THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE

en

PRODUCTIQUE, AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE

par

Didier CRUETTE
Ingénieur IDN

METHODOLOGIE DE CONCEPTION DES SYSTEMES COMPLEXES A EVENEMENTS DISCRETS : APPLICATION A LA CONCEPTION ET LA VALIDATION HIERARCHISEE DE LA COMMANDE DE CELLULES FLEXIBLES DE PRODUCTION DANS L'INDUSTRIE MANUFACTURIERE

soutenue le 8 Février 1991 devant la Commission d'Examen

Membres du Jury :		
	Mr. R. VALETTE	Rapporteur
	Mr. C. VASSEUR	Rapporteur
	Mr. C. ANDRE	Examinateur
	Mr. J.P. BOUREY	Examinateur
	Mr. J.C. GENTINA	Examinateur, Directeur de Thèse
	Mr. M. KAPUSTA	Examinateur
	Mr. C. SIBERTIN-BLANC	Examinateur

AVANT-PROPOS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Automatique et d'Informatique Industrielle de l'INSTITUT INDUSTRIEL DU NORD sous la direction de Monsieur GENTINA, Professeur à l'IDN. Je tiens à le remercier de m'avoir accueilli dans son équipe de recherche.

Je tiens également à remercier
Monsieur VALETTE, Directeur de Recherche au LAAS-CNRS,
Monsieur VASSEUR, Professeur à l'Université de Lille I, Directeur de l'ENSAIT, d'avoir accepté d'être les rapporteurs de cette thèse,
et
Monsieur SIBERTIN-BLANC, Maître de Conférences à l'Université des Sciences Sociales de Toulouse,
Monsieur ANDRE, Professeur à l'Université de Nice,
Monsieur BOUREY, Maître de Conférences à l'IDN,
pour l'honneur qu'ils me font en participant à mon jury de thèse.

Je suis en outre très heureux de la présence à ce Jury de Monsieur KAPUSTA, Ingénieur de Recherche à la Division Recherches et Développements de la Télémécanique.

Je voudrai ici remercier tous les membres du Laboratoire d'Automatique et d'Informatique Industrielle de l'IDN, et en particulier Messieurs Jean-Pierre BOUREY, Emmanuel CASTELAIN et Etienne CRAYE pour le soutien et les conseils avisés qu'ils m'ont apportés, et la liberté qu'ils m'ont accordée lors de mon travail.

Enfin, je remercie Monsieur VANGREVENINGE pour la reprographie.

SOMMAIRE

CHAPITRE I
LE PROJET CASPAIM

CHAPITRE II
LE PROJET CASPAIM : LIMITES ET EVOLUTION PROPOSEE

CHAPITRE III
ANALYSE DU PROCEDE

CHAPITRE IV
GENERATION ASSISTEE DES GAMMES OPERATOIRES

CHAPITRE V
PRISE EN COMPTE DES PROTOCOLES D'ACCES ET ANALYSE DES BLOCAGES

INTRODUCTION GENERALE

Le concept d'atelier flexible s'est développé il y a une vingtaine d'années, dans le but d'améliorer la productivité.

Parallèlement, il est devenu indispensable de gérer au mieux l'exploitation des moyens de production tels que les robots, les machines à commande numérique principalement. En effet, les investissements induits, aussi bien en matériel qu'en formation du personnel et en maintenance sont tels que leur polyvalence intrinsèque doit être exploitée au maximum.

Nous considérons qu'une installation flexible se caractérise par les spécificités suivantes :

- un même produit manufacturé peut être réalisé selon des séquences d'opérations différentes. Le choix d'une séquence est réalisé, soit une fois pour toutes dès la conception, soit en cours d'exploitation, dynamiquement en fonction de la charge courante des postes par exemple,

- un même produit peut suivre différents trajets au sein du système de production pour atteindre les postes de transformation,

- un même poste de transformation peut réaliser indifféremment plusieurs opérations distinctes (polyvalence des machines). Changer de nature d'opération peut cependant induire des coûts de changement d'outils, de téléchargement de programme etc.

- plusieurs produits indépendants entre eux peuvent cohabiter au sein du même système de production, et donc interagir du point de vue des délais de réalisation.

La conception d'une unité flexible nécessite de ce fait la mise en facteur des compétences d'un nombre important d'intervenants.

