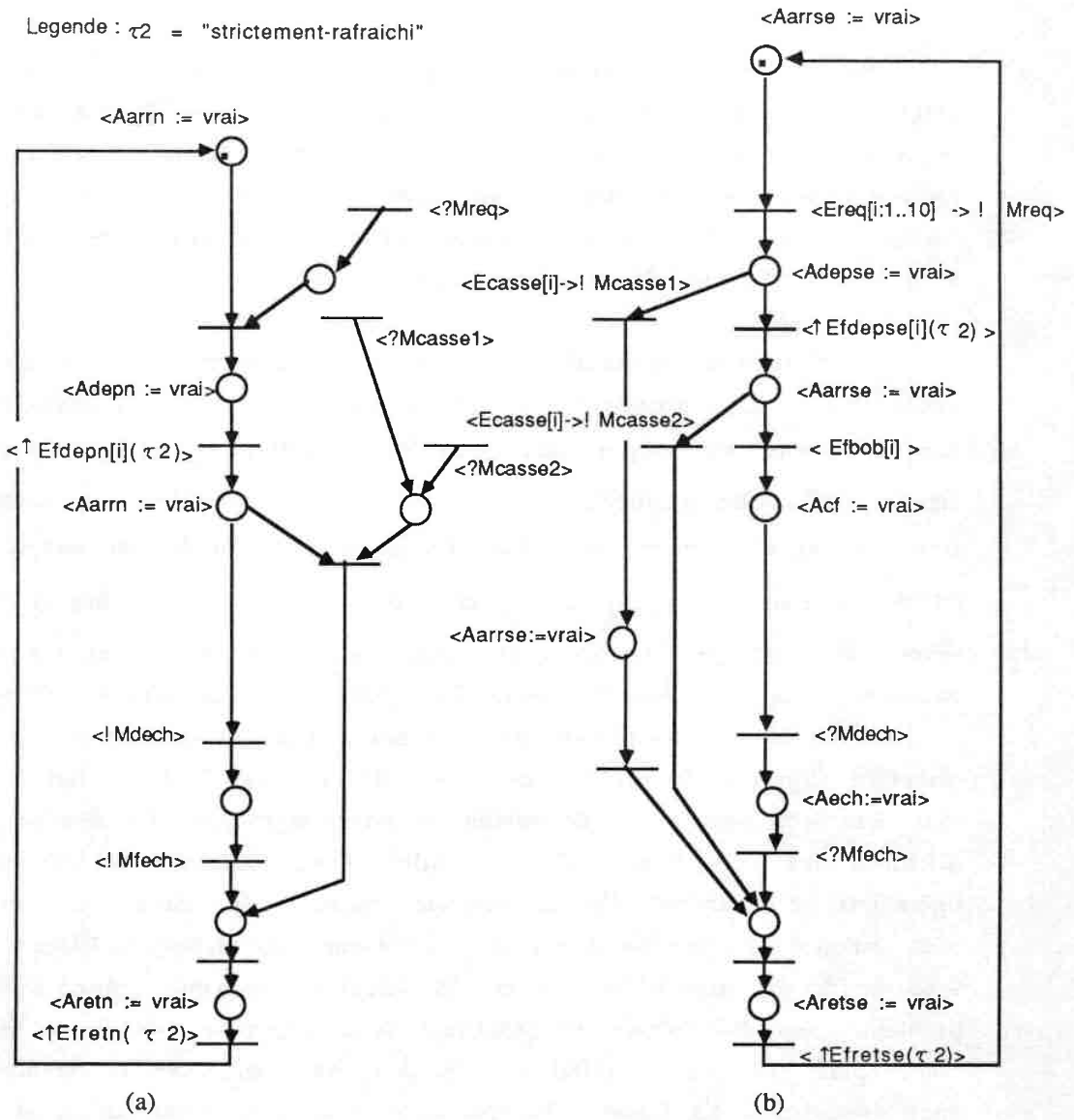

Legende : τ2 = "strictement-rafraichi"
<Aarrn := vrai>
<?Mreq>
<?Mcasse1>
<Adepn := vrai>
<↑ Efdepn[i](τ 2)>
<?Mcasse2>
<Aarrn := vrai>
<! Mdech>
<! Mfech>
<Aretn := vrai>
<↑Efretn(τ 2)>
(a)
<Aarrse := vrai>
<Ereq[i:1..10] -> ! Mreq>
<Ecasse[i]->! Mcasse1>
<Adepse := vrai>
<↑ Efdepse[i](τ 2) >
<Aarrse := vrai>
<Ecasse[i]->! Mcasse2>
< Efbob[i] >
<Acf := vrai>
<Aarrse:=vrai>
<?Mdech>
<Aech:=vrai>
<?Mfech>
<Aretse := vrai>
< ↑Efretse(τ 2)>
(b)

Figure 7.29

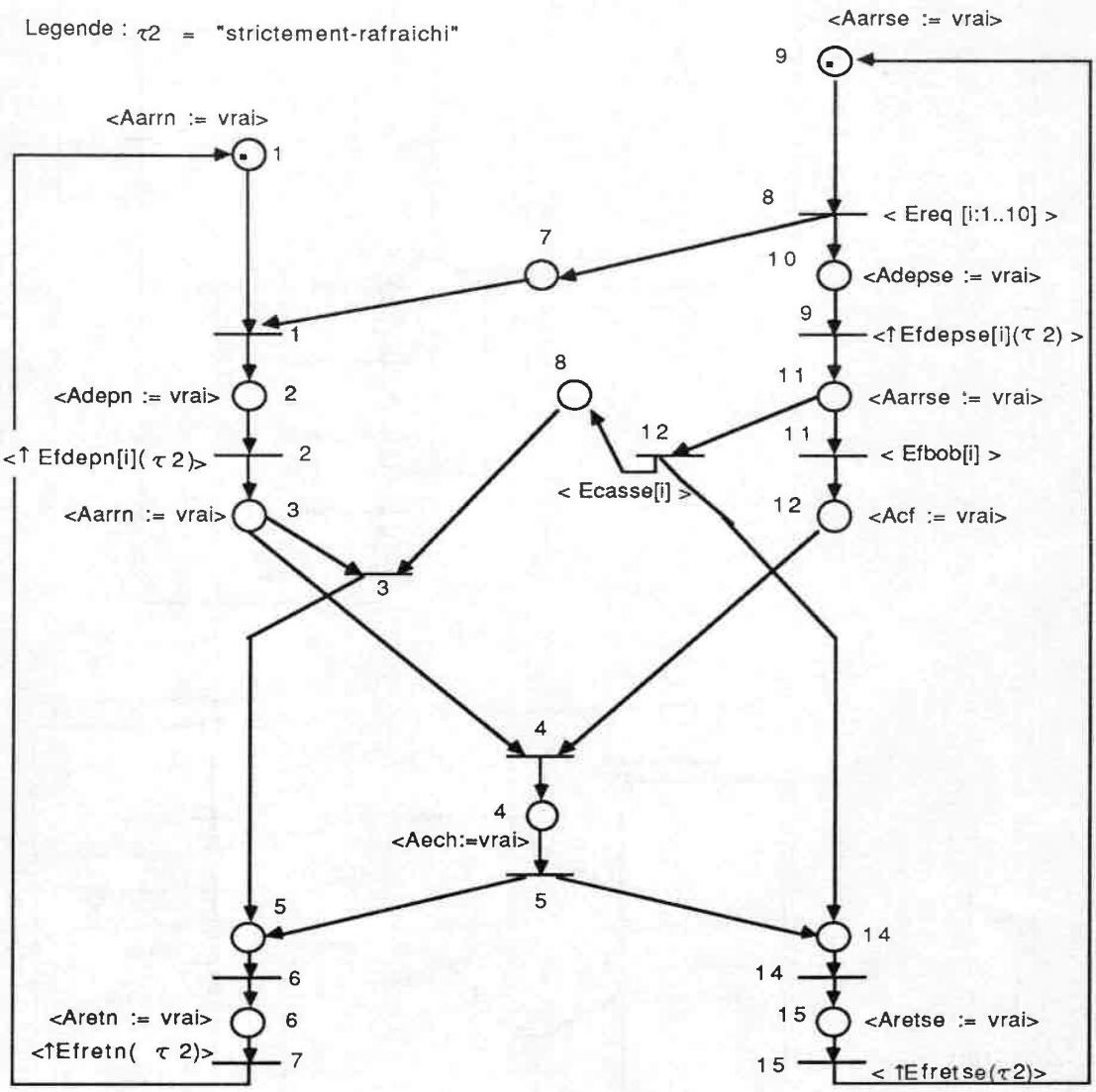
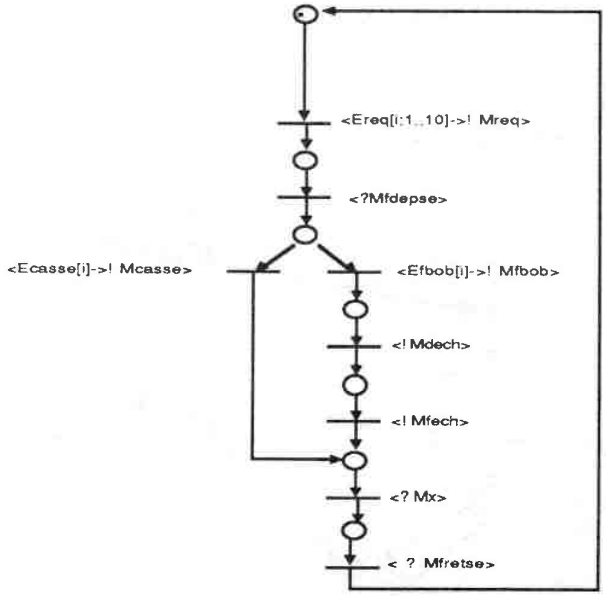

Legende : τ2 = "strictement-rafraichi"
<Aarrse := vrai>
9
<Aarrn := vrai>
1
8
< Ereq [i:1..10] >
7
10
<Adepse := vrai>
9
<↑Efdepse[i](τ 2) >
1
8
11
<Aarrse := vrai>
<Adepn := vrai>
2
12
11
< Efbob[i] >
<↑ Efdepn[i](τ 2)>
2
< Ecasse[i] >
12
<Acf := vrai>
<Aarrn := vrai>
3
3
4
4
<Aech:=vrai>
5
5
14
6
14
<Aretn := vrai>
6
15
<Aretse := vrai>
<↑Efretn(τ 2)>
7
15
< ↑Efretse(τ2)>

Figure 7.30


<Ereq[i:1..10]->! Mreq>
<?Mfdepse>
<Ecasse[i]->! Mcasse>
<Efbob[i]->! Mfbob>
<! Mdech>
<! Mfech>
<? Mx>
< ? Mfretse>

(a)

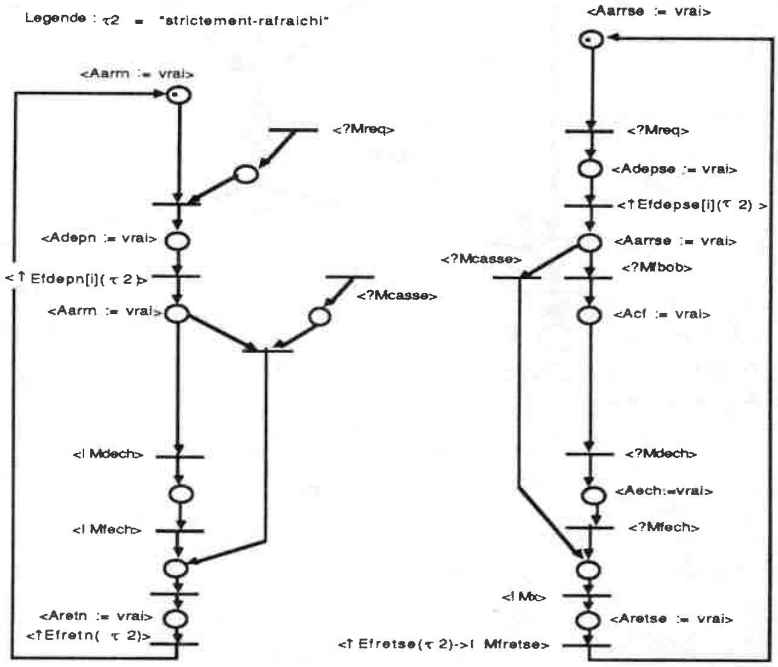

Legende : τ2 = "strictement-rafraichi"
<Aarm := vrai>
<?Mreq>
<Adepn := vrai>
<↑ Efdepn[i](τ 2)>
<?Mcasse>
<Aarm := vrai>
<! Mdech>
<! Mfech>
<Aretn := vrai>
<↑Efretn(τ 2)>
(b)
<Aarrse := vrai>
<?Mreq>
<Adepse := vrai>
<↑Efdepse[i](τ 2) >
<Aarrse := vrai>
<?Mcasse>
<?Mfbob>
<Acf := vrai>
<?Mdech>
<Aech:=vrai>
<?Mfech>
<! Mx>
<Aretse := vrai>
<↑ Efretse(τ 2)->! Mfretse>
(c)

Figure 7.31

7.7 COMPARAISON

Une approche de structuration des systèmes de commande répartis fondée sur la décomposition de RdPI est également proposée dans [Valette 82] [Courvoisier 83]. Dans cette approche, les interactions entre entités communicantes sont exprimées à l'aide de places partagées. Trois règles de décomposition sont alors proposées.

La règle des étiquettes

Les variables (capteurs ou variables internes) figurant dans les étiquettes associées aux transitions doivent impérativement rester locales, c'est-à-dire, ne jamais être utilisées dans plus d'un sous-réseau.

La règle des places

Une place modélisant une communication asynchrone entre sous-réseaux ne doit pas posséder plus d'une transition de sortie, ou alors toutes les transitions de sortie doivent appartenir au même sous-réseau.

La règle des transitions

Lorsqu'une transition commune, c'est-à-dire correspondant à un rendez-vous possède plusieurs places d'entrée, il ne faut pas que plus d'une de ces places soit partagée.

La procédure de décomposition correspondant à ces trois règles est en fait une procédure de décomposition par partitionnement par rapport à des places frontières qui deviennent de fait des places de communication. Par conséquent, la mise en oeuvre de cette procédure est beaucoup plus simple, que la mise en oeuvre de la procédure de décomposition par partitionnement que nous proposons. En contre-partie, elle offre moins de possibilités de décomposition.

Cela tient en partie, au fait que le pouvoir d'expression de la communication à l'aide de places partagées est plus faible que le pouvoir d'expression de la communication à l'aide de rendez-vous. Dans la section 2.5.1, l'étude comparative de ces deux modes d'expression de la communication laissait apparaître cet état de fait.

Ainsi, les deux schémas de décomposition de la figure 7.32, qui sont réalisables en utilisant les règles RD02 et RDO9 ne peuvent être réalisés en utilisant la règles des transitions.

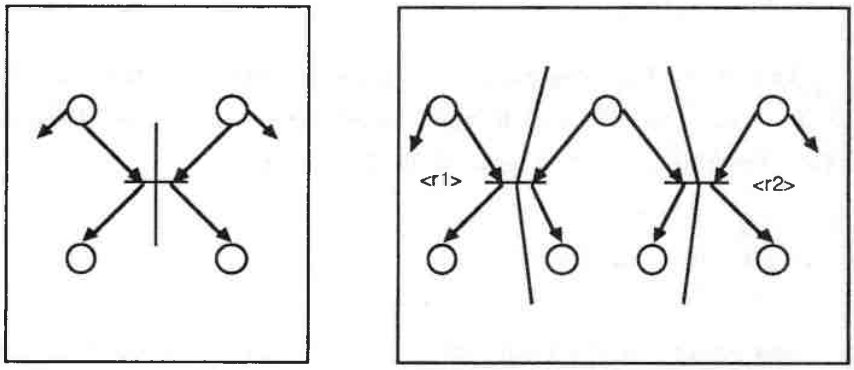


Figure 7.32

Par ailleurs, comme le montre la figure 7.33, les schémas de décomposition qui ne respectent pas la règle des places peuvent être réalisés en utilisant la communication par rendez-vous(c.f. règles RD02 et RD04).

A titre d'illustration, les décompositions que nous avons réalisées dans les deux derniers exemples de la section précédente ne peuvent être réalisées en utilisant la communication par place partagée. Dans l'exemple 3, la transition t_5 ne respecte pas la règle des transitions. Dans l'exemple 4, aucune des règles ne permet de regrouper sur un sous-réseau distinct les événements liés au métier, et sur un autre les événements et les opérations de commande liés au système embarqué.

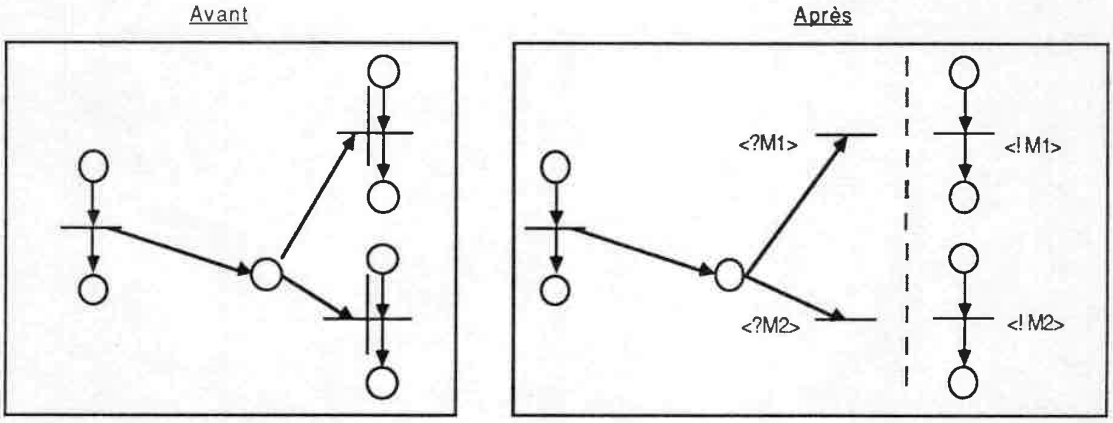


Figure 7.33

CONCLUSION

Pour conclure, nous resumons les points essentiels du travail réalisé et nous indiquons comment il peut être poursuivi.

Le travail réalisé

Le but de ce travail a été de compléter un système de programmation construit autour d'un langage de description d'applications réparties (FLEXI) par un système d'aide à la conception, l'objectif à terme étant de converger vers un environnement de développement ciblé sur les applications temps réel et réparties.

Pour cela, nous avons choisi de nous appuyer sur une démarche de conception qui comporte une étape globale et une étape détaillée, et pour chacune de ces deux étapes, trois niveaux d'abstraction: le niveau conceptuel, le niveau organisationnel, et le niveau opérationnel. Cette démarche permet d'aborder de façon progressive et ordonnée la résolution de divers problèmes de conception, en séparant spécification du problème, organisation du système, et prise en compte des contraintes de langage de programmation et d'architecture physique de réseau de processeurs.

Par ailleurs, nous avons choisi de nous appuyer dans cette démarche sur un modèle de spécification unique que nous avons adapté aux problèmes de la répartition : les réseaux de Petri interprétés. Ce modèle nous a permis d'aborder à la fois les problèmes de spécification du problème, d'organisation du système, de validation, et de simulation. Il peut également servir éventuellement de langage d'implémentation.

L'apport essentiel de notre travail se situe toute fois beaucoup plus au niveau des outils logiciels que nous proposons pour supporter la démarche préconisée. Ces outils permettent d'aborder les tâches intimement liées de spécification, d'organisation, de validation et d'évaluation, en combinant une approche formelle à une approche empirique.

L'approche formelle retenue s'appuie sur des techniques de transformation

par raffinements successifs et décompositions, qui permettent non seulement de guider et de discipliner le raisonnement, mais aussi et surtout de simplifier voire d'éliminer les étapes délicates de validation de la spécification du comportement d'un système.

L'approche plus empirique s'appuie quant à elle sur des techniques de prototypage rapide fondées sur une modélisation adéquate de l'environnement et du flux d'activité, qui permet de simuler hors site le fonctionnement d'un système de commande , soit pour une mise au point fine de la spécification, soit pour dialoguer avec l'utilisateur afin de clarifier ses besoins, soit pour évaluer les performances et dimensionner l'atelier. Cette approche est un complément indispensable de la première approche pour aborder de façon réaliste une application dans toute sa complexité.

Les prolongements envisageables

Un certain nombre d'idées présentées dans ce mémoire ont été validées en construisant deux prototypes de produit. Le premier concerne les règles de décomposition par partitionnement, et le second la simulation conjointe de la partie commande et de la partie opérative. Cette phase d'expérimentation mérite d'être poursuivie avec des objectifs de mise en oeuvre plus précis, et des ambitions plus larges. Ces objectifs et ces ambitions doivent en particulier permettre la définition du noyau dur d'un système ouvert d'aide à la conception sur lequel il sera ensuite possible de greffer progressivement:

- différentes opérations de transformation
- différentes règles de construction
- différentes fonctions de validation
- et des outils de prototypage rapide

Ceci permettrait de comprendre encore plus les liens parfois très subtils qui existent entre les différentes tâches qui caractérisent le processus de conception, et de mieux étudier la complémentarité des différentes techniques de conception.

ANNEXE A

LES RESEAUX DE PETRI

GENERALITES :

DEFINITIONS

PROPRIETES

ABREVIATIONS

ET EXTENSIONS

PLAN

A.1 DEFINITIONS [Brams 83]

Un **réseau de Petri autonome** (RdP) R est défini par un quadruplet < P , T ; Pré , post > et par des règles de fonctionnement. Dans le quadruplet

- P désigne un ensemble de places,
- T un ensemble de transitions,
- et Pre et Post deux fonctions de P x T dans $\mathbb{N}$.

Si $\forall p \in P$, $\forall t \in T$: Pre (p , t) ≤ 1 et Post (p , t) ≤ 1 , on dit que R est **réseau ordinaire.**

Marquage d'un réseau :

Les places peuvent contenir zéro ou plusieurs marques. On appelle marquage d'un réseau, le vecteur d'entiers M désignant à un instant donné le nombre de marques contenues dans chaque place du réseau. Si p désigne une place, on note alors par M(p) le nombre de marques dans cette place. On appelle **RdP marqué** $<R;M_0)$, un RdP R auquel il est associé un marquage initial M_0 .

Règles de franchissement :

Le passage d'un marquage à un autre s'effectue de façon instantanée en franchissant une transition. Les fonctions Pre et Post définissent la précondition et la postcondition de ce franchissement: Pour un marquage M, une transition $t \in T$ est franchissable ssi

$$\forall p \in P \quad M(p) \geq Pre(p,t).$$

Le franchissement de t provoque alors le passage à un marquage M' tel que

$$\forall p \in P \quad M'(p) = M(p) - Pre(p,t) + Post(p,t).$$

On note alors

$M\ (t\rangle$ ou $M\ (t\rangle\ M'$.

Ce qui se lit que, t est franchissable à partir de M ($M\ (t\rangle$) et que le franchissement de t conduit au marquage M' ($M\ (t\rangle\ M'$). Cette notation s'étend de façon naturelle aux séquences de franchissement.

A un instant donné, une seule transition peut être franchie à la fois. Si pour un marquage accessible M plusieurs transitions sont franchissables, le choix de la transition à franchir s'effectue de façon aléatoire. On dit qu'un RdP marqué $<R.M_0>$ est **persistant** si pour tout marquage accessible M, le franchissement de tout couple de transitions franchissables (t1,t2) est commutatif. Dans un RdP marqué persistant, les transitions franchissables pour un marquage accessible M peuvent donc toujours être franchies dans n'importe quel ordre. Les réseaux non persistants modélisent par contre des choix exclusifs de franchissement caractérisés par des conflits structurels effectifs. On dit que deux transitions t_1 et t_2 sont en **conflit structurel** ssi elles ont en commun au moins une place en entrée. On dit alors que ces deux transitions sont en **conflit effectif** pour un marquage accessible M ssi

$$M\ (\ t_1\ \rangle\ ,\ M\ (\ t_2\ \rangle \text{ et } \exists\ p \text{ telle que } M(p) < Pre(p, t_1) + Pre(p, t_2).$$

On note par

$$L(R;M_0) = \{\ s \mid M_0(\ s\ \rangle\}$$

le langage que forme l'ensemble des séquences de franchissement possibles à partir du marquage initial M_0.

Par ailleurs, on note par

$$A(R;M0) = \{ M \mid \exists s \; M_0(s > M\}$$

l'ensemble des marquages accessibles à partir de M_0.

Représentation graphique :

Un RdP est habituellement représenté par un **graphe** bipartite, où les places sont représentées par des cercles, et les transitions par des barres ou des rectangles. Dans ce graphe, une place p est reliée à une transition t par un arc (p,t) partant de p vers t, de poids k = Pre(p,t) (on dit aussi de valuation k), ssi Pre(p,t) > 0. On dit alors que la place p est en entrée de la transition t et on note par

$^{\cdot}t = \{ p \in P \mid Pre(p,t) > 0 \}$ l'ensemble des places en entrée de la transition t

et par

$p^{\cdot} = \{ t \in T \mid Pre(p,t) > 0 \}$ l'ensemble des transitions en sortie de la place p.

De la même façon, une transition t est reliée à une place p par un arc (t,p) partant de t vers p, de poids k = Post(p,t), ssi Post(p,t) > 0. On dit que la place p est en sortie de la transition t et on note par

$t^{\cdot} = \{ p \in P \mid Post(p,t) > 0 \}$ l'ensemble des places en sortie de la transition t,

et par

$^{\cdot}p = \{ t \in T \mid Post(p,t) > 0 \}$ l'ensemble des transitions en entrée de la place p.

De façon plus intuitive, on peut alors dire qu'une transition $t \in T$ est franchissable pour un marquage accessible M, ssi chaque place p_e en entrée de cette transition contient un nombre de marques au moins égal à la valuation de

l'arc (p_e,t). Le tir d'une transition franchissable t conduit à un nouveau marquage en retranchant de chaque place p_e en entrée de t un nombre de marques égal la valuation de l'arc (p_e,t), et en ajoutant à chaque place p_s en sortie de t un nombre de marques égal à la valuation de l'arc (t,p_s).

Exemple :

Ce modèle de spécification permet essentiellement de modéliser le comportement d'un système de processus parallèles soumis à des contraintes de synchronisation. Dans cette modélisation, les marquages accessibles servent à représenter l'état du système, et les tirs de transitions, des occurrences d'événements. La figure A.1 (a) est un exemple simple de RdP modélisant le comportement d'un système constitué de deux processus P_1 et P_2 qui fonctionnent en utilisant une ressource commune en exclusion mutuelle. Dans cet exemple, la place p_{i1} n'est marquée que si le processus P_i est en attente de la ressource , et la place p_{i2} si le processus P_i est en cours d'exécution dans la section critique. La disponibilité de la ressource est modélisée par la place p_r. Le marquage de cette place signifie que la ressource est disponible. La figure A.1 (b) représente pour cet exemple le vecteur M_0 correspondant au marquage initial, et sous forme matricielle les fonctions Pre et Post du réseau, tandis que la figure A.1 (c) représente par un graphe appellé **graphe des marquages accessibles**, les évolutions possibles du réseau.

L'intérêt des RdP réside toutefois beaucoup plus dans les possibilités d'analyse qu'il est possible d'effectuer sur les propriétés du système modélisé.

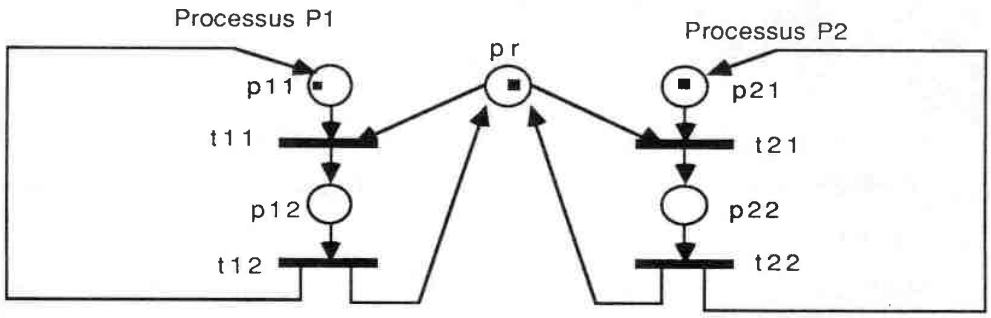


Figure A.1 (a)

p11	1
p21	1
P12	0
P22	0
pr	1

Marquage initial M_0

	t11	t21	t12	t22
p11	1	0	0	0
p21	0	1	0	0
p12	0	0	1	0
p22	0	0	0	1
pr	1	1	0	0

Matrice Pré

	t11	t21	t12	t22
p11	0	0	1	0
p21	0	0	0	1
p12	1	0	0	0
p22	0	1	0	0
pr	0	0	1	1

Matrice Post

Figure A.1 (b)

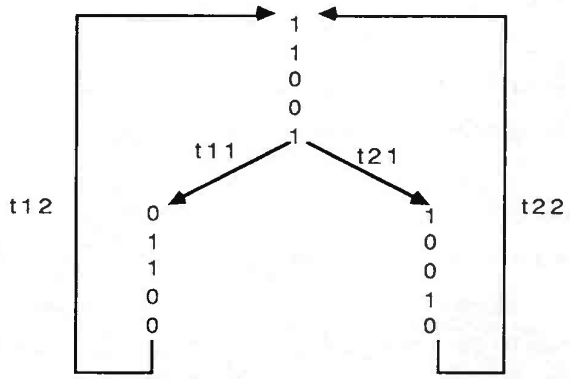


Figure A.1 (c)

A.2 ETUDE DES PROPRIETES FONDEE SUR L'ANALYSE DES MARQUAGES ACCESSIBLES [Brams 83] [Karp 69]

L'une des propriétés que l'on peut vérifier avec cette approche est le **bornage des places**. On vérifie que quelle que soit l'évolution d'un réseau marqué $<R;M_0>$, le marquage d'une place ou de l'ensemble des places ne peut jamais dépasser une borne donnée k, en énumérant à partir du marquage initial les différents marquages accessibles. Si à partir d'un marquage M on accède à un marquage M' tel que $\forall\ p \in P\ \ M'(p) \geq M(p)$, en se référant à la définition d'une transition franchissable, on voit bien que toute séquence de transitions franchissable à partir de M l'est également à partir de M'. Il s'ensuit que pour une place $p \in P$, si on a $M'(p) > M(p)$, en répétant la séquence de transitions qui conduit de M à M', on peut augmenter à l'infini le nombre de marques dans p. Karp et Miller [Karp 69] proposent un algorithme fini basé sur ce principe permettant de détecter toutes les places non bornées d'un réseau marqué.

Lorsque toutes les places d'un réseau marqué $<R;M_0>$ sont bornées, il est possible, par simple examen des différents marquages accessibles, de vérifier un certain nombre de propriétés dont:

- **Le bornage de l'ensemble des places à 1**: On dit alors que $<R;M_0>$ est un **réseau sauf**.

- **Les invariants de places** : On vérifie qu'une relation donnée sur le marquage des places d'un réseau est vraie quel que soit le marquage atteint.

- **L'absence de blocage** : On vérifie que pour tout marquage accessible M, il existe au moins une transition franchissable.

- **L'accessibilité** : On vérifie que pour un marquage donné M, il existe une séquence de franchissements de transitions conduisant à M'.

- **La vivacité** : On vérifie que le franchissement d'une transition ou de l'ensemble des transitions de $<R;M_0>$ est toujours possible quel que soit le marquage atteint.

- **L'existence d'un état d'accueil** : On vérifie que un marquage donné, appelé état d'accueil, sera toujours accessible quelle que soit l'évolution du réseau.

Toutefois, l'étude des propriétés d'un réseau par l'analyse des marquages accessibles présente quelques inconvénients majeurs: Cette analyse est impraticable si le nombre de marquages accessibles est excessivement grand. En plus, le fait qu'un réseau ne soit pas borné ne correspond pas nécessairement à une erreur de modélisation. Lorsqu' un réseau n'est pas borné, le nombre de marquages accessibles devient infini. On a alors recours à un graphe de couverture où chaque composant non borné d'un marquage accessible est remplacé par un symbole spécial ω désignant un nombre entier arbitrairement grand. L'utilisation de ce symbole dans un graphe introduit une perte d'information qui rend impossible la vérification systématique de certaines propriétés telles l'absence de blocage, l'accessibilité, la vivacité et l'existence d'un état d'accueil.

A.3 ETUDE DES PROPRIETES FONDEE SUR L'ALGEBRE LINEAIRE

[Brams 83] [Sifakis 79] [Berthomieu 80] [Genrich 81 b] [Jensen 81] [Memmi 83] [Silva 85]

Cette approche permet d'analyser le comportement décrit par un RdP en s'appuyant sur les propriétés structurelles du réseau plutôt que sur un examen exhaustif de ses marquages accessibles.

Soit pour un RdP R=<P,T;Pre,Post> C la matrice P x T appelée **matrice d'incidence** et calculée de la manière suivante:

C = Post - Pre.

En se référant aux règles de modification du marquage d'un réseau lors du tir d'une transition, on voit bien que chaque composante C(p,t) de cette matrice contient la valeur à ajouter à la place p lors de tout tir de la transition t. On notera en particulier que si une place p est à la fois en entrée et en sortie d'une transition t, et si la valuation de l'arc (p,t) est la même que celle de l'arc (t,p), alors C(p,t) = 0. On dit dans ce cas que R est un **réseau impur.** Si R est un réseau pur, la seule donnée de C suffit pour reconstituer sa structure : les places en entrée d'une transition $t \in T$ correspondent aux places $p \in P$ telles que $C(p,t) < 0$, et la valuation d'un arc (p,t) à -C(p,t); les places en sortie d'une transition $t \in T$ correspondent aux places $p \in P$ telles que $C(p,t) > 0$, et la valuation d'un arc (t,p) à C(p,t).

Soit s une séquence de franchissements de transitions. On appelle **vecteur caractéristique** de s, le vecteur $\underline{s}$ à coefficients entiers tel que la ième coordonnée contient le nombre de fois que la transition t_i apparaît dans la séquence s.

Exemple :

Si s = t1 t2 t3 t3 t1 t4 t1
alors $\underline{s}$ = 3 1 2 1

De ces définitions, il s'ensuit l'équation suivante qui fonde l'étude des propriétés d'un réseau à l'aide de l'algébre linéaire :

Si $M(s> M'$ alors $M' = M + C * \underline{s}$.

Si f est un vecteur d'entiers désignant une pondération des places du réseau, on peut écrire à partir de cette équation ce qui suit :

$f^t * M = f^t * M + f^t * C * \underline{s}$ où f^t désigne le transposé de f.

On appelle **p-semi-flot** tout vecteur d'entiers f solution de l'équation

$$f^t * C = 0.$$

On peut remarquer que pour un tel vecteur, on obtient la relation invariante suivante sur le marquage des places du réseau :

$$f^t * M' = f^t * M.$$

En utilisant les techniques de l'algèbre linéaire, on peut déterminer, pour un réseau donné, ses p-semi-flots, et prouver ainsi certaines de ses propriétés relatives au marquage des places (invariants, bornage,...).

Exemple :

Pour le réseau de la figure A.1, f = (0 0 1 1 1) est un p-semi-flot. Pour tout marquage accessible M à partir d'un marquage initial M_0 , la relation suivante doit donc être vérifiée:

$$M(p_{12}) + M(p_{22}) + M(p_r) = M_0(p_{12}) + M_0(p_{22}) + M_0(p_r).$$

M_0 étant égal à (1 1 0 0 1), on doit alors avoir pour tout marquage accessible M

$$M(p_{12}) + M(p_{22}) + M(p_r) = 0 + 0 + 1 = 1.$$

On en déduit ce qui suit:

$$M(p_{12}) = 1 \Rightarrow M(p_{22}) = M(p_r) = 0$$
$$M(p_{22}) = 1 \Rightarrow M(p_{12}) = M(p_r) = 0$$

Ce qui permet de conclure que l'accès à la ressource par les deux processus P_1 et

P_2 s'effectue toujours en exclusion mutuelle.

De façon similaire, à partir de l'équation fondamentale, on peut se convaincre que s'il existe une séquence de franchissements de transitions permettant de passer d'un marquage M à un marquage M', alors cette séquence s est nécessairement solution de l'équation

$$C * \underline{s} = M' - M.$$

Par ailleurs, si s est une séquence qui peut être franchie de façon répétitive, alors elle doit aussi être nécessairement solution de l'équation

$$C * \underline{s} \geq 0.$$

Ces deux équations fournissent des conditions nécessaires d'accessibilité et de vivacité dans un RdP marqué. Ces conditions ne sont toutefois pas suffisantes; étant donné un RdP, les solutions de ces équations peuvent en effet ne pas correspondre à des séquences franchissables. Le réseau de la figure A.2 est un exemple cité dans [Peterson 81] pour illustrer ce phénomène, qui constitue la limitation majeure de l'étude des propriétés des RdP par l'algèbre linéaire. Si l'on considère M' = (0, 0, 0, 1) et M = (1, 0, 0, 0), s = (1,1) est solution de l'équation $C * \underline{s} = M' - M$. Pourtant, ni la séquence (t1, t2) ni la séquence (t2, t1) ne permet d'accéder au marquage M' à partir du marquage M.

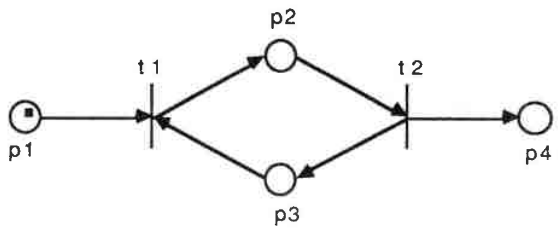


Figure A.2

A.4 ETUDE DES PROPRIETES FONDEE SUR LA THEORIE DES GRAPHES [Brams 83] [Commoner 72] [Memmi 83]

Comme le montrent les deux sections précédentes, la vérification des propriétés liées à la vivacité d'un réseau ne sont pas toujours décidables. Cependant, des études fondées sur la théorie des graphes ont permis de dégager les résultats que nous présentons pour des classes particulières de réseaux.

On dit qu'un sous-ensemble V non vide de places constitue un verrou ssi

$\forall p \in V, \forall t \in {}^{\cdot}p : \exists p' \in V$ telle que $t \in p'^{\cdot}$

(*Toute transition en entrée d'une place de V est aussi en sortie d'une place de V*). ⊗

Si pour un marquage accessible l'ensemble des transitions en sortie des places d'un verrou sont infranchissables, elles resteront donc infranchissables quelle que soit l'évolution du réseau.

Par ailleurs, on dit qu'une place p est déficiente pour un marquage accessible M ssi

(1) $p^{\cdot} \neq \varnothing$

(2) $M(p) < \mathrm{Min}_{t \in p^{\cdot}} \mathrm{Pre}(p,t)$. ⊗

Par extension, un sous-ensemble de places Q est déficient pour un marquage M ssi toute place de Q est déficiente pour ce marquage.

On dit alors qu'un sous-ensemble A non vide de places constitue une trappe ssi

(1) $\forall p \in A, \forall t \in p^{\cdot} : \exists p' \in A$ telle que $t \in {}^{\cdot}p'$

(*Toute transition en sortie d'une place de A est aussi en entrée d'une place*

de A)

(2) $\forall p \in A, \forall t \in p^{\cdot} : \exists p' \in A \cap t^{\cdot}$ telle que
soit $p'^{\cdot} = \varnothing$

soit $Post(p',t) \geq Min_{t' \in p'^{\cdot}} Pre(p', t')$

(*Pour toute transition* t *en sortie d'une place de* A, *il existe une place* p' *en sortie de* t *incluse dans* A *qui n'est jamais déficiente après tout tir de* t). ⊗

En d'autres termes, une trappe définit un sous-ensemble de places qui ne perdent jamais toutes leurs marques quelle que soit l'évolution du réseau.

Le thérome de Commoner [Commoner 72] donne pour des classes particulières de réseaux une condition de vivacité qui consiste à vérifier que tout verrou contient une trappe non déficiente au marquage initial. Les trois classes que nous évoquons ici nous sont utiles dans le chapitre 6. Pour les deux premières classes (les graphes d'événements et les machines à états), la condition de Commoner constitue une condition nécessaire et suffisante. Pour la troisième classe (les graphes simples), elle constitue une condition suffisante.

Les graphes d'événements :

Un réseau R = < P , T ; Pre , Post > est un graphe d'événements ssi

$\forall p \in P : |p^{\cdot}| = |{}^{\cdot}p| = 1$
(*Chaque place a exactement une transition en entrée, et une transition en sortie*). ⊗

Pour ce type de réseau, la vérification de la condition de vivacité se ramène à s'assurer que chaque circuit élémentaire contient au moins une marque.

Les machines à états :

Un réseau R = < P , T ; Pre , Post > est une machine à états ssi

$\forall t \in T : |t^{\cdot}| = |^{\cdot}t| = 1$

(*Chaque transition a exactement une place en entrée, et une place en sortie*) .⊗

La vérification de la condition de vivacité sur une machine à états s'effectue également de façon très simple. Elle se ramène à s'assurer que le réseau est fortement connexe et contient au moins une marque.

Les graphes simples :

Un réseau R = <P,T;Pre,Post> est considéré comme un graphe simple ssi

$\forall (p,p') \in P^2$, si $p^{\cdot} \cap p'^{\cdot} \neq \varnothing$ alors soit $p^{\cdot} \subset p'^{\cdot}$ soit $p'^{\cdot} \subset p^{\cdot}$ soit $p^{\cdot} = p'^{\cdot}$. ⊗

On peut noter que les graphes simples incluent les graphes d'événements et les machines à états.

A.5 NOTIONS D'ABSTRACTION ET D'EQUIVALENCE DE COMPORTEMENT [André 81]

L'étiquetage permet de restraindre l'observation du comportement décrit par un RdP, à un sous-ensemble d'événements considérés comme essentiels, ou de confondre les tirs de transitions modélisant l'occurrence d'un même événement.

De façon générale, on définit un **RdP étiqueté** R_E par un quadruplet $< R;M_0;E;e>$ où

- R = <P,T;Pre,Post> désigne un RdP
- M_0 le marquage initial
- E un vocabulaire

- et e : $T \mapsto E + \{\lambda\}$ une application qui associe à chaque transition $t \in T$ une étiquette de E ou l'élément neutre λ

et on note par e : $T^* \mapsto E^*$ l'extension naturelle de e qui associe à chaque mot de T^* un mot de E^* définie comme suit:

Soit $s \in T^*$. Si $s = \lambda$ alors $e(s) = \lambda$

sinon $s = s't$ et $e(s) = e(s')\, e(t)$. $\otimes$

L'ensemble des mots que l'on obtient par concaténation des étiquettes associées aux transitions des différentes séquences de franchissement possibles dans $< R;M_0 >$ forme un langage

$$L_E(R_E) = \{ e(s) \mid M_0 (s > \}$$

qui définit l'ensemble des séquences d'événements observables dans le système modélisé. Chaque élément $e(s) \in L_E(R_E)$ est alors appelé image de s par e.

Si E = T et si e désigne l'application identité, on remarquera que le langage que l'on obtient correspond à $L(R;M_0)$.