L'équipe chargée de la conception d'une unité de production flexible est confrontée à une triple difficulté que nous exprimons successivement sous la forme d'interrogations :

(i) quels sont les modèles et les méthodes permettant de décrire de manière exhaustive les possibilités de séquencement d'opérations pour un produit, et de choisir les séquencements admissibles,

(ii) quels sont les modèles et les méthodes permettant de construire la

commande des machines existantes ou futures,

(iii) quelle est la méthode permettant de réaliser l'adéquation entre les opérations qui doivent être réalisées pour un produit donné, et les moyens de production disponibles ; quels sont les moyens de validation, et comment remettre en cause les points (i) et (ii) si cette adéquation est impossible.

L'objectif du Laboratoire d'Automatique et d'Informatique Industrielle (L.A.I.I.) de l'Institut Industriel du Nord (I.D.N.) est de contribuer au développement d'une méthodologie globale qui fournit une assistance dans la démarche de conception des systèmes flexibles de production, depuis la spécification jusqu'à l'implantation finale. Cette démarche constitue le projet C.A.S.P.A.I.M. (Conception Assistée de Systèmes de Production Automatisée en Industrie Manufacturière).

La méthodologie intègre de manière rigoureuse et structurée les résultats d'outils élaborés en amont de la démarche, tel que des méthodes d'élaboration des gammes d'assemblage (BOU 84, BOU 87, BOU 90a, CHA 88, BOU 90b, BOU 90c, CAM 89, KAP 88).

Les moyens de production sont alors pris en compte pour créer le modèle de la commande (BOU 88) qui sera finalement évaluée (CAS 87) et implantée (CRA 89). Les problèmes décisionnels et stratégiques sont identifiés de manière systématique afin d'assister le concepteur dans sa démarche.

Le projet CASPAIM s'inscrit dans le cadre ambitieux de gestion complète du cycle de vie d'un automatisme, à savoir, conception, validation, exploitation et maintenance. Ces travaux s'inscrivent donc de fait dans l'esprit du Génie Logiciel appliqué aux automatismes.

Compte tenu de l'ampleur du travail du concepteur et du caractère évolutif des spécifications, il est apparu comme indispensable de valider le plus tôt et le plus souvent possible les choix réalisés. De plus, une solution acceptable n'est jamais considérée comme la meilleure ou comme définitive. C'est pourquoi, la gestion des variantes de conception et des configurations nous semble d'une grande importance. La simulation, qui sera réalisée à plusieurs niveaux de la démarche de conception, vient en complément dans le but d'évaluer la ou les solutions retenues.

Il s'agit donc d'un véritable outil d'assistance à la conception des systèmes flexibles.

Nous nous sommes attachés dans un premier temps à tenter d'appliquer la méthodologie sur des exemples industriels complexes. Certains d'entre eux se sont révélés impossibles à prendre en compte, et ont ainsi contribué à mieux

préciser les limites d'application de la méthodologie de conception proposée par le LAII.

Ces difficultés nous ont conduits à proposer une refonte complète de la méthodologie CASPAIM, pour aboutir à une formulation nouvelle sous le label CASPAIM-2.

Le premier chapitre de ce mémoire dresse un constat qui a été à l'origine de nos travaux concernant les limitations intrinsèques de la méthodologie CASPAIM.

Le second chapitre précise les raisons de ces limites ainsi que le contexte et la philosophie de l'extension que nous proposons dans le cadre de ce mémoire.

Nous avons été amenés à séparer la spécification des moyens de production de celle du processus de fabrication des produits.

Au chapitre III, nous abordons l'étude et l'analyse du système de production selon le point de vue de la commande, indépendamment de son utilisation effective dans le cadre d'une production particulière. Le modèle arborescent obtenu à l'issue de cette modélisation précise, entre autre, les possibilités de transfert des pièces.

Le chapitre IV détaille, sur la base d'exemples, la modélisation du processus de fabrication d'une pièce ou d'un ensemble de pièces liées entre elles. Le modèle purement fonctionnel ainsi obtenu est ensuite enrichi et validé progressivement en prenant en compte le modèle arborescent des moyens de production et de transport.

Le chapitre V précise la manière dont nous prenons en compte la gestion des ressources (machines, transports) afin d'obtenir un modèle complet de la Partie Commande d'un atelier flexible. Une première approche de détection de blocages est également présentée, dans le but d'assister la validation du comportement dynamique.

L'ensemble de ce mémoire s'appuie sur un exemple simple et original qui permet d'illustrer les concepts et méthodes présentées.

CHAPITRE I

LE PROJET CASPAIM

INTRODUCTION

Le projet CASPAIM, dont l'objectif fut à l'origine d'élaborer une méthodologie de conception d'ateliers flexibles, a conduit le LAII à mettre en place une chaîne complète de logiciels cohérents, de la spécification à la génération de modèles implantables. Ce projet est à la base de notre travail, nous proposons pour cette raison d'en décrire les principaux éléments.

La démarche de conception CASPAIM est détaillée dans une première partie de ce chapitre. Nous mettrons ensuite en lumière un certain nombre de limitations qui nous ont conduits à envisager une extension et une refonte de la démarche présentée au chapitre II.

I.1. La démarche et les modèles

I.1.1. Le produit de la démarche (PO, PC, NH)

Depuis 1979, l'élaboration des modèles de représentation et de commande du système de pilotage des unités s'est peu à peu enrichie au LAII afin de prendre en compte :

- le caractère polyvalent des machines de production et de transport,

- le caractère flexible de la production.

Le produit de la démarche de conception des systèmes de production flexibles utilisé dans le projet de Conception Assistée de Systèmes de Production Automatisés dans l'Industrie Manufacturière (CASPAIM) est le suivant :

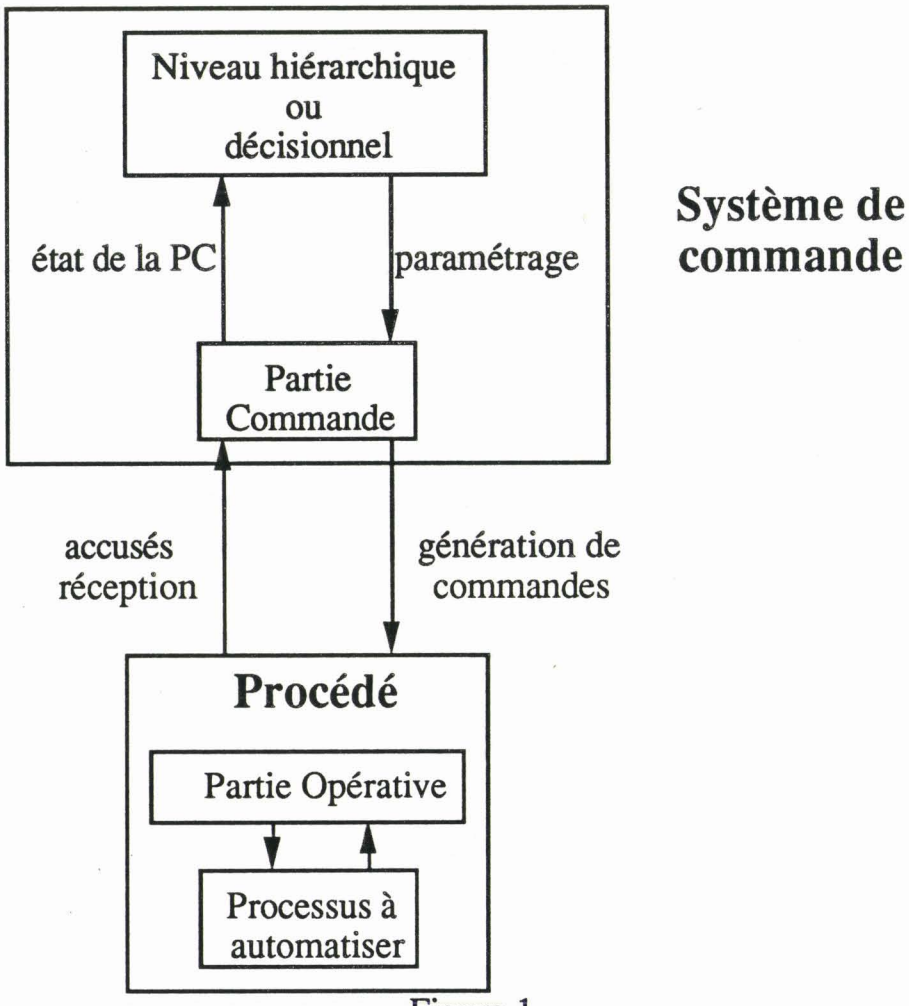


Figure 1

I.1.1.1. La Partie Commande (BOU 88a)

La **Partie Commande** est chargée d'assurer le séquencement des opérations sur les postes de transformation, ainsi que la synchronisation entre les postes et les moyens de transport. La Partie Commande intègre donc un double aspect :

- la gestion des commandes à destination des organes physiques de transformation,

- la gestion du séquencement des opérations qui doivent être réalisées sur chaque pièce produite.

Elle est modélisée par réseaux de Petri structurés adaptatifs et colorés.