La **projection** sur un ensemble de transitions distinguées $T_d \subset T$ définit une application d'étiquetage particulière telle que

$$\forall t \in T : e(t) = t \text{ si } t \in T_d \text{ et } \lambda \text{ si } t \in T - T_d.$$

Elle permet de restraindre l'observation du comportement du réseau $< R ; M_0 >$ aux transitions contenues dans T_d. On note alors par $Proj(T_d,s)$ la projection de $s \in L(R;M_0)$ sur T_d qui s'obtient en supprimant de s les symboles n'appartenant pas à T_d, et par $Proj(T_d, L(R;M_0))$ la projection de $L(R;M_0)$ sur T_d.

Différents critères sont proposés dans la littérature pour décider de l'équivalence de deux RdP étiquetés [Sifakis 79] [Berthelot 78] [André 81].

Les critères d'équivalence fondés sur la notion d'abstraction proposés dans [Andre 81] permettent de garantir que si l'on contraint deux systèmes considérés comme équivalents à évoluer en réalisant dans le même ordre les mêmes événements, à chaque instant, ces deux système offriront chaque fois les mêmes possibilités d'évolution future. De ce fait, ces critères sont équivalents aux critères d'observabilité proposés dans [Milner 80]. Ils s'énoncent plus simplement lorsque l'étiquetage des réseaux définit une projection sur un sous-ensemble de transitions.

Soit $< R;M0 >$ avec $R = < P,T;Pre,Post >$ un RdP marqué, et $T_d \subset T$ un sous-ensemble de transitions distinguées. Si pour $M,M' \in A(R;M_0)$ il existe $s1,s2 \in (T - T_d)^* + \{\lambda\}$ et $t \in T_d$ tels que $M (s1\ t\ s2 > M'$, nous noterons par convention

$M (. t . > M'.$

On dit que $< R;M_0 >$ admet une abstraction sur T_d ssi

$\forall\ s1,s2 \in L(R;M_0), \forall\ t \in T_d, \forall\ M_1,M_2 \in A(R;M_0)$:

$M_0(s1 > M_1\ ,\ M_0(s2 > M_2\ ,\ Proj(T_d,s1)=Proj(T_d,s2)$

$\Rightarrow\ M_1(. t .> \Leftrightarrow\ M_2(. t .>. \otimes$

Cette condition permet de définir l'équivalence de deux réseaux de la façon suivante:

Soient $< R;M_0 >$ avec $R=< P,T;Post,Pre >$ et $< R';M_0' >$ avec $R' = < P',T';Pre',Post' >$ deux RdP tels que $T_d \subset T$ et $T_d \subset T'$. On dit que $< R;M_0 >$ et $< R';M_0' >$ décrivent le même comportement sur T_d ssi

(1) $Proj(T_d, L(R;M_0)) = Proj(T_d, L(R',M_0'))$

(2) $\forall s \in L(R;M_0), \forall M \in A(R;M_0)$

$\forall s' \in L(R';M_0'), \forall M' \in A(R';M_0')$

$\forall t \in T_d$:

$M_0(\ s >M, M_0'(\ s' >M', Proj(T_d,s) = Proj(T_d, s') \Rightarrow M(.\ t\ .> \Leftrightarrow M'(.\ t\ .>\ .\ \otimes$

A.6 LES REGLES DE REDUCTION [Brams 83] [Berthélot 85 a]

Les règles de réduction permettent de réduire ou d'augmenter la taille d'un réseau sans modifier certaines de ses propriétés (caractère borné et conservatif, propriétés liées à la vivacité, et abstraction). Lorsque de par sa structure un réseau est complètement réductible, ces règles constituent de ce fait un excellent moyen d'analyse, sinon, elles peuvent servir à réduire sa taille pour le rendre plus facilement analysable. Dans ce qui suit, nous présentons quelques règles qui nous sont utiles pour fonder nos règles de construction et de transformation.

N.B. : $\underline{M_0}$ désigne un ensemble de marquages initiaux.

Définition DA.1 : Place substituable (ou post-agglomération) :
[Brams 83] , [Berthelot 85 a]

Soit $N = < R ; \underline{M_0} >$ avec $R = < P, T ; Pre, Post >$ un réseau marqué. Une place p_s de N est substituable ssi il existe un entier m positif et non nul et deux sous-ensembles de transitions H et F non vides qui satisfont les conditions suivantes :

(1) $\forall f \in F$: $Pre(p, f) =$ m si $p = p_s$ 0 sinon. (*La seule entrée de* f *est* p_s).

(2) $\forall f \in F$: $Post(p_s, f) = 0$ (p_s *n'est pas une place sortie de* f)

(3) $\forall f \in F$: $\exists\ p \in P$ t. q. $Post(p, f) > 0$

(*Une transition au moins de* F *à une sortie*)

(4) $\forall h \in H$: Pre $(p_S, h) = 0$ (p_S *n'est pas en entrée de* h)

$\forall h \in H: \exists K_h \in \mathbb{N}, K_h \neq 0$ t.q. Post $(p_S, h) = m * K_h$

(*Le nombre de marques mises par* h *dans* p_S *est multiple de* m)

(5) $\forall t \notin H + F$: Pre (p_S, t) = Post $(p_S, t) = 0$

(*Toute transition qui n'appartient ni à* H, *ni à* F, *n'est pas reliée à* p_S).

La substitution de la place p_S donne le réseau réduit N' = < R' ; $\underline{M_0}$'> avec R' = < P', T' ; Pre', Post ' > où

(6) $P' = P - \{ p_S \}$

(7) $T' = (\Sigma_{h \in H} R1 (h)) + T - (H + F)$

où R1 (h) est l'ensemble de transitions obtenu en substituant dans Post (. , h) toutes les combinaisons possibles des Post (. , f), (f ∈ F) à Post (p_S, h).

(8) $\forall t \in T - (H + F)$: Pre'(., t) = Pre(., t), Post'(., t) = Post(., t)

(9) $\underline{M_0}' = \{ M_0 - Q_M * v + \Sigma_{f \in F} nf * Post(., f) \mid M_0 \in \underline{M_0}$ et $\Sigma_{f \in F} nf = Q_M$,

v(p) = m si p = p_S 0 sinon }.

où Q_M est la partie entière de M (p)/m . ⊗

Définition DA.2 : Place implicite [Brams 83] [Berthelot 85 a]

Une place implicite est telle que, quel que soit le marquage atteint, elle n'est jamais un obstacle aux franchissements des transitions dont elle est une entrée. Plus formellement, une place p_I d'un RdP N = < R; $\underline{M_0}$ > avec R = < P, T, Pre, Post > est une place implicite ssi il existe un ensemble de référence I, ne contenant pas p_I et éventuellement vide, une pondération f de I + $\{p_I\}$ tels que :

(1) $f : I+\{p_I\} \mapsto \mathbb{N}^+$ *(Toutes les places concernées ont une pondération positive)*

(2) $\forall M_0 \in \underline{M_0} : \exists b_M \in \mathbb{N}$ t.q. $f(p_I) * M0(pI) - \Sigma_{q\in I}(f(q) * M_0(q)) = b_M$

(Pour tout marquage initial M_0, *la différence entre le marquage pondéré de* p *et la somme des marquages pondérés des places de* I *est positive ou nulle)*

(3) $\forall t \in T : f(p_I) * Pre(p_I, t) - \Sigma_{q\in I}(f(q) * Pre(q, t)) \leq Min_{M0\in \underline{M0}} \{b_M\}$

(Pour toute transition t, la différence entre le marquage pondéré de p_I *nécessaire au franchissement de* t *et la somme des marquages pondérés nécessaires à ce franchissement n'est pas supérieure à celle de chacun des marquages initiaux).*

(4) $\forall t \in T : \exists c_t \in \mathbb{N}$ t.p. $f(p_I) * C(p,t) - \Sigma_{q\in I}(f(q) * c(q, t)) = c_t$

(Pour le franchissement de toute transition t, *la différence entre l'accroissement (positif ou négatif) du marquage pondéré de* p_I *et l'accroissement de la somme des marquages pondérés des places de* I *n'est pas négative).*

La simplification du réseau s'effectue alors en supprimant tous les arcs entrants et sortants de la place p_I et en rajoutant un arc valué c_t de t vers p_I pour toute transition t. Si tous les c_t sont nuls, la place peut être supprimée.

Définition DA.3 : Transitions identiques [Brams 83]

Soit N = < R ; $\underline{M_0}$ > avec R = < P, T ; Pre, Post > un RdP marqué. Deux transitions t1 et t2 sont identiques ssi

(1) Pre (., t1) = Pre (., t2) et Post (., t1) = Post (., t2)

La suppression de la transition identique t2 donne le réseau réduit N' = < R' ; $\underline{M_0}$' > avec R' = < P', T' ; Pre', Post' > où

(2) P' = P

(3) T' = T - { t2 }

(4) Pre' = Pre

(5) Post' = Post

(6) $\underline{M_0}' = \underline{M_0} \cdot \otimes$

Définition DA.4 : Transition neutre [Brams 83]

Une transition t_n d'un réseau marqué N = < R ; $\underline{M_0}$ > avec R = < P,T; Pre,Post > est une transition neutre ssi t_n a les mêmes places en entrée et en sortie, i.e.:

(1) $\forall\ p \in P$: Pre (p, t_n) = Post (p, t_n)

La transition t_n est alors supprimable ssi il existe une transition t' différente de t_n telle que:

(2) Post (., t') $\geq$ Pre (., t)

La suppression de t_n donne le réseau réduit N' = < R' ; $\underline{M_0}$' > avec R' = < P', T' ; Pre', Post' > défini comme suit :

(3) P' = P

(4) T' = T - { t_n }

(5) Pre' = Pre

(6) Post' = Post

(7) $\underline{M_0}' = \underline{M_0} \cdot \otimes$

Définition DA.5 : Pré-agglomération [Berthelot 85 a]

Soit N = < R ; $\underline{M_0}$ > avec R = < P, T ; Pre, Post > un réseau marqué. Un sous

ensemble de transitions F est pré-agglomérable à une transition h ssi il existe une place p_s telle que les conditions suivantes sont satisfaites:

(1) $\forall$ p $\in$ P : Post (p, h) = 1 si p = p_s et 0 sinon (*La seule sortie de* h *est* p_s)

(2) Pre (p_s, h) = 0 (p_s *n'est pas entrée de* h)

(3) Pre(., h) > 0 (h *a au moins une entrée*)

(4) $\forall$ t $\in$ F : Pre(p_s, t) = 1 (*Toute transition de* F *prend une marque dans* p_s)

(5) $\forall$ t $\in$ F : Post(p_s, t) = 0 (*Aucune transition de* F *ne met de marque dans* p_s)

(6) $\forall$ t $\in$ T - ({h} + F) : Pre(p_s, t) = Post(p_s, t) = 0

(*Aucune autre transition n'est connectée à* p_s)

(7) $\forall$ $M_0 \in \underline{M_0}$: M_0 (p_s) = 0 (p_s *est initialement vide*)

(8) $\forall$ p $\in$ P , $\forall$ t $\neq$ h : Pre(p, h) $\neq$ 0 $\Rightarrow$ Pre(p, t) = 0

(*h ne partage pas ses entrées*)

La pré-agglomération de F à h donne le réseau réduit N' = <R' ; $\underline{M_0}'$ > en substituant les entrées de chaque transition t $\in$ F par les entrées de h.$\otimes$

Comme on peut le constater, la pré-agglomération est une règle de réduction voisine de la règle de réduction par substitution de place. La place p_s doit ici être la seule sortie de h et non plus la seule entrée de F. Par ailleurs, h ne doit pas partager ses entrées.

A.7 LES ABREVIATIONS DES RdP

Les abréviations de RdP ont été introduites pour faciliter la construction et la lecture de réseaux de très grande taille, tout en conservant la puissance d'expression et d'analyse du modèle initial.

Les **RdP à prédicats** [Genrich 81 a] **et les RdP colorés** [Jensen 81] [Alla 87] constituent les formes les plus parachevées des abréviations de RdP.

Les RdP à prédicats :

Dans une place p d'un RdP à prédicats, on associe à une marque, caractérisée par un n-uplet de paramètres, un n-uplet de valeurs qui permet de le distinguer des autres marques de la place. On précise alors les types de marques qu'une transition t doit consommer ou produire dans une place p

- soit de façon explicite en désignant sur l'arc (p,t) ou (t,p) les n-uplets de valeurs que doivent avoir les paramètres des marques

- soit de façon implicite en associant aux arcs des n-uplets de variables, et aux transitions des prédicats sur ces variables décrivant la relation qui doit exister entre les valeurs de paramètres des marques qu'on consomme ou produit.

La modélisation par RdP d'une voie de communication présentée dans la figure A.3 est un exemple souvent cité pour illustrer le pouvoir d'abréviation des RdP [Brams 83] [Alla 84]. La voie de communication modélisée fonctionne comme une file FIFO (First-In, First-Out) de capacité 3, et permet la transmission de deux types de messages, m et m'. La figure A.3 (a) décrit le comportement de la voie par un RdP généralisé, et la figure A.3 (b) par un RdP à prédicats.

Dans la figure A.3 (a), l'émetteur transmet les messages de type m en franchissant la transition t_e, et les messages de type m' en franchissant la transition t_e', tandis que le récepteur reçoit un message de type m en franchissant la transition t_s, et un message de type m' en franchissant la transition t_s'. Le marquage d'une place v_i signifie que la case de rang i est vide, et le marquage d'une place c_i (respectivement d'une place c_i') que la case de rang i contient un message de type m (respectivement de type m').

Dans la figure A.3 (b), les places d'entrée p_e et p_e' ont été regroupées en une seule place pp_e, les places de sortie p_s et p_s' en une seule place pp_s, les

places v_i en une seule place v, et les places c_i et c_i' en une seule place c. Les marques dans les places pp_e et pp_s sont caractérisées par un paramètre qui désigne le type de message à transmettre, les marques dans la place v par un paramètre qui indique le rang d'une case vide, et les marques dans la place c par deux paramètres, l'un désignant le rang d'une case contenant un message, et l'autre désignant le type de ce message.

Les transitions t_e et t_e' ont été regroupées en une seule transition tt_e. Pour être franchie, cette transition doit consommer une marque dans v dont le paramètre indique que la première case du tampon est vide, et une marque dans pp_e dont le paramètre indique le type de message à transmettre. Son franchissement produit alors dans c une marque dont le premier paramètre indique que la première case du tampon contient un message. Le type de ce message est désigné par le second paramètre.

De la même façon, les transitions t_s et t_s' ont été regroupées en une seule transition tt_s. Pour être franchie, cette transition doit consommer une marque dans c dont le premier paramètre indique que la troisième case du tampon contient un message. Le type de ce message est indiqué par le second paramètre. Le franchissement de tt_s produit alors dans v une marque dont le paramètre indique que la troisième case du tampon a été libérée, et une marque dans pp_s dont le paramètre indique le type du message reçu.

Les transitions t_1, t_1', t_2, t_2', t_3, et t_3' ont également été regroupées en une seule transition désignée par tt. Pour être franchie, cette transition doit consommer dans c une marque dont le premier paramètre indique que la case de rang i est non vide. Le type du message contenu dans cette case est indiqué par le deuxième paramètre. La transition t doit aussi consommer une marque dans la case v dont le paramètre indique que le la case de rang i+1 est vide. Le franchissement de t produit alors dans v une marque dont le paramètre indique

que la case de rang i est devenue inoccupée, et dans c une marque dont le premier paramètre indique que la case de rang i+1 est non vide. Le type du message contenu dans cette case est indiqué par le second paramètre.

Les RdP colorés :

Les RdP colorés offrent une autre variante pour l'abréviation des RdP. Dans cette variante, les marques sont également différenciées par des n-uplets de valeurs. Chaque n-uplet de valeurs représente une couleur. On précise les types de marques qu'une transition t doit consommer ou produire en associant à cette transition un ensemble de couleurs, et à chacun de ses arcs une fonction. Les couleurs associées aux transitions permettent de distinguer différents types de tirs de transition, et les fonctions associées aux arcs de déterminer pour chacun de ces tirs, les types de marques concernés. Le domaine d'une fonction associée à un arc est donc l'ensemble des couleurs de la transition correspondante, et son codomaine l'ensemble des couleurs que doivent avoir les marques consommées ou produites dans la place correspondante.

La figure A.3 (c) correspond à la modélisation de la file FIFO à l'aide d'un RdP coloré proposée dans [Alla 84]. Comme on peut le constater, il conduit au même pliage que dans la figure A.3 (b).

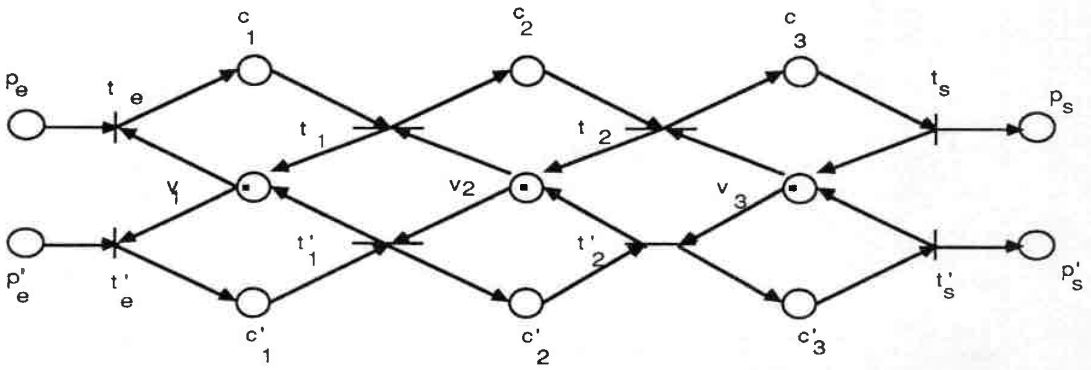


Figure A.3 (a)

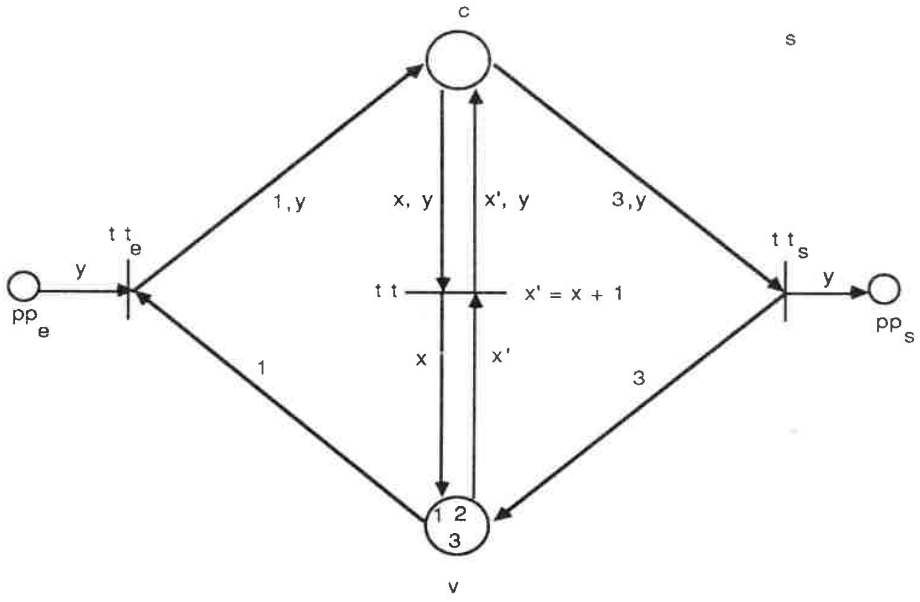
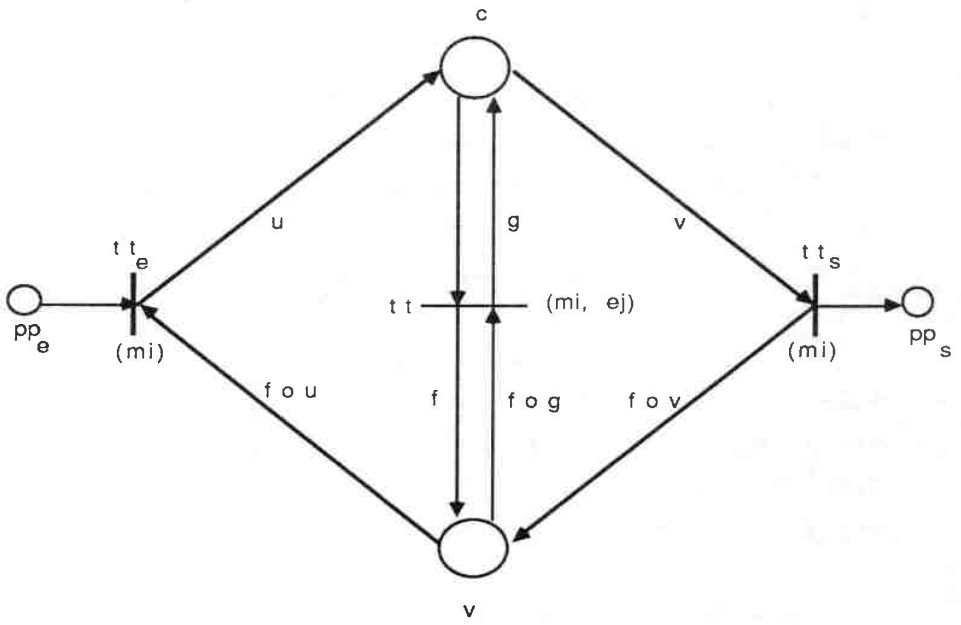


Figure A.3 (b)



ENSEMBLE DES COULEURS

{ mi, ej, <mi, ej> } ∀ i ∈ (1,2) j ∈ (1 à 3)

MARQUAGE INITIAL

$$M(v) = \sum_{j=1}^{3} e_j$$

DEFINITION DES FONCTIONS

f(mi,ej) = ej
u(mi) = mi, e1
g(mi,ej) = mi, ej+1
v(mi) = mi, e3

Figure A.3 (c)

A.8 LES EXTENSIONS DES RdP

Les extensions des RdP sont en général proposées pour augmenter le pouvoir d'expression des RdP généralisés. Elles conduisent à une modification des règles de fonctionnement du modèle de base, ce qui remet en cause parfois

de façon profonde la décidabilité des propriétés du réseau.