Le modèle de base, les Réseaux de Petri, permet d'exprimer le parallélisme et les contraintes de séquencement et d'exclusion des opérations effectuées.

Le caractère **structuré** a été introduit afin d'exploiter un nombre restreint de modules type reliés entre eux, et dont on a validé au préalable les propriétés (BOU 88b, BOU 87, GEN 87).

Le graphe de commande est construit en assemblant un ensemble de **graphes de processus** reliés entre eux par des **graphes de liaison.**

Un **processus** est défini comme un enchaînement de tâches et peut être décrit par un réseau de Petri structuré au moyen des trois structures suivantes :

- l'action,

- l'alternative,

- la répétitive.

Un graphe de processus est associé à chaque lieu de transformation des produits, à savoir :

- les machines d'usinage avec ou sans tampon d'entrée/sortie,

- les machines d'assemblage avec ou sans contrainte d'antériorité,

- les palettisations simples ou multiples,

- les désassemblages et dépalettisations,

- les transferts unitaires de produits.

Les **graphes de liaison** représentent les interactions entre les divers processus. Elles peuvent être de trois types :

- liaison de synchronisation,

- liaison d'exclusion mutuelle,

- liaison producteur/consommateur.

Le caractère **adaptatif** du modèle permet de traduire le mécanisme de changement de mode de marche grâce à des places d'interface avec le Niveau Hiérarchique. Le marquage de ces places permet de déconnecter ou de geler certaines portions du réseau de la PC dans le but de gérer les modes de marche (GEN 87, BOU 87).

La **coloration** du modèle permet une agrégation importante du réseau du fait de la polyvalence des machines de transformation ou de transport.

La couleur est ici utilisée pour caractériser les différents types d'objets produits, généralement différenciés par des gammes spécifiques.

La Partie Commande est générée à partir d'un modèle agrégé appelé Prégraphe. Il s'agit d'un RdP coloré pour lequel :

- un jeton coloré représente un objet dans un état d'avancement donné dans son processus de fabrication,

- les transitions représentent des actions effectuées sur les objets (pièces, palettes),

- les places représentent des lieux physiques sur lesquels s'effectuent des opérations de transformation ou d'assemblage d'objets.

Ce modèle initial Prégraphe (KAP 88) est lui-même la synthèse d'un ensemble de règles de production.

Une règle de production traduit une opération élémentaire de la forme :

Prémisses => Conséquents

Les prémisses et les conséquents sont constitués d'une conjonction de prédicats qui sont de la forme :

(présence objet lieu)

qui signifie la présence de l'objet sur le lieu.

La forme générale des règles de production associées aux opérations de la gamme opératoire est donc la suivante :

(présence p1 P1) et (présence p2 P2) et (présence pn Pn)

=>

(présence q1 Q1) et (présence q2 Q2) et (présence qk Qk)

Le Prégraphe est la synthèse de l'ensemble des règles de production, obtenue en associant une place RdP par lieu rencontré, une marque colorée par objet pi, et une transition par règle de production. Les transitions ayant mêmes places d'entrée et de sortie sont agrégées. Aux arcs amont et aval de

cette place sont associés des domaines de couleur correspondant à l'ensemble des différents types de jetons transitant par les transitions de départ.

Les figures 2 et 3 présentent deux exemples de développement du graphe de commande sur la base du Prégraphe.

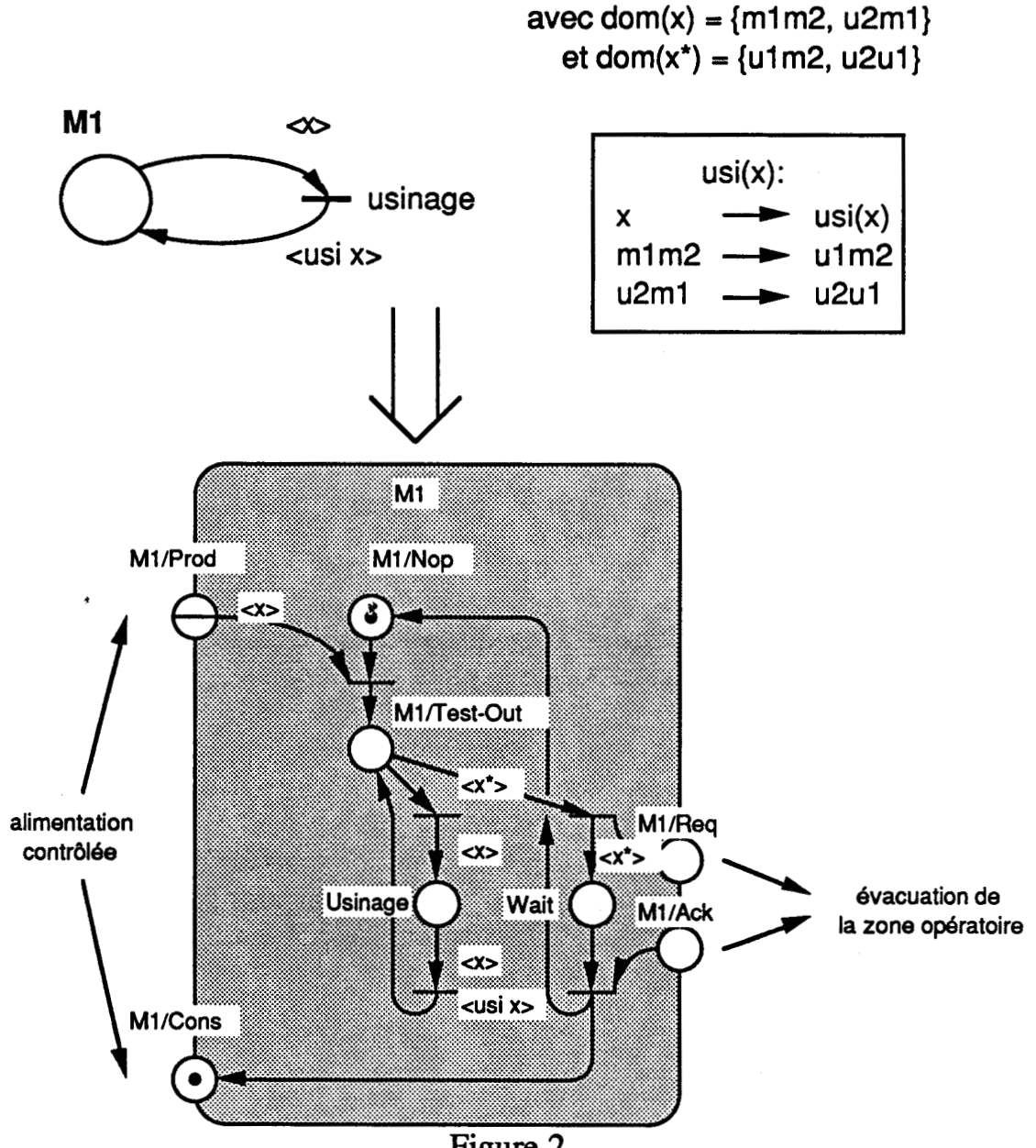


Figure 2

La figure 2 présente une fonction d'usinage sur une machine M1.

Les pièces de type m1m2 et u1m2 doivent subir un usinage sur le lieu M1. Les règles deux règles de production de départ sont les suivantes :

(présence m1m2 M1) => (présence u1m2 M1)

et

(présence u2m1 M1) => (présence u2u1 M1)

Elles ont été agrégées en une seule transition usinage dont les arcs amont et aval sont reliés à la place M1.

La place M1 modélise une machine d'usinage .Le développement du Prégraphe consiste à lui associer un graphe de commande qui assure l'alimentation de la machine en pièces de type FIFO, le séquencement des opérations effectuées sur la machine, puis l'évacuation de la machine.

Le graphe structuré est composé :

- d'une synchronisation par liaison Producteur/Consommateur qui gère l'alimentation en pièces,

- une place d'attente pour la fonction d'usinage (Usinage),

- une place d'attente pour l'évacuation (Wait),

- et une liaison par Req/Ack pour l'évacuation de la zone opératoire.

Les deux dernières liaisons sont distribuées par une structure de type répétitive multiple articulée autour de la place M1/Test-Out.

La figure 3 décrit la fonction de transfert entre deux postes M1 et M2. Le Prégraphe est développé en un processus M1->M2 qui est synchronisé avec les processus M1 et M2 via leur liaison Prod/Cons et Req/Ack respective. Le processus M1->M2 réalise en séquence le début de transfert (place Trsf-Start) qui correspond à l'évacuation de la zone de déchargement de la machine M1, puis la fin de transfert (place Trsf-End) qui correspond au chargement de la machine M2.