Parmi les extensions les plus significatives, citons les **RdP à priorité.** Dans un RdP à priorité, on définit un ordre entre les différentes transitions. Pour un marquage accessible M, si plusieurs transitions sont franchissables, on choisit de franchir une transition t telle que, selon l'ordre défini, la transition t est plus petite ou égale à tout autre transition franchissable.

Les RdP non autonomes constituent des classes d'extensions de réseaux non moins importantes, qui permettent de lier l'évolution d'un réseau à celui d'un environnement. Les **RdP temporisés** [Chrétienne 84] [Chrétienne 85], les **RdP stochastiques** [Florin 85] et les **RdP interprétés** [Moalla 85] sont les plus caractéristiques de ces classes.

Dans les RdP temporisés et dans les RdP stochastiques, on associe aux places ou aux transitions des temps qui sont fixes dans le premier cas et aléatoires dans le second. Le temps associé à une place désigne la durée pendant laquelle toute marque qui arrive dans cette place doit être considérée comme indisponible, tandis que le temps associé à une transition désigne la durée de franchissement de cette transition. Selon que les actions sont associées aux places ou aux transitions, on peut ainsi prendre en compte dans une modélisation, leur durée d'exécution , et réaliser de façon statique des analyses sur les performances du système modélisé.

Les RdP interprétés qui constituent le modèle sur lequel nous nous appuyons sont présentés de façon plus détaillée dans la section 2.4.

ANNEXE B

DEMONSTRATIONS DU CHAPITRE 6

PLAN

B.1 DEMONSTRATION DE LA PROPRIETE P6.2

a) Cas de la primitive SEQ

Si R = < P, T ; Pre, Port > est une machine à états, alors par définition :

$$\forall\, t \in T : (\Sigma_{p\in P} \text{Pre}(p, t) = 1) \text{ et } (\Sigma_{p\in P} \text{Post}(p, t) = 1)$$

En se référant à la définition D6.2 qui précise la règle de transformation de la primitive SEQ, à partir des points (2), (3) et (8), on en déduit que :

$$\forall\, t \in T' - \{t_h, t_f\} : \Sigma_{p\in P'} \text{Pre}'(p, t) = \Sigma_{p\in P} \text{Pre}(p, t) = 1$$

$$\Sigma_{p\in P'} \text{Post}'(p, t) = \Sigma_{p\in P} \text{Post}(p, t) = 1.$$

Par ailleurs :

$\Sigma_{p\in P'} \text{Pre}'(p, t_h) = \Sigma_{p\in P} \text{Pre}(p, t_x) = 1$ (c.f. D 6.2 (4))

$\Sigma_{p\in P'} \text{Post}'(p, t_h) = 1$ (c.f. D 6.2 (5))

$\Sigma_{p\in P'} \text{Pre}'(p, t_f) = 1$ (D 6.2 (6))

$\Sigma_{p\in P'} \text{Post}'(p, t_f) = \Sigma_{p\in P} \text{Post}(p, t_x) = 1$ (c.f. D 6.2 (7))

Ce qui revient à dire que R' = < P', T', Pre', Post' > est une machine à états.⊗

b) Cas de la primitive ALT

La démonstration de cette propriété s'effectue en constatant tout simplement que les nouvelles transitions dans T', t_{x1} et t_{x2} , ont les mêmes places en entrée et en sortie que t_x (c.f. D 6.3 (3) (4) et (5)).

Si $\Sigma_{p\in P} \text{Pre}(p, t_x) = \Sigma_{p\in P} \text{Post}(p, t_x) = 1$

Alors $\Sigma_{p\in P'} \text{Pre}'(p, t_{x1}) = \Sigma_{p\in P'} \text{Post}'(p, t_{x1}) = 1$

$$\Sigma_{p\in P'} \text{ Pre' } (p, t_{x2}) = \Sigma_{p\in P'} \text{ Post' } (p, t_{x2}) = 1.\otimes$$

c) Cas de la primitive REP

Si R = < P, T ; Pre, Post > est une machine à états, alors

$\forall\, t \in T$: $\Sigma_{p\in P}$ Pre (p, t) = 1 et $\Sigma_{p\in P}$ Post (p, t) = 1.

La définition D 6.4 permet alors de déduire ce qui suit :

- $\forall\, t \in T' - \{ t_h, t_f, t_n \}$: $\Sigma_{p\in P'}$ Pre' (p, t) = $\Sigma_{p\in P}$ Pre (p, t) = 1 (c.f. D6.4 (8))

  $\Sigma_{p\in P'}$ Post (p, t) = $\Sigma_{p\in P}$ Post (p, t) = 1 (c.f. D6.4 (8))

- $\Sigma_{p\in P'}$ Pre' (p, t_h) = $\Sigma_{p\in P}$ Pre (p, t_x) = 1 (c.f. D6.4 (4))

- $\Sigma_{p\in P'}$ Post' (p, t_h) = 1 (c.f. D6.4 (5))

- $\Sigma_{p\in P'}$ Pre (p, t_f) = 1 (c.f. D6.4 (6))

- $\Sigma_{p\in P'}$ Post' (p, t_f) = $\Sigma_{p\in P}$ Post (p, t_n) = 1 (c.f. D 6.4 (7))

- $\Sigma_{p\in P'}$ Pre' (p, t_n) = 1 (c.f. D6.4 (9))

- $\Sigma_{p\in P'}$ Post (p, t_n) = 1 (D6.4 (10))

ce qui permet de conclure que R' = < P', T', Pre', Post' > est une machine à états.⊗

B.2 DEMONSTRATION DE LA PROPRIETE P6.3

a) Cas de la primitive SEQ

La propriété P6.3 se démontre pour la primitive SEQ en constatant que les transformations de cette primitive est un cas particulier des transformations inverses de la règle plus générale de réduction par substitution de place, qui préserve plusieurs propriétés dont la vivacité et le caractère conservatif.

De la définition DA.1 (c.f. section A.6), si l'on considère le réseau étiqueté R_E' transformé par la primitive SEQ, en posant m = 1, H = { t_h } et F = { t_f }, p_s est une place substituable. En effet, on peut noter que les conditions de la définition DA.1 sont toutes satisfaites :

(1) Pre' (p, t_f) = 1 si $p = p_s$ 0 sinon (c.f. D6.2 (6))

(2) Post' (p_s, t_f) = 0 (c.f. D6.2 (7))

(3) $\exists$ p $\in$ P' t.q. Post' (p, t_f) > 0 (c.f. D6.2 (1) et (7))

(4) Pre' (p_s, t_h) = 0 (D 6.2 (4))

Post' (p_s, t_h) = 1 * m = 1 (c.f. D 6.2 (5))

(5) $\forall$ t $\notin$ { t_h, t_f } : Pre' (ps, t) = Post' (p_s, t) = 0 (c. f. D6.2 (8))

La réduction du réseau R' par substitution de la place p_s donne le réseau R" = < P", T" ; Pre", Post" > défini comme suit :

(6) P" = P' - { p_s } = P (c.f. D6.2 (2))

(7) T" = { t_x } + T' - ({ t_h } + { t_f }) = T (c.f. D6.2 (3))

avec { t_x } = R1 (t_h)

donc Pre" (p, t_x) = Pre' (p, t_h) = Pre (p, t_x) $\forall$ p $\in$ P" (c.f. D6.2 (4))

Post" (p, t_x) = Post' (p, t_f) = Post (p, t_x) $\forall$ p $\in$ P" (c.f. D6.2 (7))

(8) $\forall$ t $\in$ T' - { t_h + t_f } : Pre" (., t) = Pre ' (., t) = Pre (., t) (c.f. D6.2 (8))

Post" (.,t) = Post' (., t) = Post (., t) (c.f. D6.2 (8))

(9) $\forall$ p $\in$ P" : M_0" (p) = M_0' (p) = M_0 (p)

Comme on peut le constater, les réseaux R" et R' sont tout à fait identiques. On en déduit que les transformations de la primitive SEQ correspondent à un cas particulier des transformations inverses de la règle de réduction par substitution de place, et que de ce fait, elles préservent les mêmes propriétés que cette règle, dont la vivacité et le caractère conservatif [Brams 83].⊗

b) Cas de la primitive ALT

La démonstration de la propriété P6.3 pour la primitive ALT découle du fait que cette primitive induit des transformations qui correspondent aux transformations inverses de la régle de réduction par suppression de transitions identiques (c.f. définition DA.3 section A.6).

De la définition D6.3, si l'on considère le RdP R' obtenu par application de la primitive ALT, on voit bien que les transitions t_{x1} et t_{x2} qui ont mêmes places en entrée et mêmes places en sortie sont des transitions identiques (c.f. D6.3 (4) et (5)). La suppression de t_{x2} donne le réseau R" = < P", T" ; Pre", Post" > défini comme suit :

(2) $P'' = P' = P$ (c.f. D6.3 (2))

(3) $T'' = T' - \{ t_{x2} \} = T - \{ t_x \} + \{ t_{x1} \}$ (c.f. D6.3 (3))

(4) $Pre'' = Pre' \Rightarrow$ $Pre''(., t_{x1}) = Pre(., t_x)$ (c.f. D6.3 (4))

$\forall t \in T'' - \{t_{x1}\} : Pre''(., t) = Pre(., t)$ (c.f. D6.3 (6))

(5) $Post'' = Post' \Rightarrow$ $Post''(., t_{x1}) = Post(., t_x)$ (c.f. D6.3 (5))

$\forall t \in T'' - \{ t_{x1} \} : Post''(., t) = Post(., t)$ (D6.3 (6))

(6) $M_O'' = M_0' = M_0$ (D.6.3 (7)).

On voit que la différence entre R" et R est que R" contient la transition t_{x1} à la place de la transition t_x. Ces deux transitions ayant mêmes entrées et mêmes sorties, on en déduit qu'il s'agit du même réseau. De ce fait, les transformations induites par la primitive ALT correspondent bien aux transformations inverses de la règle de réduction par suppression de transitions identiques. Comme cette régle de réduction préserve la vivacité et le caractère conservatif [Brams 83], nous en déduisons que la primitive ALT préserve les mêmes propriétés . ⊗

c) Cas de la primitive REP

La propriété P6.3 se démontre pour la primitive REP en constatant que les transformations de cette primitive correspondent aux transformations inverses de deux règles de réduction préservant la vivacité et le caractère conservatif : la règle de réduction par substitution de place (cf. définition DA.1 section A.6) et la règle de suppression de transition neutre (c.f. définition DA.4 section A.6).

En se référant à la définition D6.4, dans un réseau R_E' qu'on obtient par application de la primitive REP, la transition t_n est une transition neutre supprimable :

(1) $Pre(p, t_n) = Post(p, t_n) = 1$ si $p = p_s$ et 0 sinon (c.f. D6.4 (9) et (10))

(2) $Post(p, t_h) = 1$ si $p = p_s$, et 0 sinon (c.f. D6.4 (5))

La suppression de la transition t_n donne un réseau intermédiaire $R'' = < P'', T''$; Pre", Post" > défini comme suit :

(3) $P'' = P' = P + \{ ps \}$ (c.f. D 6.4 (2))

(4) $T'' = T' - \{ t_n \} = T - \{ t_x \} + \{ t_h, t_f \}$ (D6.4 (3))

(5) $Pre'' = Pre' \Rightarrow$ $Pre''(p, t_h) = Pre(p, t_x)$ si $p \in P$ et 0 sinon (c.f. D6.4 (4))

$Pre''(p_s, t_f) = 1$ (c.f. D6.4 (6))

$\forall t \in T - \{ t_x \} : Pre''(p, t) = Pre(p, t)$ si $p \in P$ et 0 sinon (c.f. D6.4 (8))

(6) $Post'' = Post' \Rightarrow$ $Post''(p_s, t_h) = 1$ (D6.4 (5))

$Post''(p, t_f) = Post(p, t_x)$ si $p \in P$ et 0 sinon (D6.4 (7))

$\forall t \in T - \{ t_x \}$ $Post''(p, t) = Post(p, t)$ si $p \in P$ et 0 sinon (c.f. D 6.4 (8)).

En se référant à la définition D6.2, on voit bien que R" correspond tout à fait au réseau que nous aurions eu en appliquant à R_E la primitive SEQ. La propriété P6.3 nous assure que si R est vivant et conservatif, alors R" aussi l'est .

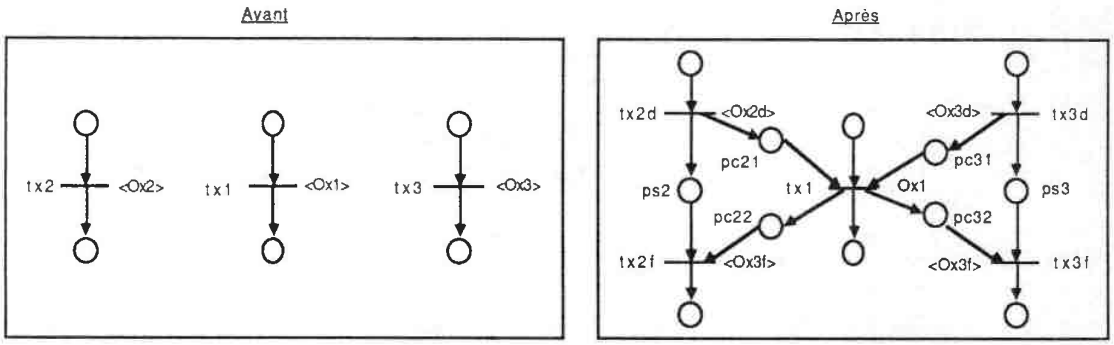
Par ailleurs, comme la suppression de transition neutre supprimable préserve ces deux propriétés, on est ainsi assuré que le réseau R' qu'on obtient à partir de R" en appliquant la transformation inverse d'une suppression de transition neutre est aussi vivant et conservatif. ⊗

d) Cas de la primitive EXC

Pour cette primitive, la propriété P6.3 découle du fait que les transformations qu'elle entraine sur un réseau correspondent aux transformations inverses de la règle de réduction par simplification de place implicite (c.f. section A.6 définition DA.2). La place p_I qu'elle introduit est à la fois en entrée et en sortie des transitions t_{x1}, ... t_{xn}. Le marquage de cette place sera donc toujours égal à 1, et ne sera jamais un obstacle au franchissement d'une transition. Elle est de ce fait une place implicite dont l'ajout ne modifie pas la vivacité et le caractère conservatif du réseau R. ⊗

B.3 DEMONSTRATION DE LA PROPRIETE P6.4

La propriété P6.4 se démontre en constatant que l'on peut obtenir le réseau R_E' à partir du réseau R_E" en appliquant trois types de règles de réduction qui préservent les propriétés de R_E": la simplification de place implicite, la substitution de place substituable, et la pré-agglomération. Nous illustrons cette démonstration à partir de l'exemple de la figure B.1. Cet exemple schématise pour trois opérations O_{x1}, O_{x2}, et O_{x3} , les transformations qu'il faut appliquer à R_E pour obtenir R_E". Nous montrons donc qu'après réduction de R", le réseau qui en résulte est le même que le réseau R_E' que l'on obtient en appliquant sur R_E la primitive SIM.



(a) (b)

Figure B.1

Dans la figure B.1 (b), les places p_{s2} et p_{s3} introduites par la primitive SEQ sont de toute évidence des places implicites. Leur marquage n'est jamais un obstacle au franchissement des transitions qu'elles ont en sortie. Leur suppression donne le réseau de la figure B.2.

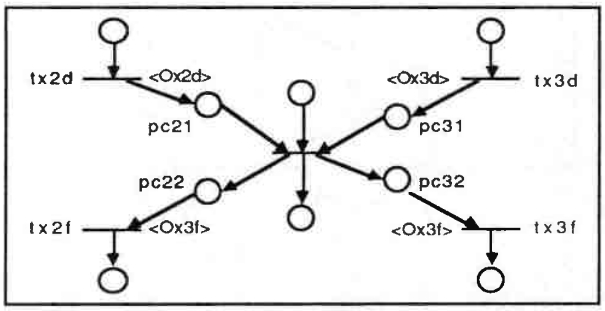


Figure B.2

Dans le réseau de cette figure, les places p_{c22} et p_{c32} sont des places substituables dont la substitution donne le réseau de la figure B.3.

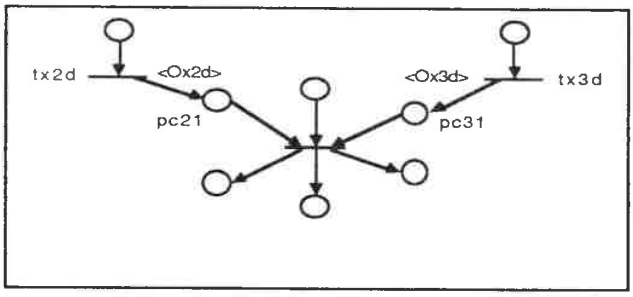


Figure B.3

Enfin, les places en entrée de tx2d et tx3d n'étant pas partagées (c.f. D6.7 (2)), on peut par pré-agglomération obtenir le réseau de la figure B.4. Ce qui montre bien que les réseaux R' et R" ont les mêmes propriétés.

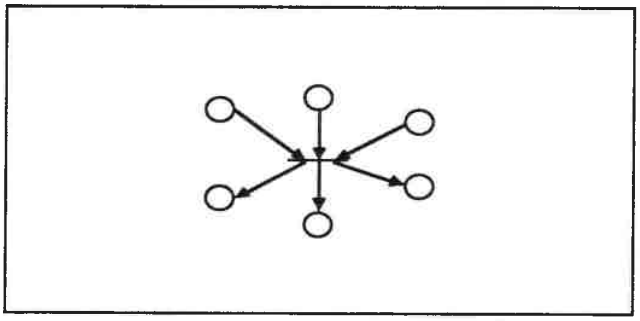


Figure B.4

B.4 DEMONSTRATION DE LA PROPRIETE P6.5

Cette propriété découle de la contrainte (2) de la règle D6.6.

Si R est un graphe simple, par définition,

$\forall (p, p') \in P^2$, si $p^{\cdot} \cap p'^{\cdot} \neq \varnothing$ alors soit $p^{\cdot} \subseteq p'^{\cdot}$ soit $p'^{\cdot} \subseteq p^{\cdot}$.

On peut noter que la place p_c qui est ajoutée par la primitive PRE est telle que :

$$\forall t \in T' - \{t_{y1},.....,t_{ym}\} , \forall p \in {}^{\cdot}t : p^{\cdot} \cap p_c{}^{\cdot} = \varnothing \quad \text{(c.f. D6.6 (5) et (9))}$$

Si m = 1, la place p_c introduite par la primitive PRE ne sera en entrée que d'une seule transition t_{y1}. Dans ce cas, la contrainte D6.6 (2) n'interdit pas d'avoir en entrée de t_{y1} des places partagées:

$$\forall p \in {}^{\cdot}t_{y1} : p_c{}^{\cdot} = \{ t_{y1} \} \subseteq p^{\cdot}$$

Par contre si m > 1, la place p_c introduite par la primitive PRE sera en entrée de toutes les transitions $t_{y1},.....,t_{ym}$. La contrainte exprimée par D6.6 (2) impose dans ces conditions que chaque transition t_{yi} n'ait au départ en entrée que des places non partagées. Par consequent,

$$\forall t \in \{ t_{y1},.....,t_{ym}\} \ \forall p \in {}^{\cdot}t : p^{\cdot} = \{ t \} \subseteq p_c{}^{\cdot}$$

On peut alors conclure que l'introduction de la place p_c préserve la classe du réseau.⊗

B.5 DEMONSTRATION DE LA PROPRIETE P6.6

Soient f ∈ F un semi-flot et C la matrice d'incidence du réseau R représentés comme suit :

$$f = \begin{pmatrix} f1 \\ f2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ fm \end{pmatrix} \qquad C = \begin{pmatrix} c11 \ldots\ldots c1n \\ c21 \ldots\ldots c2n \\ \ldots\ldots \\ \ldots\ldots \\ \\ \\ \\ cm1 \ldots\ldots cmn \end{pmatrix}$$

Comme le montre la définition D6.6, l'application de la primitive PRE sur R consiste à ajouter à ce réseau une place p_c , et à lier cette place à des transitions déjà existantes. De ce fait, la matrice d'incidence C' de R' est identique à C à une ligne près. Cette ligne correspond à la place p_c et est notée dans ce qui suit par (v1, v2, ..., vn)).

$$C' = \begin{pmatrix} c11 \ldots\ldots c1n \\ c21 \ldots\ldots c2n \\ \ldots\ldots \\ \ldots\ldots \\ \\ \\ \\ cm1 \ldots\ldots cmn \\ v1 \ldots\ldots vn \end{pmatrix}$$

Pour tout $f \in F$, le vecteur $f' \in \mathbb{N}^{m+1}$ tel que

$$f'(i) = f(i) \text{ si } i = 1, \ldots m \text{ et } 0 \text{ sinon}$$

est un semi-flot de R' car $f^t * C = 0 \Rightarrow f'^t * C' = 0$.⊗

ANNEXE C

DEMONSTRATIONS DU CHAPITRE 7

PLAN

C.0 INTRODUCTION

Dans la section 2.5.2, nous avions défini la sémantique des opérations de communication synchrone en indiquant comment des transitions étiquetées par un même rendez-vous pouvaient être fusionnées pour former un réseau global équivalent. L'exemple de la figure C.1 rappelle le principe de cette fusion. Dans cet exemple, la transition $\underline{t}$ introduite par la fusion est une transition prioritaire, et doit de ce fait être franchie immédiatement après chaque tir de la transition t, si la place p_3 est suffisamment marquée.