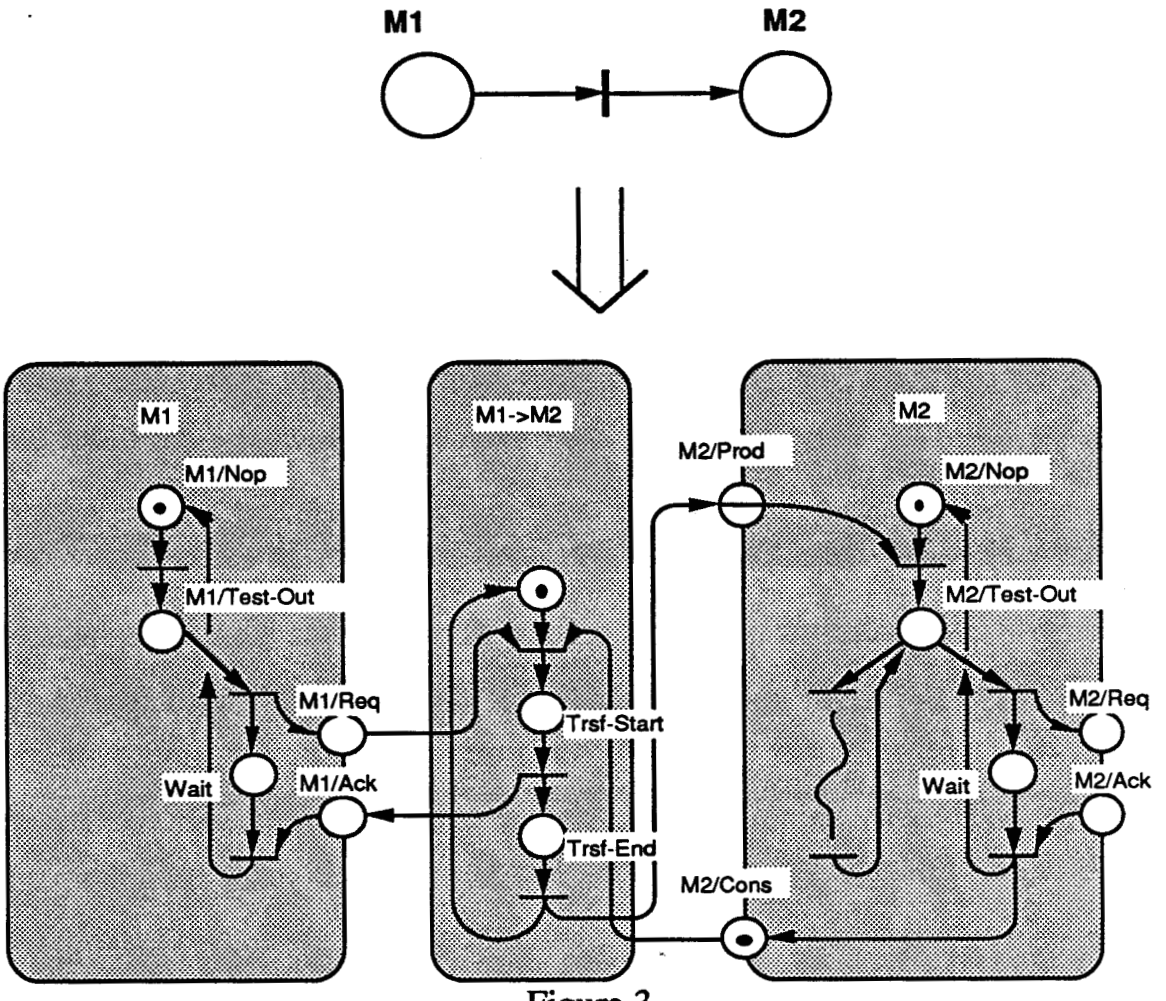


Figure 3

I.1.1.2. Le Niveau Hiérarchique (CRA 89)

Du fait de la flexibilité induite par :

- la possibilité de permutations des opérations pour un produit donné,

- les différentes possibilités de routage et de choix de moyens de transport,

le modèle de la Partie Commande est non autonome et non déterministe. Nous avons donc recours à un niveau décisionnel, appelé Niveau Hiérarchique (NH), dont le rôle est d'assurer le paramétrage de la PC afin de tenir compte des stratégies de production issues de niveaux de planification plus élevés.

Différentes approches ont été étudiées pour la modélisation du NH :

- les réseaux de Petri, homogène avec le modèle de la PC,

- les règles de production.

Bien que la transposition de règles de production en transition RdP soit immédiate, les règles de production présentent les avantages suivants :

- le traitement des incidents et des diagnostics est facilité par une approche de type Intelligence Artificielle,

- les modifications hors ligne ou en ligne du contrôle des stratégies sont aisées.