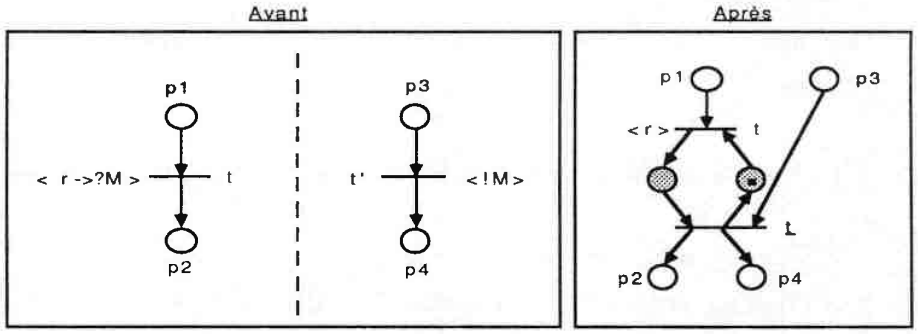


Figure C.1

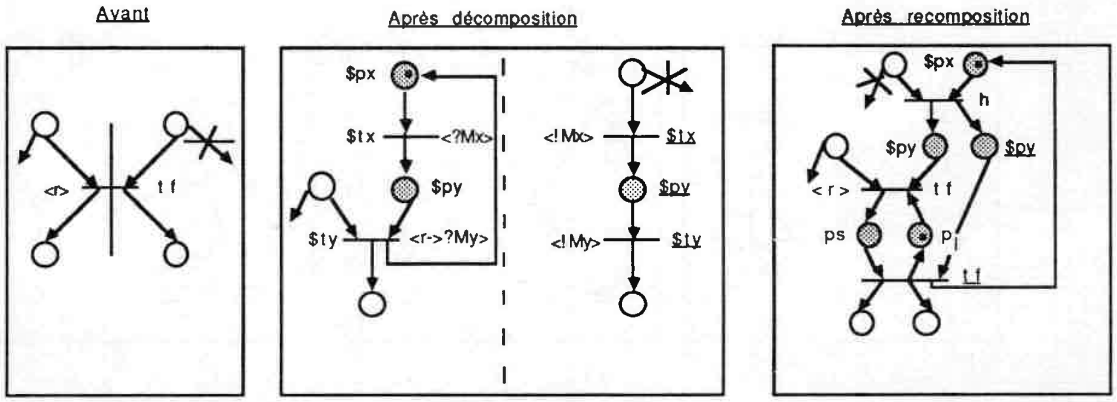


Figure C.2

On peut ainsi associer très simplement aux transformations de chaque règle de décomposition, les transformations inverses qui permettent d'obtenir à nouveau un réseau global. Comme le montre la figure C.2 pour la règle RD04, par rapport au réseau initial, le réseau que l'on obtient de cette façon est surchargé par toutes les places et transitions introduites par les règles de décomposition, pour permettre aux sous-réseaux de garantir les mêmes conditions d'évolution que celles décrites par les transitions frontières.

Dans cet annexe, nous démontrons la validité des règles de décomposition proposées en montrant qu'à partir du réseau initial, on peut, sans modifier le comportement initialement défini, introduire la surcharge qu'on obtient par recomposition des sous-réseaux, en utilisant les trois règles de réduction suivantes:

1°) La règle de réduction par simplification de place implicite

Cette règle préserve l'abstraction par rapport à tout sous-ensemble de transitions [André 81] [Berthelot 85 (a)]. La figure C.3 donne les schémas types de réduction par simplification de place implicite qui nous seront utils. Il est clair que le marquage des places supprimées n'est jamais un obstacle au franchissement des transitions dont elles sont une entrée.

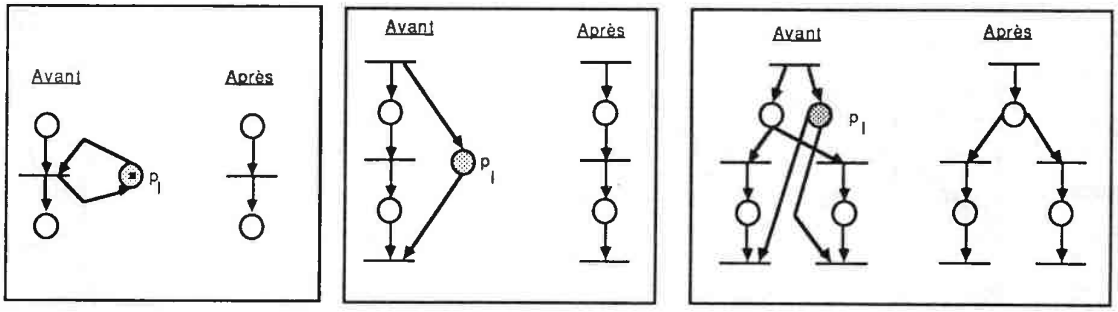


Figure C.3

2°) La règle de réduction par substitution de place

Dans les situations où H contient une seule transition notée h et F une seule transition notée f comme indiqué dans la figure C.4, la suppression d'une place substituable p_s préserve l'abstraction par rapport à tout sous-ensemble de transitions incluant h [André 81].

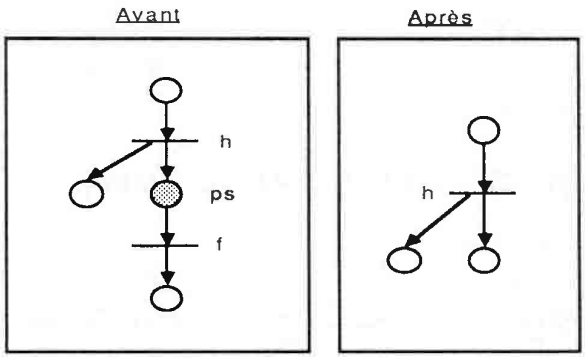
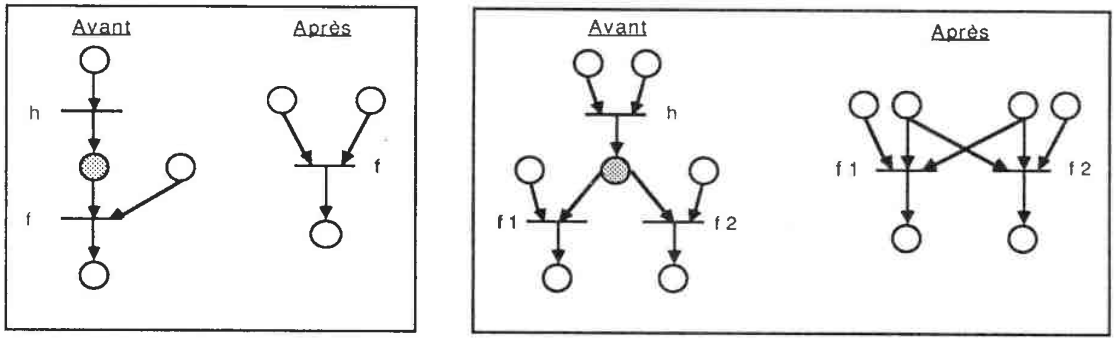


Figure C.4

3°) La règle de réduction par pré-agglomération

La figure C.5 donne les schémas types de réduction par pré-agglomération que nous utilisons. Dans le réseau de la figure C.5 (a), la pré-agglomération préserve l'abstraction par rapport à tout sous-ensemble de transitions incluant f. De la même façon, dans le réseau de la figure C.5 (b), la pré-agglomération préserve l'abstraction par rapport à tout sous-ensemble de transitions incluant f1 et f2 [Berthelot 85 (a)].



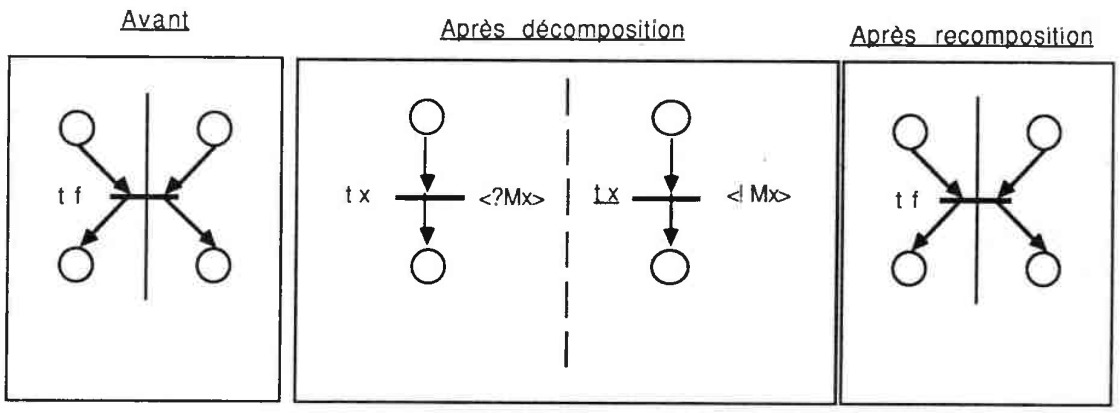
(a)

(b)

Figure C.5

C.1 VALIDATION DE LA REGLE RD02

Comme le montre la figure C.6, pour cette règle, le réseau que l'on obtient aprés recomposition est identique au réseau initial. De ce fait, la décomposition ne modifie pas le comportement du réseau initial.



(a)

(b)

(c)

Figure C.6

C.2 VALIDATION DE LA REGLE RD03

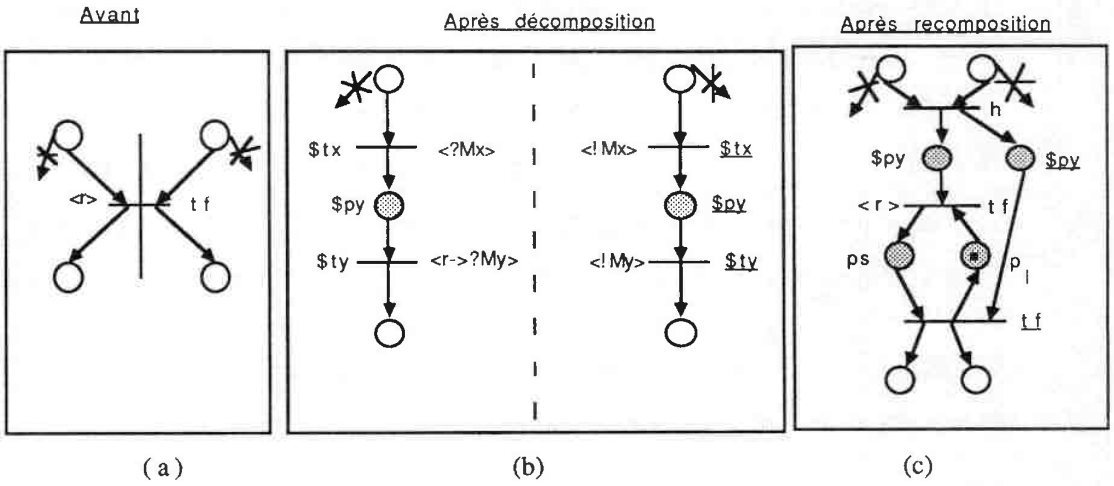


Figure C.7

Pour cette règle, le réseau que l'on obtient par recomposition est différent du réseau initial (c.f. figure C.7 (c)). On peut cependant l'obtenir de la façon suivante à partir du réseau initial en appliquant quatre opérations inverses de réduction.

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction.

1°) Les places en entrée de la transition frontière t_f n'étant pas partagées, on peut appliquer sur cette transition l'opération inverse de la règle de réduction par pré-agglomération pour obtenir le réseau de la figure C.8 (a). Cette transformation préserve l'abstraction sur T_d.

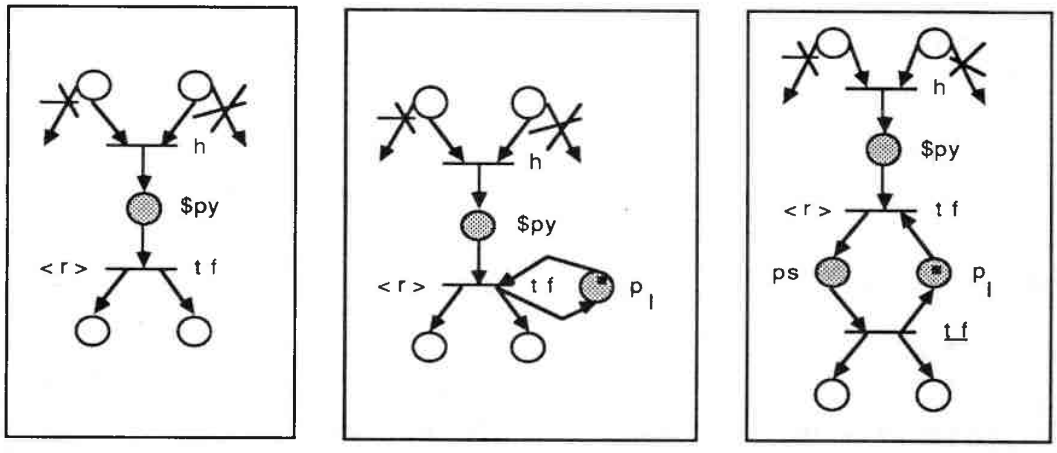
2°) Par ailleurs, l'ajout de la place implicite p_I qui permet d'obtenir le réseau de

la figure C.8(b) préserve l'abstraction sur T_d.

3°) On peut alors ensuite appliquer sur la transition t_f l'opération inverse de la règle de réduction par substitution de place pour obtenir le réseau de la figure C.8 (c). Cette opération préserve également l'abstraction sur T_d. En outre, si l'on considère $\underline{t_f}$ comme une transition prioritaire, chaque tir de t_f sera immédiatement suivi du tir de $\underline{t_f}$. En d'autres termes, les conséquences du tir de t_f seront toujours identiques dans le réseau initial et dans le réseau transformé.

4°) Enfin, en ajoutant la place implicite $\underline{\$p_y}$ qui ne modifie pas non plus l'abstraction sur T_d, on obtient le réseau de la figure C.7(c).

Ce qui permet de conclure que la règle RD03 ne modifie pas le comportement du réseau initial : Dans le réseau initial tout comme dans le réseau transformé, le tir de la transition t_f est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions incluant t_f.



(a) (b) (c)

Figure C.8

C.3 VALIDATION DE LA REGLE RD04

Pour cette règle également, le réseau résultant de la recomposition diffère du réseau initial (c.f. figure C.2(c)), mais peut se déduire de ce réseau par application des règles de réduction.

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction.

1°) L'ajout de la place implicite $\$p_x$ comme indiqué sur la figure C.9(a) préserve l'abstraction sur T_d.

2°) Comme les places en entrée de t_f contenues dans E_f" (c.f. D7.4(5)) et la place $\$p_x$ ne sont pas partagées, on peut ensuite appliquer les transformations inverses de la pré-agglomération pour obtenir le réseau de la figure C.9(b). Cette transformation préserve également l'abstraction sur T_d.

3°) Par ailleurs, l'ajout de la place implicite p_I comme indiqué sur la figure C.9(c) préserve également l'abstraction sur T_d.

4°) L'application des transformations inverses de la règle de réduction par substitution de place sur t_f donne alors le réseau de la figure C.9(d). Cette transformation préserve également l'abstraction sur T_d. En outre, si l'on considère $\underline{t_f}$ comme une transition prioritaire, chaque tir de t_f dans le réseau transformé sera immédiatement suivi d'un tir de $\underline{t_f}$ et entrainera les mêmes conséquences que dans le réseau initial.

5°) Enfin, en ajoutant la place implicite $\underline{\$p_y}$ qui ne modifie pas non plus

l'abstraction sur T_d, on obtient le réseau de la figure C.2(c).

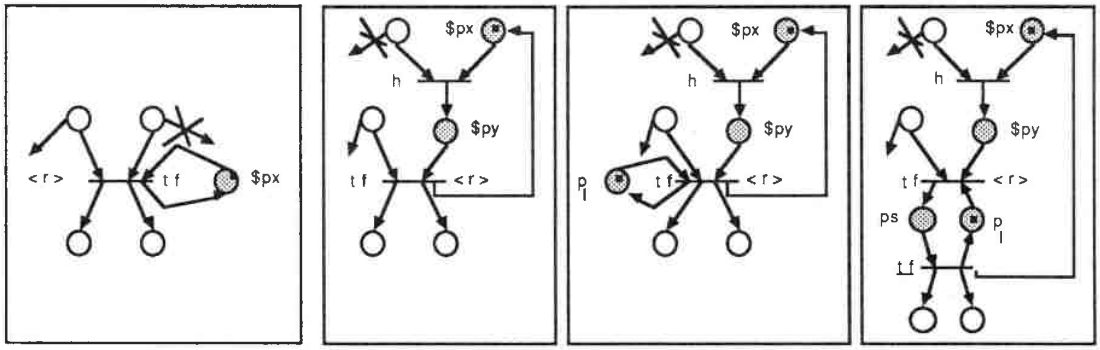


Figure C.9

Ce qui permet de conclure que la règle RD04 ne modifie pas le comportement du réseau initial : Dans le réseau initial tout comme dans le réseau transformé, le tir de la transition t_f est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions incluant t_f.

C.4 VALIDATION DE LA REGLE RD05

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction. Pour montrer que la règle RD05 ne modifie pas le comportement défini par le réseau initial, nous montrons que les transformations qu'on applique à ce réseau pour obtenir le réseau de la figure C.10(c) issu de la recomposition, préservent le comportement.

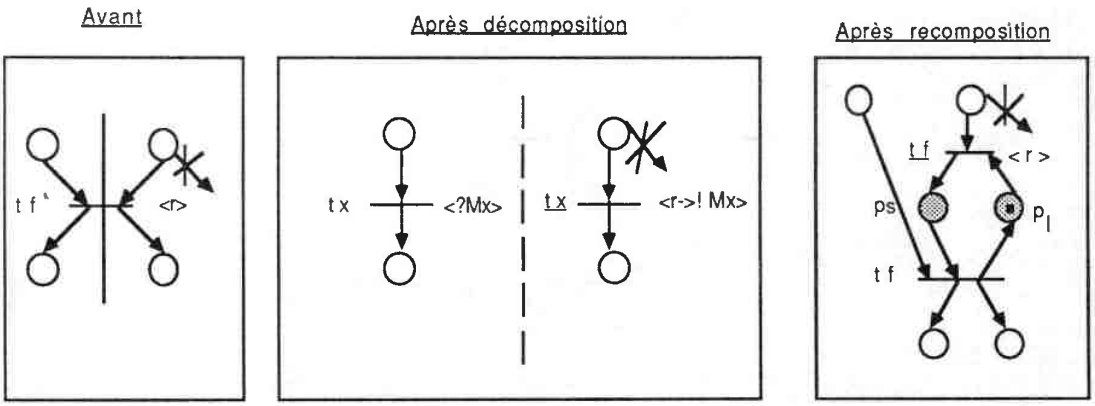


Figure C.10

1°) L'ajout de la place implicite p_I comme indiqué sur la figure C.11 préserve l'abstraction sur T_d.

2°) Comme les places en entrée de t_f contenues dans E_f" (c.f. D7.5(4)) et la nouvelle place p_I ne sont pas partagées, on peut alors ensuite appliquer les transformations inverses de la pré-agglomération pour obtenir le réseau de la figure C.10(c). Cette transformation préserve également l'abstraction sur T_d. Par ailleurs les conséquences du tir de la transition t_f sont les mêmes dans le réseau initial et dans le réseau transformé, et le tir de la transition t_f dans le réseau transformé sera toujours précédé du tir de la transition $\underline{t_f}$ à laquelle le prédicat initialement associé à t_f est affecté. Comme ce prédicat correspond à un évènement dont la politique de prise en compte des occurrences est indépendante du temps (c.f. définition D7.5(1)), on peut considérer que le comportement initialement défini sera préservé.

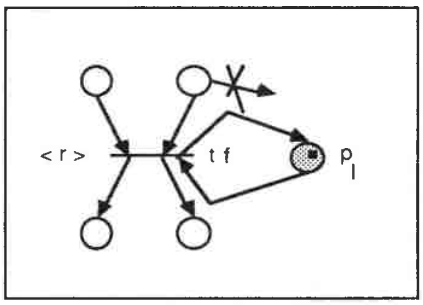
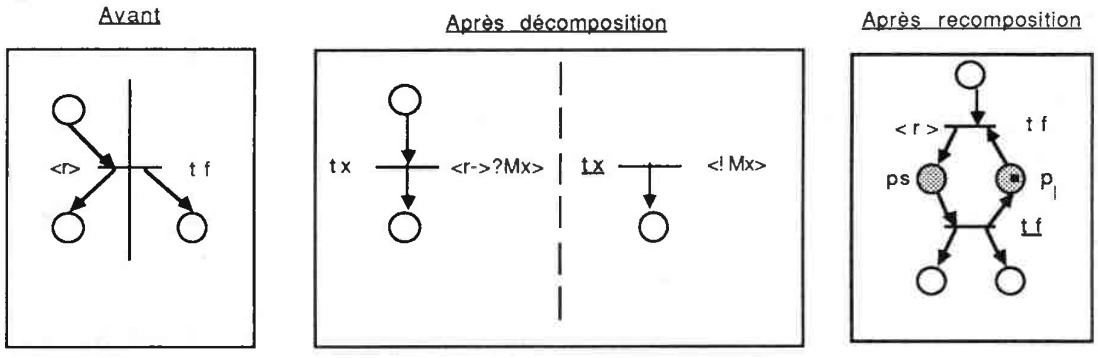


Figure C.11

C.5 VALIDATION DE LA REGLE RD08

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction. Le résultat de la recomposition (c.f. figure C.12(c)) peut s'obtenir à partir du réseau initial en appliquant deux règles de réduction qui préservent l'abstraction sur T_d :



(a) (b) (c)

Figure C.12

1°) On ajoute tout d'abord au réseau initial la place implicite p_I comme indiqué sur la figure C.13. Cette transformation ne modifie pas l'abstraction sur T_d.

2°) Ensuite, on applique à t_f les transformations inverses de la règle de réduction par substitution de place pour obtenir le réseau de la figure C.12(c). Cette transformation préserve également l'abstraction sur T_d. En outre, comme la transition $\underline{t_f}$ est prioritaire, chaque tir de t_f dans le réseau transformé sera immédiatement suivi d'un tir de $\underline{t_f}$ et entrainera les mêmes conséquences que dans le réseau initial.

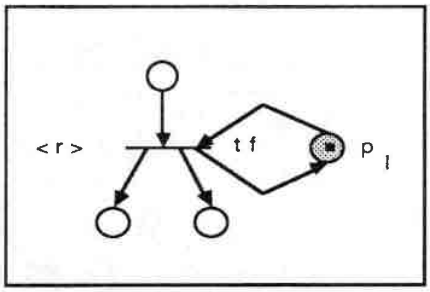


Figure C.13

Ce qui permet de conclure que la règle RD08 ne modifie pas le comportement du réseau initial : Dans ce réseau tout comme dans le réseau transformé, le tir de la transition t_f est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions incluant t_f.

C.6 VALIDATION DE LA REGLE RD09

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction. Pour la règle RD09, le réseau résultant de la recomposition (c.f. figure C.14(c)) peut s'obtenir en appliquant sur le réseau initial cinq transformations qui préservent l'abstraction sur T_d :

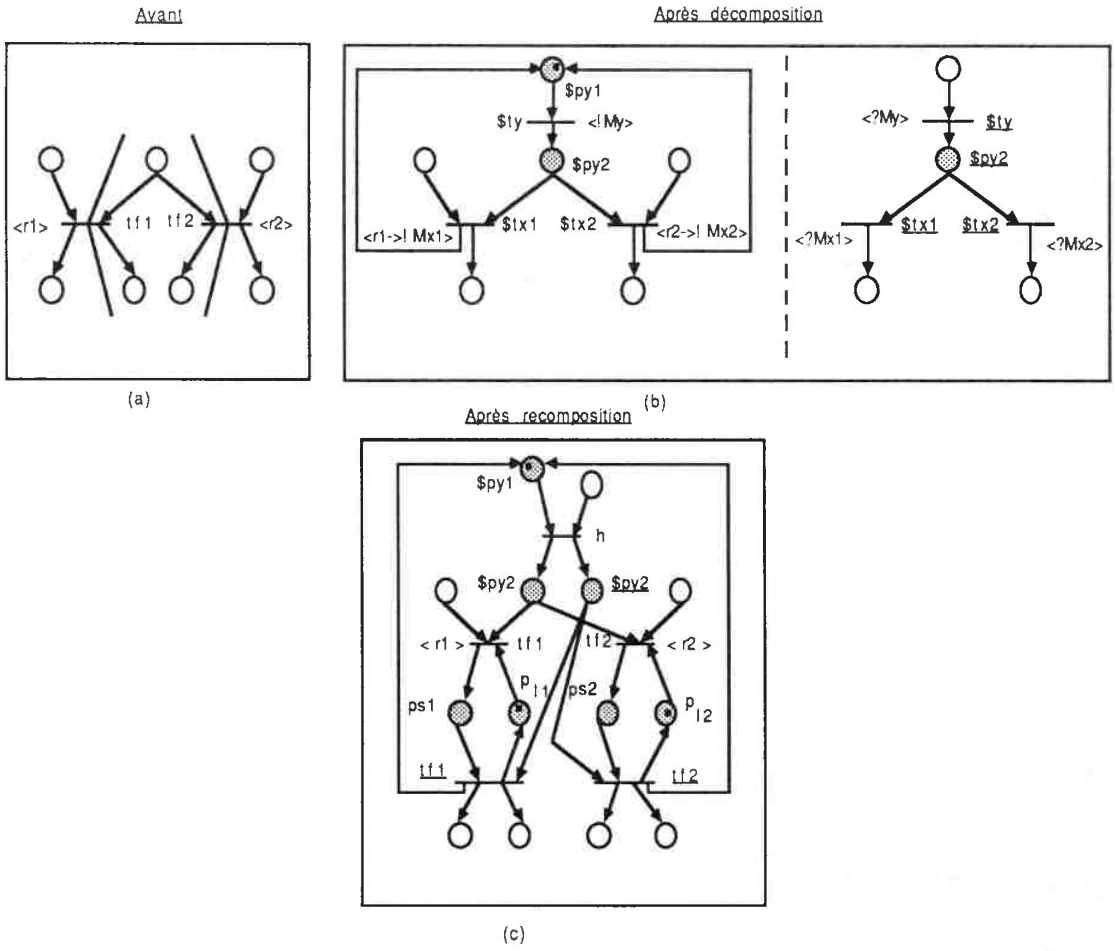


Figure C.14

1°) Chaque transition frontière $t_{fi} \in T_f$ reçoit en entrée et en sortie la même place implicite $\$p_{y1}$ comme indiqué sur la figure C.15(a).

2°) Comme les places contenues dans E_c (c.f. définition D7.9(1)) et la nouvelle place $\$p_{y1}$ n'ont pour transitions de sortie que les transitions contenues dans T_f, on peut appliquer sur ces transitions les transformations inverses de la pré-agglomération pour obtenir le réseau de la figure C.15(b).

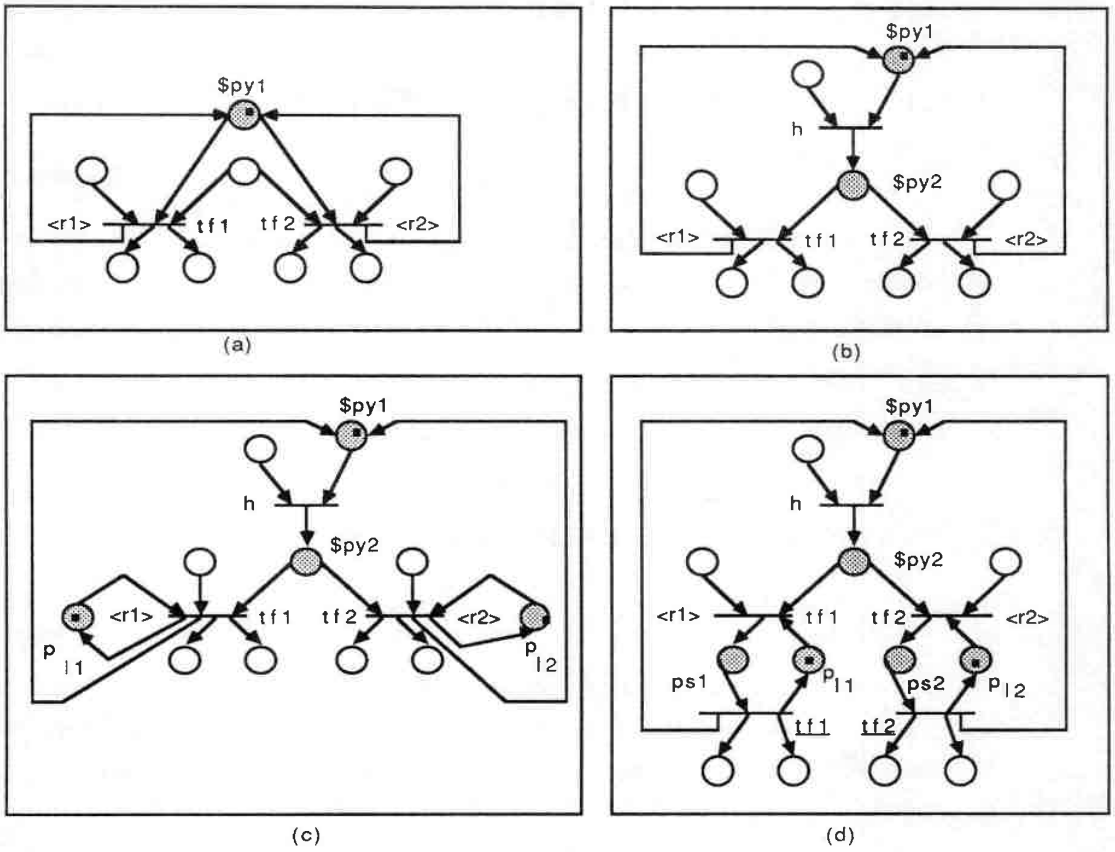


Figure C.15

3°) Chaque transition t_{fi} reçoit ensuite en entrée et en sortie une place implicite spécifique p_{Ii} comme indiqué par le réseau de la figure C.15(c).

4°) On applique alors à chaque transition t_{fi} les transformations inverses de la règle de réduction par substitution de place pour obtenir le réseau de la figure C.15(d). Si l'on considère les transitions $\underline{t_{fi}}$ comme des transitions prioritaires, le tir d'une transition t_{fi} dans le réseau transformé sera toujours immédiatement suivi du tir de la transition $\underline{t_{fi}}$ qui lui correspond pour entrainer les mêmes

conséquences que dans le réseau initial.

5°) Enfin, en ajoutant la place implicite $\$p_{y2}$, on obtient le réseau de la figure C.14(c).

On peut alors conclure que la règle RD09 ne modifie pas le comportement du réseau initial : Dans ce réseau tout comme dans le réseau transformé, le tir d'une transition $t_{fi} \in T_f$ est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions incluant T_f.

C.7 VALIDATION DE LA REGLE RD10

Soit T_d avec $t_f \in T_d$ un sous-ensemble de transitions pour lequel le réseau initial admet une abstraction. Pour cette règle également, on peut obtenir à partir du réseau initial, le réseau qu'on obtient par recomposition (c.f. figure C.16(c)) en appliquant successivement sept transformations qui préservent l'abstraction sur T_d :

1°) Chaque transition frontière $t_{fi} \in T_f$ reçoit en entrée et en sortie la même place implicite $\$p_{z1}$ comme indiqué sur la figure C.17(a).

2°) Comme les places communes contenues dans E_c (c.f. définition D7.10(1)) et la nouvelle place $\$p_{z1}$ n'ont pour transitions de sortie que les transitions contenues dans T_f, on peut appliquer sur ces transitions les transformations inverses de la pré-agglomération pour obtenir le réseau de la figure C.17(b).

3°) Chaque transition t_{fi} reçoit ensuite en entrée et en sortie une place implicite spécifique $\$p_{wi}$ comme indiqué par le réseau de la figure C.17(c).

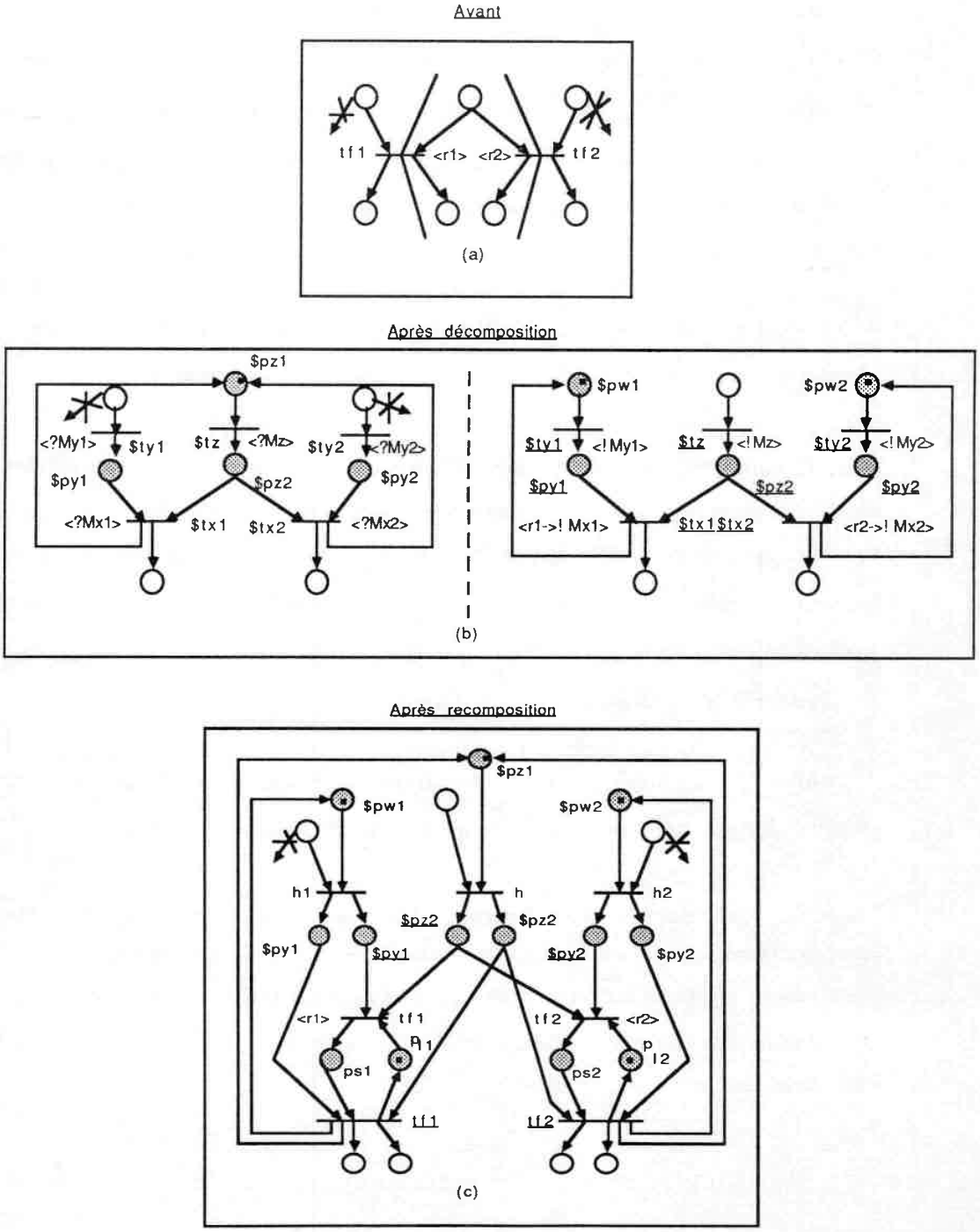

Avant
tf1
<r1>
<r2>
tf2
(a)
Après décomposition
$pz1
<?My1>
$ty1
$tz
<?Mz>
$ty2
<?My2>
$py1
$pz2
$py2
<?Mx1>
$tx1
$tx2
<?Mx2>
$pw1
$pw2
$ty1
<!My1>
$tz
<!Mz>
$ty2
<!My2>
$py1
$pz2
$py2
<r1->! Mx1>
$tx1 $tx2
<r2->! Mx2>
(b)
Après recomposition
$pz1
$pw1
$pw2
h1
h
h2
$pz2
$pz2
$py1
$py1
$py2
$py2
<r1>
tf1
tf2
<r2>
p11
p12
ps1
ps2
tf1
tf2
(c)

Figure C.16

4°) Comme pour chaque transition $t_{fi} \in T_f$ les places contenues dans E'_{fi} (c.f. définition D7.10(3)) et la nouvelle place $\$p_{wi}$ n'ont pour transition de sortie que la transition t_{fi}, on peut appliquer sur chacune de ces transitions les transformations inverses de la pré-agglomération pour obtenir le réseau de la figure C.17(d).

5°) Chaque transition t_{fi} reçoit à nouveau en entrée et en sortie une place implicite spécifique p_{Ii} pour donner le réseau de la figure C.17(e).

6°) On applique alors à chaque transition t_{fi} les transformations inverses de la règle de réduction par substitution de place pour obtenir le réseau de la figure C.17(f). Si l'on considère les transitions $\underline{t_{fi}}$ comme des transitions prioritaires, le tir d'une transition t_{fi} dans le réseau transformé sera toujours immédiatement suivi du tir de la transition $\underline{t_{fi}}$ qui lui correspond pour entrainer les mêmes conséquences que dans le réseau initial.

7°) Enfin, en ajoutant la place implicite $\$p_{z2}$, et pour chaque transition t_{fi} la place implicite $\$p_{yi}$ on obtient le réseau de la figure C.16(c).

Ce qui permet de conclure que la règle RD10 ne modifie pas le comportement du réseau initial : Dans ce réseau tout comme dans le réseau transformé, le tir d'une transition $t_{fi} \in T_f$ est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions incluant T_f.

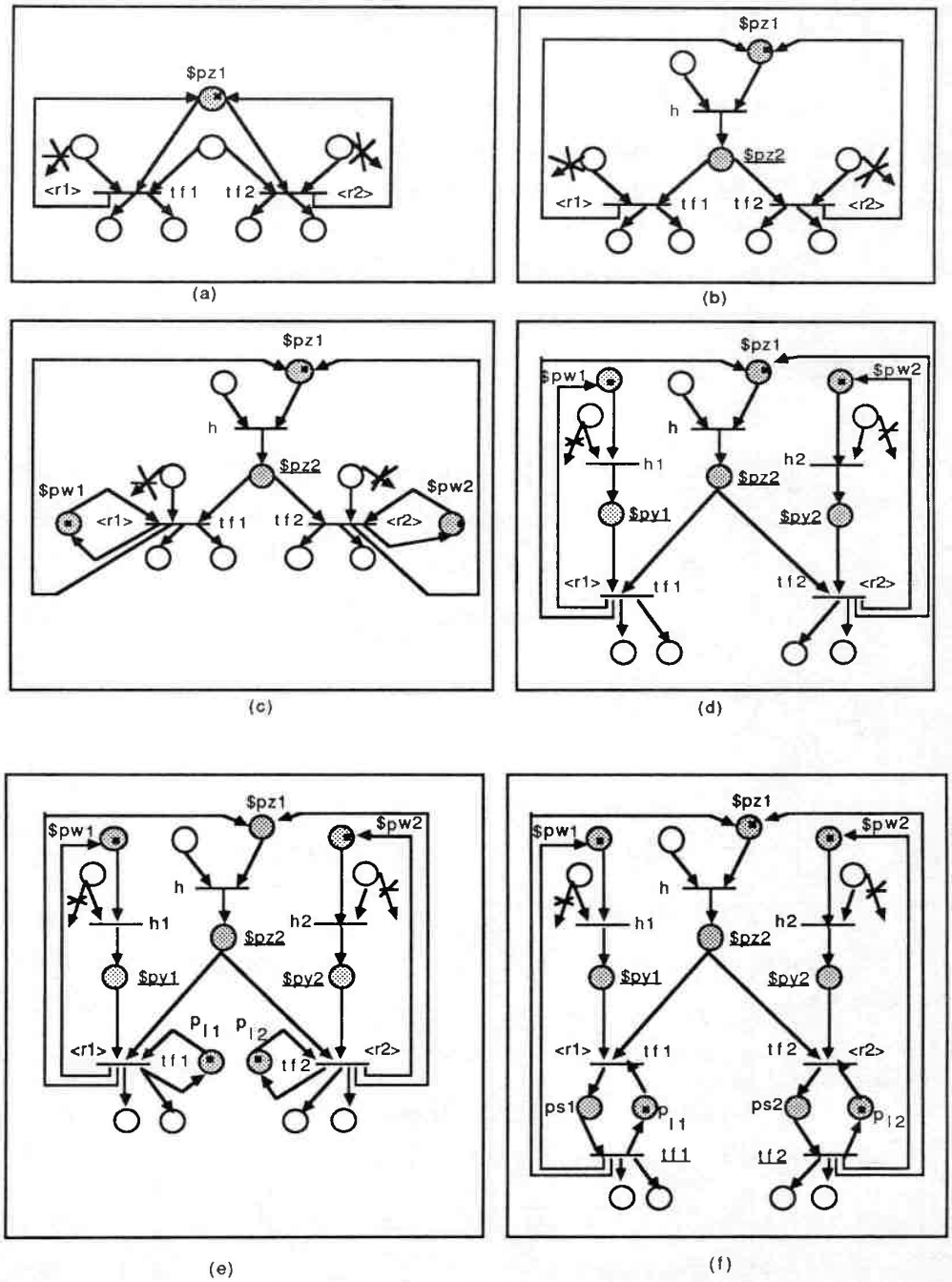

$pz1
<r1>
tf1
tf2
<r2>
(a)
h
$pz2
(b)
$pw1
$pw2
(c)
h1
h2
$py1
$py2
(d)
$pl1
$pl2
(e)
ps1
ps2
(f)

Figure 17

C.8 VALIDATION DE LA REGLE RD13

Pour valider la règle RD13, nous montrons ici également que le réseau global que l'on obtient par fusion des sous-réseaux résultant de la décomposition décrit le même comportement externe que le réseau initial.