La démarche retenue a donc été basée sur un NH par règles de production interprétées par chaînage avant.

I.1.1.3. Le procédé

Le procédé recouvre l'ensemble des objets matériels auxquels s'adressent les ordres de la Partie Commande. Il s'agit principalement :

- des machines, outils et moyens de l'unité de production, dont la durée des opérations est synthétisée par une temporisation dans la PC,

- des produits manufacturés, dont les jetons colorés constituent une image dans la PC.

La capacité des files d'attente en amont des postes de transformation est modélisée par le mécanisme de liaison type Producteur/Consommateur.

Cependant, cette représentation simple du Procédé dans la PC peut s'avérer insuffisante lors de :

- l'élaboration de statistiques de fonctionnement,

- la simulation des défaillances et anomalies,

- la vérification de la bonne conception du système de commande vis à vis de son effet sur le Procédé.

Dans ces conditions, il est alors nécessaire de disposer d'un modèle spécifique du Procédé, en parallèle avec l'image qu'en a la PC.

La modélisation par langage objet (CAS 87) permet d'exploiter l'aspect modulaire et générique des entités matérielles considérées, et de faciliter les phases de prototypage d'architectures.

Les modèles associés à la Partie Commande, au Niveau Hiérarchique et au

Procédé sont élaborés au cours de la démarche de conception que nous allons maintenant présenter.

I.1.2. La démarche de conception

Les modèles sont construits selon la démarche séquentielle suivante :

Description formelle des processus de fabrication
analyse de cohérence et de complétude sous forme de règles de production

puis

Construction de la description fonctionnelle de l'unité de production
appelé Prégraphe

puis

Structuration : génération de graphe de commande développé RdPSAC
de la PC à parallélisme maximal,
en tenant compte de la nature et de la capacité des zones opératoires

puis

Réduction du parallélisme : exclusions mutuelles et synchronisations
traduisant le partage des ressources de transport et de stockage,
et la synchronisation des fonctions de manutention

puis

Construction de Niveau Hiérarchique : détection des conflits
et indéterminismes, écriture des règles de gestion des ressources partagées

puis

Modélisation du Procédé : temporisation des actions,
dimensionnement des zones de stockage, choix des séquences d'entrée

puis

simulation des modèles
PC
NH
Procédé

puis

Répartition, implantation

Ce schéma directeur illustre la volonté de prendre en compte de manière progressive les spécifications issues du cahier des charges, ainsi que la possibilité de retours arrière à chaque étape du développement d'un projet.

Cette démarche est satisfaisante dans des cas industriels peu complexes tels qu'ils ont été présentés dans (CAS 87, BOU 88, KAP 88, CRA 89). Cependant, de nombreux exemples n'ont pu être pris en compte.

I.2. Limites de la démarche

Nous allons détailler quelques exemples qui n'entrent pas dans le cadre de la démarche CASPAIM, puis nous analyserons les raisons de ces limitations.

I.2.1. Exemples révélateurs des limitations

- Le premier exemple concerne les processus de transfert. La démarche conduit à créer un processus de transfert pour chaque fonction de transport d'une pièce d'un poste de départ à un poste destination. Lorsqu'il existe un nombre important de combinaisons de transfert entre plusieurs postes, il est clair que l'on aboutit rapidement à une explosion combinatoire des processus de transfert.

- La phase de structuration génère une zone de stockage en amont de chaque zone opératoire. Une zone de stockage partagée entre plusieurs postes de transformation distincts et indépendants est difficile à gérer.

- Un moyen de transport disposant d'une capacité de plusieurs emplacements de stockage de pièces est impossible à gérer. L'exemple de gestion d'un stockeur rotatif pose en effet difficulté. La solution présentée dans (KAP 88) en est une illustration. En effet, lors de la phase de réduction du parallélisme du graphe de commande, il n'est pas possible de prendre en compte des contraintes autres que la mutuelle exclusion entre transferts (un robot manipule une pièce à la fois), ou la synchronisation permanente entre transferts (un carrousel réalise les transferts synchronisés de n pièces et pas n-1 ou n-2).

- Un assemblage de deux pièces peut conduire à un blocage par famine en attente d'un des composants. En effet, l'opération d'assemblage sur la machine d'assemblage est déclenchée alors que l'un des composants peut ne pas être disponible.

- Le traitement de pièces liées logiquement entre elles (les composants

d'une carte électronique donnée par exemple) est très lourd voire impossible à gérer avec l'outil RdP coloré.

- Les situations bloquantes ne sont détectées qu'en phase de simulation du graphe de commande complet. De plus, l'horizon de simulation doit être très important pour obtenir des résultats significatifs.