A partir de la définition D7.13, nous déduisons tout d'abord ce qui suit:

Corollaire C.1 :

Appliquer la règle RD13 sur un réseau $R_I = < R ; V ; OP ; PR ; \Phi ; \Psi >$ avec $R = < P, T ; Pre, Post >$, puis fusionner ensuite les deux sous-réseaux obtenus revient à construire à partir de R_I un réseau $R_I' = < R' ; V' OP' ; PR'' ; \Phi' ; \Psi' >$ avec $R' = < P', T' ; Pre', Post' >$ de la façon suivante :

(1) Au départ $P' = \varnothing$, $T' = \varnothing$, $Pre' = Post' = 0$

(2) $\forall\ p_j \in P$:

- $P' <- P' + \{ p_j , \underline{p_j} \}$

  (*Pour chaque place* $p_j \in P$, *on inclut dans* P' *une place de même nom et une place associée* $\underline{p_j}$)

  NB : *Chaque couple* $(p_j, \underline{p_j})$ *représente des places dédoublées par la règle RD13 (c.f. définition D7.13 (3)).*

- $\Phi' (p_j) = \Phi (p_j)$, $\Phi' (\underline{p_j}) = NOP$

  (*Dans* R_I' *on associe l'opération définie par* $\Phi (p_j)$ à p_j et NOP à $\underline{p_j}$ (*c.f. également définition D7.13 (5)*))

(3) $\forall\ t_i \in T$ telle que $\Psi (t_i) = pr_0$:

- $T' <- T' + \{ t_i \}$

  (*Pour chaque transition* $t_i \in T$ *étiquetée par le prédicat toujours vrai, on*

inclut dans T' *une transition de même nom).*

- $\Psi'(t_i) = pr_0$

  (t_i *conserve la même interprétation dans* R_I *et* R_I')

- $\forall (p_j, \underline{p_j}) \in P' : Pre'(p_j, t_i) = Pre'(\underline{p_j}, t_i) = Pre(p_j, t_i)$

  $\forall (pj, \underline{p_j}) \in P' : Post'(p_j, t_i) = Post'(\underline{p_j}, t_i) = Post(p_j, t_i)$

  (*Dans* R_I', *chaque couple* $(p_j, \underline{p_j})$ *est placé en entrée ou en sortie de* t_i, *si dans* R_I p_j *est en entrée ou en sortie de* t_i)

N.B. : *Ces transformations découlent de la règle de fusion de transitions étiquetées par un même rendez-vous et non synchronisées à l'évolution de l'environnement. Les transitions fusionnées correspondent aux transitions dédoublées par la règle RD13 qui sont étiquetées par le prédicat* pr_0 (*c.f. définition D7.13(3)).*

(4) $\forall t_i \in T$ telle que $\Psi(t_i) \neq pr_0$:

- $T' \leftarrow T' + \{ t_i + \underline{t_i} \}$

  (*Pour chaque transition* $t_i \in T$ *étiquetée par un prédicat, on inclut dans* T' *une transistion de même nom et une transition prioritaire associée* $\underline{t_i}$).

- $\Psi'(t_i) = \Psi(t_i)$

  $\Psi'(\underline{t_i}) = pr_0$

  (t_i *conserve le même prédicat dans* R_I *et* R_I' *et on associe le prédicat toujours vrai à* $\underline{t_i}$)

- $P' \leftarrow P' + \{ p_{1i}, p_{2i} \}$

  (*On inclut dans* P' *deux nouvelles places* p_{1i} *et* p_{2i} *associée à* t_i)

  $M'(p_{1i}) = 1$ (*On marque la place* p_{1i} *à* 1)

- $\forall (p_j, \underline{p_j}) \in P' : Pre'(p_j, t_i) = Pre'(\underline{p_j}, \underline{t_i}) = Pre(p_j, t_i)$

  (*Pour chaque couple* $(p_j, \underline{p_j}) \in P'$ *tel que* p_j *est en entrée de* t_i *dans* R_I,

dans R_I' p_j *est placée en entrée de* t_i *et* $\underline{p_j}$ *en entrée de* $\underline{t_i}$).

- $\forall\ (p_j, \underline{p_j}) \in P' : Post'(p_j, \underline{t_i}) = Post'(\underline{p_j}, \underline{t_i}) = Post(p_j, t_i)$

  (*Pour chaque couple* $(p_j, \underline{p_j}) \in P'$ *tel que* p_j *est en sortie de* t_i *dans* R_I, *dans* R_I' p_j *et* $\underline{p_j}$ *sont placées en sortie de* $\underline{t_i}$)

- $Pre'(p_{1i}, t_i) = Post'(p_{1i}, \underline{t_i}) = 1$

  (*La nouvelle place* p_{1i} *est placée en entrée de* t_i *et en sortie de* $\underline{t_i}$)

- $Pre'(p_{2i}, \underline{t_i}) = Post'(p_{2i}, \underline{t_i}) = 1$

  (*La nouvelle place* p_{2i} *est placée en entrée de* $\underline{t_i}$ *et en sortie de* t_i).

N.B. : *Ces transformations découlent de la règle de fusion de transitions étiquetées par un même rendez-vous et synchronisées à l'évolution de l'environnement. Ici la fusion conserne les transitions dédoublées par la règle RD13 qui sont étiquetées par un prédicat différent de* pr_0 (*c.f. définition D7.13(3)).*

La figure C.18 illustre l'application de ces différentes transformations. Les places p_1, $\underline{p_1}$, p_2, $\underline{p_2}$, p_3, $\underline{p_3}$, p_4, $\underline{p_4}$ du réseau fusionné ont été obtenues comme indiqué en (2) , les places p_{12} et p_{22} comme indiqué en (4), les transitions t_1 et t_3 comme indiqué en (3), et les transitions t_2, $\underline{t_2}$ comme indiqué en (4).

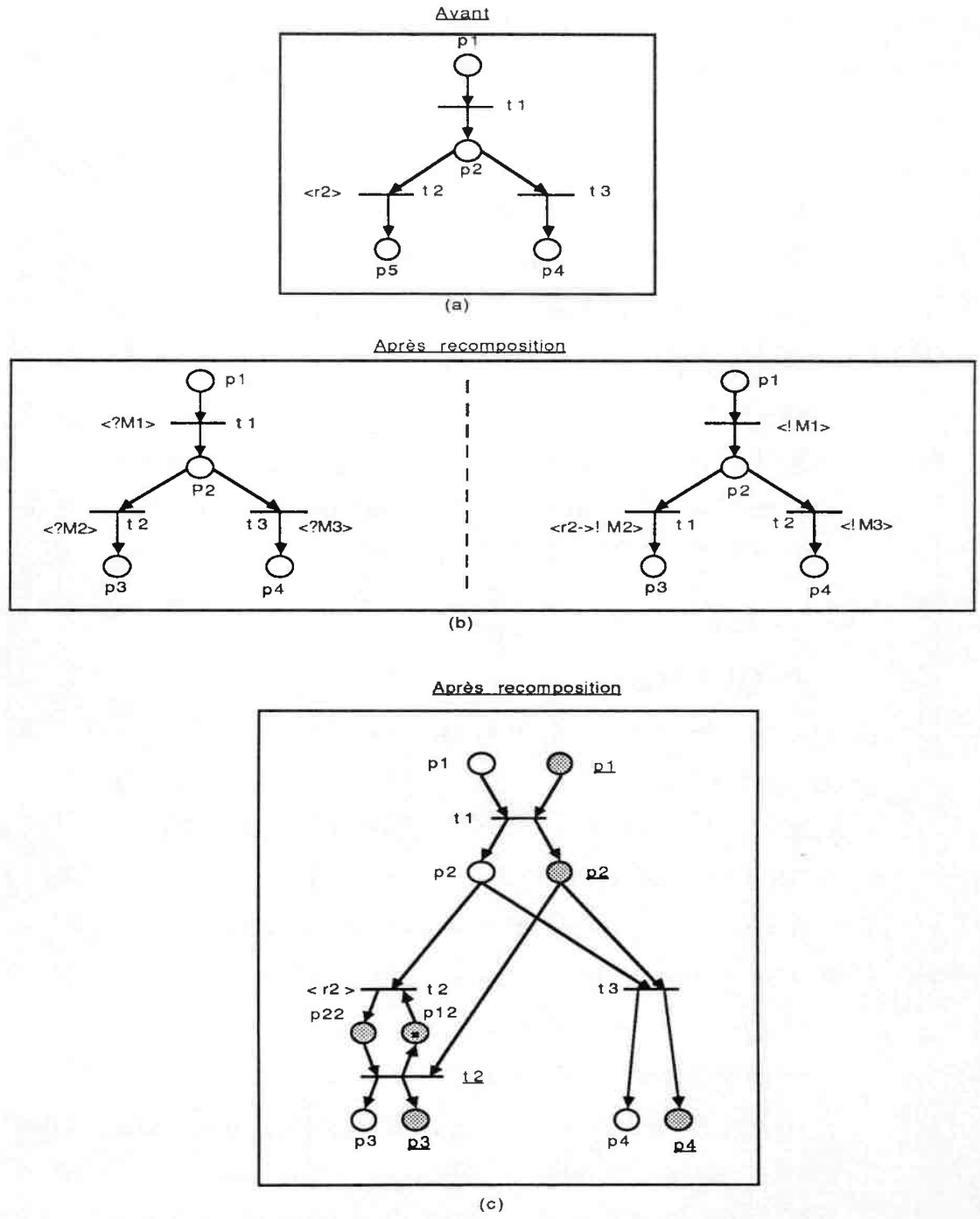

Avant
p1
t 1
p2
<r2>
t 2
t 3
p5
p4
(a)
Après recomposition
p1
<?M1>
t 1
P2
<?M2>
t 2
t 3
<?M3>
p3
p4
p1
<! M1>
p2
<r2->! M2>
t 1
t 2
<! M3>
p3
p4
(b)
Après recomposition
p1
p1
t 1
p2
p2
< r2 >
t 2
t 3
p22
p12
t 2
p3
p3
p4
p4
(c)

Figure C.18

En partant du réseau initial R_I, on peut également obtenir le réseau fusionné R_I' en appliquant successivement les transformations suivantes :

(1) Au départ $R_I' = R_I$

(2) $\forall\ t_i \in T$ telle que $\Psi\ (t_i) \neq pr_0$

- $P' \leftarrow P' + \{ p_{1i} \}$ (*On inclut dans* P' *la place* p_{1i} *associée à* t_i)
- $M'(p_{1i}) = 1$ (*On marque la place* p_{1i} à 1)
- $Pre'\ (p_{1i}, t_i) = Post'\ (p_{1i}, t_i) = 1$

  (*On met la place* p_{1i} *en entrée et en sortie de la transition* t_i).

NB : *Les places* p_{1i} *ajoutées de cette façon sont toutes des places implicites. De ce fait, cette première transformation préserve l'abstraction sur tout sous-ensemble de transitions* $Td \subset T' = T$.

(3) $\forall\ t_i \in T$ telle que $\Psi\ (t_i) \neq pr_0$:

- $P' \leftarrow P' + \{ p_{2i} \}$

  (*On inclut dans* P' *la place* p_{2i} *associée à* t_i)
- $T' \leftarrow T' + \{ \underline{t_i} \}$

  (*On inclut dans* T' *la transition prioritaire* $\underline{t_i}$ *associée à* t_i)
- $\forall\ p \in P' : Post'\ (p, t_i) = 1$ si $p = p_{2i}$ 0 sinon

  (*La place* p_{2i} *devient la seule place en sortie de* t_i)
- $\forall\ p \in P' : Pre'\ (p, \underline{t_i}) = 1$ si $p = p_{2i}$ 0 sinon

  (*La place* p_{2i} *est placée en entrée de* $\underline{t_i}$)
- $\forall\ p \in P' : Post'\ (p, \underline{t_i}) = Post\ (p, t_i)$ si $p \in P$ 0 sinon

  (*Toutes les places sortie de* t_i *dans* R_I *deviennent des places sortie de* $\underline{t_i}$)

NB : *Toutes les places* p_{2i} *ajoutées de cette façon correspondent à des places*

substituables. Cette deuxième transformation préserve donc également l'abstraction sur tout sous-ensemble de transitions $Td \subset T' - \{ \sum \underline{t_i} \} = T$ ($\sum \underline{t_i}$ *désigne l'ensemble des nouvelles transitions* $\underline{t_i}$ *ajoutées). Par ailleurs, si l'on considère les transitions* $\underline{t_i}$ *comme des transitions prioritaires, le tir d'une transition* t_i *dans le réseau transformé sera toujours immédiatement suivi du tir de la transition* $\underline{t_i}$ *qui lui correspond pour entrainer les mêmes conséquences que dans le réseau initial. On notera également que si l'on fait abstraction des places* $\underline{p_j}$, *ces deux premières transformations sont équivalentes au point (4) du corollaire précédent.*

(4) $\forall p_j \in P$:

- $P' \leftarrow P' + \{ \underline{p_j} \}$ (*On inclut dans P' la place* $\underline{p_j}$ *associée à* p_j)
- $\forall t_i \in T$ telle que $\Psi (t_i) = pr_0$: $Pre' (\underline{p_j}, t_i) = Pre (p_j, t_i)$

  $\forall t_i \in T$ telle que $\Psi (t_i) \neq pr_0$: $Pre' (\underline{p_j}, \underline{t_i}) = Pre (p_j, t_i)$

  (*Toute transition* t_i *qui a en entrée* p_j *dans* R_I' *reçoit en plus en entrée* $\underline{p_j}$ *si elle est étiquetée par* pr_0, *sinon* $\underline{p_j}$ *est placée en entrée de la transition* $\underline{t_i}$ *qui lui est associée*).
- $\forall t_i \in T$ telle que $\Psi (t_i) = pr_0$: $Post' (\underline{p_j}, t_i) = Post (p_j, t_i)$

  $\forall t_i \in T$ telle que $\Psi (t_i) \neq pr_0$: $Post' (\underline{p_j}, \underline{t_i}) = Post (p_j, t_i)$

  (*Toute transition qui a en sortie* p_j *dans* R_I' *reçoit en plus une sortie* $\underline{p_j}$ *si elle est étiquetée par* pr_0, *sinon* $\underline{p_j}$ *est placée en sortie de la transition* $\underline{t_i}$ *qui lui est associée*).

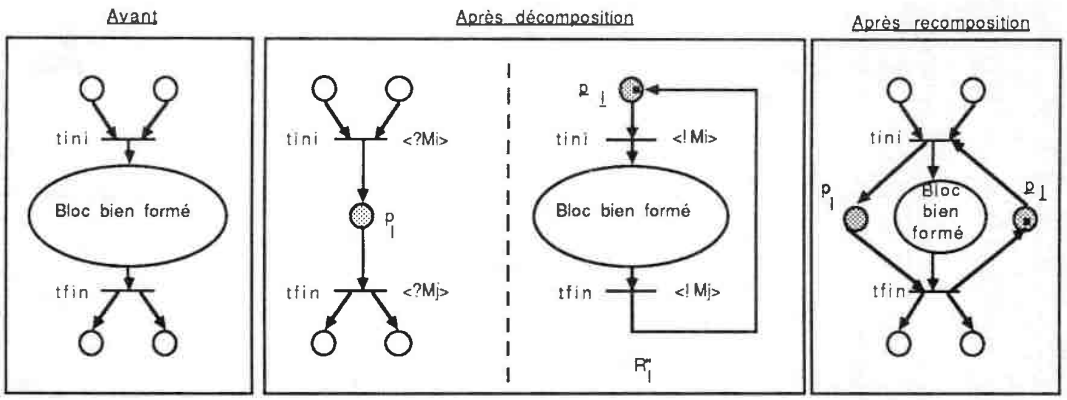
N B : *Chaque couple* $(p_j, \underline{p_j})$ *est donc placé en sortie d'une même transition et en entrée d'une même transition ou de deux transitions* (ti, $\underline{t_i}$) *franchissables en séquence. Les places* $\underline{p_j}$ *sont de ce fait des places*

implicites. Cette troisième transformation préserve donc aussi l'abstraction sur tout sous-ensemble de transitions $Td \subset T' - (\Sigma \underline{t_i}) = T$. *Elle complète les transformations définies dans les points (3) et (4) du corollaire précédent.*

Ceci permet de conclure que la règle RD13 ne modifie pas le comportement du réseau initial : Dans ce réseau tout comme dans le réseau transformé, le tir d'une transition $t_i \in T$ est conditionné par le même prédicat et entraine les mêmes conséquences. Par ailleurs, l'abstraction est préservé sur tout sous-ensemble de transitions inclu dans T.

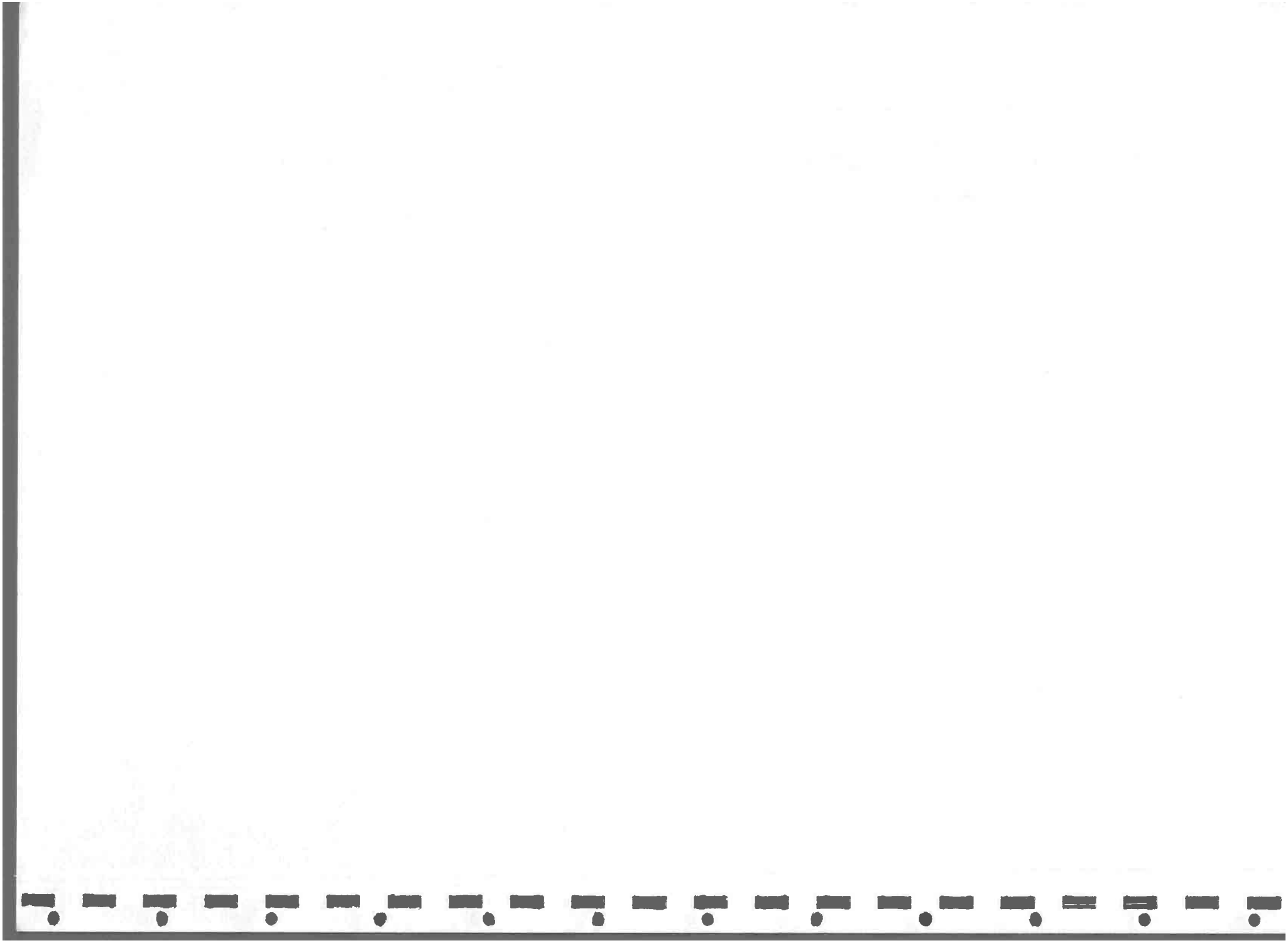
C.9 VALIDATION DE LA REGLE RD15

Le réseau global que l'on obtient par fusion des sous-réseaux résultant de l'application de la règle RD15 ne diffère du réseau initial que par la présence de deux places : P_I et $\underline{P_I}$ (cf. figure C.19(c)). La transition t_{ini} n'étant pas t-sensibilisée simultanément avec elle-même, si elle est franchie, elle ne pourra plus être franchie à nouveau tant que la transition t_{fin} n'aura pas été franchie et inversement. De ce fait, les places P_I et $\underline{P_I}$ sont des places implicites qui peuvent être supprimées sans modifier le comportement du réseau. Ce qui permet de conclure que la règle RD15 préserve le comportement externe du réseau initial.



(a) (b) (c)

Figure C.19



BIBLIOGRAPHIE

[**Abrial 79**] J. R. Abrial, S. A. Schuman
Nondéterministic system specification.
LNCS 70, Springer-Verlag, 1979 p. 34 à 50.

[**AFNOR 75**] Commission d'étude de la productique AFNOR
Une démarche productique pour l'entreprise.
Convention Automatique Productique, Paris 1985.

[**Alaiwan 85**] H. Alaiwan, J.M. Toudic
Recherche des sémi-flots, des verrous et des trappes dans les réseaux de Petri
TSI Vol 4, no. 1 ,1985, p. 103 à 112

[**Alanche 84**] P. Alanche, K. Benzakour, F. Dollé, P. Gillet, R. Rodrigues, R. Valette
PSI : A Petri Net based simulator for flexible manufacturing systems.
LNCS 188, Springer-Verlag, 1984, p. 1 à 14.

[**Alanche 86**] P. Alanche, P. Lhoste, G. Morel, M. Roche, M. Salim, P. Salvi
Application de la modélisation de la partie opérative à la structuration de la commande
Journées d'études organisées par l'AFCET Automatique: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, 1986, p. 1 à 13

[**Alla 84**] H. Alla, P. Ladet, F. Martin, J. Martinez, M. Silva
Modeling and validation of complex systems by coloured Petri nets . Application to a flexible manufacturing system.
LNCS 188, Springer-Verlag 1984, p. 13 à 31.

[**Alla 86**] H. Alla, P. Ladet, F. Martin, J. Martinez, M. Silva
Spécification de la commande des ateliers flexibles à l'aide de réseaux de Petri colorés.

Journées d'études organisées par l'AFCET Automatique: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, 1986, p. 85 à 95

[**Alla 87**] H. Alla
Réseaux de Petri colorés et réseaux de Petri continus : Application à l'étude des systèmes à événements discrets.
Thèse de Docteur d'Etat, Grenoble 1987.

[**André 81**] C. André
Systèmes à évolutions parallèles : modélisation par réseaux de Petri à capacité et analyse par abstraction.
Thèse de Docteur d'Etat es-sciences, Nice, 1981

[**Andrews 82**] G.R. Andrews
The distributed programming language SR. Mechanisms, design and implementation.
Software Practice an Experiences, Vol 12, 1982, p. 719 à 753.

[**Ayache 85**] J.M. Ayache, J.P. Courtiat, M. Diaz, G. Juanole
Utilisation des réseaux de Petri pour la modélisation et la validation des protocoles
TSI Vol 4, no. 1, 1985, p. 51 à 71

[**Blanchard 79**] M. Blanchard
Comprendre, maîtriser, et appliquer le GRAFCET.
Cepadus - éditions, 1979.

[**Brams 83**] G.W. Brams
Réseaux de Petri : Théorie et pratique
Edition Masson, 1983

[**Benmaïza 84**] M. Benmaïza
Le concept d'événement dans la spécification et la programmation

d'applications temps réel.
Thèse de Docteur Ingénieur, Nancy 1984.

[**Benzakour 86**] K. Benzakour, R. Valette
Simulation conjointe de commandes logiques et de systèmes mécaniques.
Journées d'études organisées par l'AFCET Automatique: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés discontinus complexes, Montpellier, 1986, p. 97 à 105.

[**Berthelot 78**] G. Berthelot
Vérification de réseaux de Petri
Thèse 3è cycle, Paris , 1978

[**Berthelot 85 (a)**] G. Berthelot
Transformations de réseaux de Petri
TSI, Vol 4, no. 1, 1985, p. 91 à 126.

[**Berthelot 85 (b)**] G. Berthelot
Analyse de processus parallèles par transformation de réseaux de Petri : Application à un protocole de réseau.
TSI, Vol 4, no. 1, 1985, p. 73 à 83.

[**Berthomieu 80**] B. Berthomieu
Analyse structurelle des réseaux de Petri, methodes et outils.
Thèse Docteur-ingénieur, Toulouse 80

[**Betourné 85**] C. Betourné, M. Filali, G. Padiou, A. Sayah
Construction de types répartis par migration du contrôle.
TSI, Vol 4, no. 6, 1985, p. 539 à 551.

[**Bochmann 77**] G.V. Bochmann, J. Gecsei
A unified method for specification and verification of protocols.
IFIP, Infor. Proc. 77, B. Gilchrist editor, North Holland pub. company, 1977, p. 229 à 234.

[**Bochmann 79**] G.V. Bochmann, J. Tankoano
Development and structure of an X25 implementation.
IEEE Trans. on Soft. Eng., Vol SE-5, no 5, 1979, p. 429 à 439.

[**Boudebous 88**] D. Boudebous
Thèse de 3ème cycle, Nancy, 1988 (à paraître)

[**Boussin 78**] J. Boussin
Synthesis and analysis of logic automation systems.
7th trienal world congres, Helsinki 1978 (Pergamon Press)

[**Calvez 84**] J.P. Calvez, Y. Thomas
Méthodologie de conception pour les systèmes complexes de commande en temps réel.
R.A.I.R.O. Automatique/Systems Analysis and control, Vol 18, no. 2, 1984 p. 251 à 266.

[**Campbell 74**] R. H. Campbell
The specification of processes synchronization by path expressions.
LNCS 16, Springer-Verlag, 1974, p. 89 à 102.

[**Chandy 86**] M. Chandy, J. Misra
An example of stepwise refinement of distributed programs : Quiescence detection.
ACM Trans. on prog. lang. and syst., Vol 8, no. 3, 1986, p 326 à 343.

[**Chen 81**] Z. C. Chen, C.A.R. Hoare
Partial correctness of communicating sequential processes.
Proc. 2nd Int. Conf. on distributed comput. syst. , Paris, 1981, p. 1 à 12

[**Chen 83**] Bo-Shoe Chen, Raymond T. Yeh
Formal specification and verification of distributed systems.
IEEE Trans. on soft. eng., Vol SE-9, no. 6, 1983, p.710 à 722.

[**Chrétienne 84**] P. Chrétienne
Les réseaux de Petri temporisés.
Thèse de Docteur d'Etat ès-sciences, Univers. P; et M. Curie , 1984

[**Chrétienne 85**] P. Chrétienne
Analyse des régimes transitoire et asymptotique d'un graphe d'événements temporisé.
TSI Vol 4, no. 1, 1985, p. 127 à 142.

[**Commoner 72**] F. Commoner
Deadlock in Petri Nets.
Applied data research Inc., Wakefield MASS, CA7206-2311, 1972

[**Courtiat 87**] J.P. Courtiat, P. Demminski, R. Groz, C. Jard
Estelle : un langage ISO pour les algorithmes distribués et les protocoles.
TSI Vol 6, no. 2, 1987, p. 89 à 102.

[**Courvoisier 83**] M. Courvoisier, R. Valette, J.M. Bigou, P. Esteban
Réseaux d'automates et ateliers flexibles.
Congrès AFCET Automatique : Productique et robotique intelligente", Besançon, 1983.

[**Derniame 83**] J.C. Derniame, A. Zakari
Spécification d'application de conduite d'un atelier flexible.
Convention informatique Latine, Barcelone, 1983.

[**Deschizeaux 82**] P. Deschizeaux
Temps et évènements dans les systèmes distribués de controle de procédé.
R.A.R.O. Automatique/Systems Analysis and Control, Vol 16, no.3, 1982, p. 259 à 274.

[**Dijkstra 75**] E. W. Dijkstra
Guarded commands, nondeterminacy and formal derivation of programs.
CACM Vol 18, no 8, 1975, p.453 à 457.

[**Dijkstra 76**] E. W. Dijkstra
A discipline of programming.
Prentice-Hall, Englewood Cliffs, N.J., 1976.

[**El Fazziki 85**] A. El Fazziki
Contribution à la structuration et à la programmation des applications de contrôle de procédés industriels.
Thèse de Docteur de 3ème cycle, Nancy, 1985

[**Florin 85**] G. Florin, S. Natkin
Les réseaux de Petri stochastiques.
TSI Vol 4, no. 1, 1985, p. 143 à 160.

[**Floyd 68**] R. W. Floyd
Assigning meanings to programs.
Proc. Amer. Math. Soc. Symposia in Applied Mathematics, 1968
Vol 19, p. 19 à 31.

[**Floyd 85**] C. Floyd
A systematic look at prototyping, in approaches to prototyping.
(Eds. R. Budde, K. Kuhlenkamp, L. Mathiasen, H. Züllinghoven), Springer-Verlag, 1984.

[**Genrich 81(a)**] H.J. Genrich, K. Lautenbach
System modelling with high-level Petri Nets.
Theoritical Computer Science 13, 1981, p. 109 à 136.

[**Genrich 81(b)**] H.J. Genrich, K. Lautenbach, P.S. Thiagarajan
Elements of general net theory in net theory and applications.
LNCS 84, Springer-Verlag, 1980, p. 21 à 164.

[**GREPA**] GRoupe Equipement de Production Automatisée réunie à l'ADEPA
Le GRAFCET, de nouveaux concepts.
Cepadus - éditions, 1985

[**Guyot 87**] J. Guyot
Vérification de la modélisation des traitements d'une application base de données.
Inforsid 87, Lyon, 1987, p. 65 à 86.

[**Hansen 78**] P.B. Hansen
Distributed Processes : A concurrent programming concept.
CACM Vol 21, no. 11, 1978, p. 934 à 941.

[**Herrmann 87**] F. Herrmann
CHORUS : Un environnement pour le developpement et l'exécution d'applications réparties.
TSI Vol 6, no. 2, 1987, p.162 à 165.

[**Hikita 85**] T. Hikita, K. Ishihata
A method of program transformation between variable sharing and message passing.
Software Practice and Experience. Vol 15(7), 1985, p. 677 à 692.

[**Hoare 69**] C.A.R. Hoare
An axiomatic basis for computer programming.
CACM Vol 12, no. 10, 1969, p. 576 à 583.

[**Hoare 78**] C.A.R. Hoare
Communicating Sequential Processes.
CACM Vol 21, no. 8, 1978, p. 666 à 677.

[**Hollinger 85**] D. Hollinger
Utilisation pratique des réseaux de Petri dans la conception des systèmes de production.
TSI Vol 4, no. 6, 1985, p. 509 à 522.

[**Hollinger 87**] D. Hollinger
Réseaux de Petri et Flavors pour la spécification et la simulation de

systèmes productiques.
Proc. des journées AFCET sur les langages orientés objets.
Bigre+Globule 48, 1986, p 206 à 215.

[**Hughes 83**] J.W. Hughes, M.S. Powell
DTL : A language for the design and implementation of concurrent programs as structured networks.
Software Practice and Experience, Vol 13, 1983, p. 1099 à 1112.

[**Jackson 78**] M.A. Jackson
Information systems : modeling, sequencing and transformations.
IEEE Proc. Int. Conf. on Soft. Eng. , 1978, p. 72 à 81.

[**Jensen 81**] K. Jensen
Coloured Petri Nets and invariant method.
Theoritical Computer Science 14, North. Holland publ. co., 1981, p 317 à 336.

[**Kahn 77**] G. Kahn, D.B. Mac Queen
Coroutines and networks of parallel processes.
IFIP, Infor. Proc. 77, B. Gilchrist editor, North Holland pub. company, 1977, p. 993 à 998.

[**Karp 69**] K.M. Karp, R.E. Miller
Parallel program schemata.
Journal of computer and system sciences, Vol 3, no. 2, 1969.

[**Keller 76**] R. M. Keller
Formal verification of parallel programs.
CACM Vol 19, no. 7, 1976, p. 371 à 384.

[**Kemmerer 85**] R.A. Kemmerer
Testing formal specification to detect design errors.
IEEE Trans. on soft. eng., Vol SE-11, no. 1, 1985

[**Kramer 81**] J. Kramer, J. Magee, M. sloman
Intertask communication primitives for distributed computer control systems.
Proc. 2nd Int. Conf. on distr. Comp. Syst., Paris, 1981, p. 1 à 8.

[**Kramer 82**] J. Kramer, J. Magee, M. Sloman
A software architecture for distributed computer control systems.
IFAC symp. on theory and application of digital control, New Dehli, 1982.

[**Kramer 83**] J. Kramer, J. Magee, M. Sloman, A. Lister
CONIC : An integrated approach to distributed computer control system.
IEEE Proc. , Vol 130, 1, 1983.

[**Ladet 82**] P. Ladet
Contribution à l'étude des systèmes informatiques répartis pour la commande des procédés industriels.
Thèse d'Etat es-sciences, Grenoble, 1982.

[**Lam 84**] S.S. Lam, A. U. Shankar
Protocol verification via projections.
IEEE Trans. on Soft. Eng. , Vol SE-10, no 4, 1984, p.325 à 342.

[**Lamport 78**] L. Lamport
Time, clocks, and the ordering of events in a distributed system.
CACM Vol 21, no 7, 1978, p .558 à 565.

[**Lamport 80**] L. Lamport
The "Hoare logic" of concurrent programs.
Acta informatica, Springer-Verlag, 14, 1980, p.21 à 37.

[**Lamport 83**] L. Lamport
Specifying concurrent program modules.
ACM trans. on prog. lang. and syst., Vol 5, no 2, 1983, p. 190 à 222.

[**Lee 85**] S. Lee
On executable models for rule-based prototyping.
Proc. 8th Int. Conf. on Soft. Eng., London , 1985.

[**Le Lann 83**] G. Le Lann
On real time distributed computing.
IFIP Information processing 83, R.E.A. Mason, Elsevier science pub., B.V. (North Holland), 1983, p. 190 à 222.

[**Le Lann 87**] G. Le Lann
Le projet Score : les systèmes informatiques répartis temps réel.
TSI Vol 6, no 2, p 173 à 178.

[**Lister 80**] A. Lister, J. Magee, M. Sloman, J. Kramer
Distributed process control systems : programming and configuring.
Rep. RR 80/12 Imp. College Londres, 1980.

[**Lonchamp 87**] J. Lonchamp
Conception des applications informatiques réparties en commande de procédés industriels : une démarche, des outils.
Thèse de Docteur d'Etat es-sciences, Nancy 1987.

[**Mahadevan 83**] S. Mahadevan, R.K. Shyamasundar
Correctness preserving transformations for distributed programs.
Proc. IFIP Congres Information Processing 1983, p. 307 à 313.

[**Manna 82**] Z. Manna , A. Pnueli
Verification of concurrent programs: a temporal proof system.
Proc. of the 4th school of advanced programming, Amsterdam, 1982.

[**Manna 84**] Z. Manna, P. Wolper
Synthesis of communicating processes from temporal logic specifications.
ACM Trans. on prog. lang. and syst., Vol 6, no. 1, 1984, p. 68-93.

[**Martin 87**] F. Martin
Méthodologie de modélisation et simulation de systèmes complexes décrits par réseaux de Petri colorés.
Thèse de Docteur de l'Institut National Polytechnique de Grenoble, 1987.

[**Memmi 83**] G. Memmi
Méthodes d'analyse des réseaux de Petri, réseaux à files, et applications aux systèmes temps réel.
Thèse d'Etat, Paris VI, 1983.

[**Milner 80**] R. Milner
A calculus of communicating systems .
LNCS 92, Springer-Verlag, 1980.

[**Minet 87**] P. Minet
MAP : un réseau local pour un environnement industriel automatisé.
TSI Vol 6, no 2, 1987, p. 182 à 186.

[**Misra 81**] J. Misra, K.M. Chandy
Proofs of networks of processes.
IEEE Trans. on Soft. Eng., Vol SE-7, no 4, 1981, p. 417 à 426.

[**Moalla 78**] M. Moalla, J. Pulou, J. Sifakis
Réseaux de Petri synchronisés.
R.A.I.R.O. Automatique/Systems Analysis and control, Vol 12, no. 2, 1978, p. 103 à 130.

[**Moalla 85**] M. Moalla
Réseaux de Petri interprétés et Grafcet.
TSI Vol 4, no. 1, 1985, p. 17 à 30.

[**Owicki 76**] S. Owicki, D. Gries
Verifying properties of parallel programs : axiomatic approach.
CACM Vol 19, no 5, 1976, p. 279 à 285.

[**Owicki 82**] S. Owicki, L. Lamport
Proving liveness properties of concurrent programs.
ACM Trans. on progr. lang. and syst. , Vol 4, no 3, 1982, p. 455 à 495.

[**Pascal 87**] J.C. Pascal, J.M. Pons, Y. Bandin, M. Courvoisier, R. Valette
Programmation modulaire d'une application distribuée dans le contexte du projet S.E.CO.I.A.
Journées d'étude S.E.E, Gif-sur-Yvette, 1987.

[**Perrin 85**] G. R. Perrin
La communication : Un outil pour la spécification, la construction et la vérification de systèmes parallèles.
Thèse de Docteur d'Etat ès-sciences, Nancy, 1985

[**Peterson 81**] J.L. Peterson
Petri Net theory and the modeling of systems.
Prentice-Hall, inc., Englewood Cliffs, N.J. 07632, 1981.

[**Pnueli 81**] Amir Pnueli
The temporal semantics of concurrent programs.
Theoretical computer science 13, 1981, p. 45 à 60.

[**Ramamoorthy 85**] C.V. Ramamoorthy, S.T. Dony, Y. Usuda
An implementation of an automated protocol synthesizer (APS) and its application to X.21 protocol.
IEEE Trans. on soft. eng., Vol SE-11, no 9, 1985, p. 886 à 909.

[**Ramamritham 83**] K. Ramamritham, R. Keller
Specification of synchronizing processes.
IEEE Trans. on Soft. Eng., Vol SE-9 no 6, 1983, p. 722 à 732.

[**Raynal 81**] M. Raynal
Contribution à l'étude de la coopération entre processus dans les langages et systèmes informatiques.

Thèse de Docteur d'Etat ès-sciences, Rennes I, 1981

[**Sifakis 79**] J. Sifakis
Contrôle des systèmes asynchrones : concepts, propriétés, analyse statique.
Thèse de Docteur d'Etat es-sciences, Grenoble, 1979.

[**Silva 85**] M. Silva, J. Martinez, P. Ladet, H. Alla
Generalized inverses and the calculation of symbolic invariants for coloured Petri nets.
TSI Vol 4, no. 1, 1985, p. 113 à 126.

[**Sintzoff 79**] M. Sintzoff
Principles for distributing programs.
LNCS 70, Springer-Verlag 1979, p. 337 à 347.

[**Tankoano 85**] J. Tankoano, J.C. Derniame
Démarche de structuration logique des systèmes informatiques répartis pour la commande des procédés industriels.
Rapport CRIN 85-R-R028, 1985.

[**Tankoano 86**] J. Tankoano, J.C. Derniame
Conception de systèmes répartis pour la commande des ateliers de production.
Rapport CRIN 86-R-117 1986.

[**Tankoano 87**] J. Tankoano, J.C. Derniame
Structure design of distributed systems using interpreted Petri Nets.
Rapport CRIN 87-R-37, 1987

[**Tardieu 86**] H. Tardieu, A. Rechfeld, R. Colletti
La méthode MERISE
Les éditions d'organisation, 1986

[**Thiel 85**] J.J. Thiel

FLEXI, système de développement d'applications répartis.
Document interne CERILOR, 1985.

[**Thomesse 80**] J.P. Thomesse
SYGARE : Une structuration pour la conception d'applications en temps réel et réparties.
Thèse de Docteur d'Etat es-sciences, Nancy, 1980.

[**TSI 85**] Numéro spécial réseaux de Petri.
Vol 4, no 1, 1985.

[**Valette 76**] R. Valette
Sur la description, l'analyse et la validation des systèmes de commande parallèles.
Thèse de Docteur d'Etat es-sciences, Toulouse, 1976.

[**Valette 79**] R. Valette
Analysis of Petri Nets by stepwise refinements.
Journal of computer and system sciences, Vol 18, no 1, 1979, p. 35 à 46.

[**Valette 82**] R. Valette, M. Courvoisier, J.M. Bigou
Les réseaux d'automates : analyse de la coopération.
Journées d'études S.E.E., Gif-sur-Yvette, 1982, p.733 à 740.

[**Valette 85 (a)**] R. Valette, M. Courvoisier, H. Demmou, J.M. Bigou, C. Desclaux
Putting Petri Nets to work for controlling flexible manufacturing systems.
Int. symp. on circuits and systems, Vol 2 of 3, 1985, p. 929 à 932.

[**Valette 85 (b)**] R. Valette, V. Thomas, S. Bachmann
SEDRIC : un simulateur à évènements discrets basé sur les réseaux de Petri.
APII Informatique industrielle, 19, 1985, p.423 à 436.

[**Zaffiropulo 80**] P. Zaffiropulo et al
Towards analyzing and synthesizing protocols.

IEEE Trans. Comm, Vol COM-28, 1980

[**Zakari 84**] A. Zakari
FLEXI : Langage de conception d'applications de conduite de procédés industriels.
Thèse de Docteur de 3è cycle, Nancy, 1984.

[**Zave 84**] P. Zave
The operationnal versus the conventional approach to software development.
CACM Vol. 27 no 2, 1984, p.104 à 118.

[**Zave 85**] P. Zave
A distributed alternative to finite-state-machine specifications.
ACM Trans. on prog. lang. and syst., Vol 7, no 1, 1985, p. 10 à 36.

[**Wirth 77**] N. Wirth
Toward a discipline of real-time programming.
CACM Vol 20, no 8, 1977, p. 577 à 583.

NOM DE L'ETUDIANT : TANKOANO Joachim

NATURE DE LA THESE : Doctorat d'Etat ès sciences

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 4 MARS 1988 n° 367

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

R. MAINARD

Résumé :

Dans ce mémoire, nous présentons une approche méthodique de conception rigoureuse des systèmes de commande répartis, qui s'appuie sur: une démarche globale, quatre modèles de représentation, et deux outils (PETRI-C et PETRI-S). Les caractéristiques essentielles de cette approche résident dans deux points principaux qui justifient son intérêt et font son originalité.

Le premier point concerne les règles de construction et de transformation mécanisables que nous proposons pour à la fois guider et discipliner le raisonnement dans les tâches de spécification et d'organisation. Ces règles s'appuient sur des techniques de raffinement et de décomposition de réseaux de Petri interprétés, qui permettent de simplifier, voire d'éliminer les étapes délicates de validation .

Le second point concerne la définition d'une approche de modélisation de l'environnement commandé et du flux d'activité, qui a permis la réalisation d'un outil d'aide au prototypage rapide, basé sur une simulation conjointe de la partie commande et de la partie opérative. Ce simulateur peut être utilisé soit comme un outil de mise au point, soit pour dialoguer avec l'utilisateur afin de clarifier ses besoins, soit pour évaluer les performances et dimmensionner l'atelier. Il est caractérisé par sa très grande facilité de mise en oeuvre (ce qui est un point fondamental pour le prototypage), et aussi et surtout par le fait qu'il permet d'éviter les doubles définitions (avec tous les risques d'incohérence que cela comporte) en exploitant au mieux la complémentarité qui existe entre les différentes vues caractéristiques du fonctionnement d'un atelier.

Mots clés : Systèmes de commande répartis, Méthode de conception, spécification, Organisation, Validation, Evaluation, Réseaux de Petri interprétés.