- La coloration ne permet pas de manipuler plus d'une seule information par pièce gérée, indépendamment de sa localisation dans le RdP.

Voyons maintenant les hypothèses qui sont à l'origine de ces limites.

I.2.2. Hypothèses à l'origine des limitations

I.2.2.1. Le Prégraphe est un modèle trop agrégé

Le modèle Prégraphe est un RdP coloré (BOU 87, BOU 88b). A chaque place du Prégraphe est associé un graphe de processus qui modélise à la fois :

- la zone tampon d'attente en amont du poste de transformation (place Prod),

- le processus de transformation de la pièce qui se trouve sur le poste.

A une transition faisant intervenir une place du Prégraphe est associé :

- soit un traitement (usinage) sur le poste de transformation,

- soit une fonction de repositionnement (retournement) de la pièce sur le poste.

A une transition faisant intervenir deux places du Prégraphe est associé :

- soit un processus de transfert de pièce entre les deux postes associés aux places,

- soit une fonction de palettisation ou une fonction d'assemblage avec contrainte d'antériorité.

La sémantique associée au modèle Prégraphe est donc fort complexe et peut être sujette à confusion et à de multiples interprétations.

La phase de structuration consiste à parcourir le modèle Prégraphe, et à en déduire, de par la nature des liaisons associées aux places, quelle est la nature du traitement réalisé sur les pièces et les objets en général. Deux

transitions ayant la même topologie peuvent, de fait, avoir plusieurs interprétations en terme de traitement sur les pièces.

Ainsi, un assemblage sans contrainte d'antériorité, qui consiste à alimenter le poste d'assemblage avec les produits à assembler sans imposer de séquencement dans les arrivées, est interprété comme deux transferts d'alimentation du poste d'assemblage distincts, et donc en exclusion mutuelle.

La sémantique du modèle Prégraphe est donc floue et sujette à interprétation.

I.2.2.2. Prise en compte éparpillée du Procédé

Les données issues de la spécification des moyens de production sont prises en compte dans plusieurs phases de la démarche :

- dans la définition des règles de production permettant de construire le Prégraphe (lieux de transformation),

- dans le dimensionnement des files d'attente (capacité des files d'attente en palettes ou en pièces),

- lors de la temporisation des transitions,

- lors de la phase de restriction du parallémisme : un robot de manipulation est assimilé à une ressource critique.

Cependant, il n'existe pas de modèle spécifique et autonome du Procédé qui permettrait de vérifier la compatibilité mutuelle des spécifications qui concernent le Procédé, ou de réutiliser une configuration matérielle existante sur un nouveau projet de synthèse de commande.

Nous verrons au chapitre II que l'analyse fine du Procédé nous a permis de couvrir l'ensemble des limitations énoncées précédemment.

I.2.2.3. Phases de conception par essai-erreur délicates à gérer

Le projet CASPAIM se place dans le contexte des méthodologies de conception. Il s'agit donc d'assister le concepteur dans sa démarche, mais aussi de gérer les erreurs de conception. Dans ce cadre, il est indispensable que les spécifications soient validées. De plus, il est souhaitable qu'une modification mineure (architecture matérielle, processus de fabrication) soit de portée limitée, et qu'elle n'entraîne pas la remise en cause de l'ensemble des résultats validés au moment de l'erreur.

De par la nature du modèle Prégraphe, il est clair qu'une modification de l'architecture de transport par exemple entraîne une redéfinition complète des gammes opératoires et de l'architecture de commande. De plus, l'ensemble des paramètres (capacités, types des machines) doit être saisi à nouveau, alors que seule une portion du modèle Prégraphe est modifiée.

La gestion des configurations et des reprises sur erreur de conception sont donc très délicates et rapidement fastidieuses sur un projet de taille importante.

Cela constitue un handicap important pour un outil dont l'objectif est d'assister le plus efficacement possible le concepteur dans sa démarche de spécification. Nous verrons au chapitre IV que la démarche que nous proposons limite au maximum la portée des erreurs de spécification.

CONCLUSION

La présentation succincte de la démarche CASPAIM initialement mise au point par le LAII nous a permis tout à la fois de préciser le contexte de notre travail ainsi que les limitations de l'approche antérieure qui sont, de fait, à la source de notre recherche.

Dans ce sens, nous serons amenés au cours de ce mémoire, à proposer une approche plus large de la problématique d'analyse et de synthèse de la commande de cellules flexibles. La complexité toujours croissante des systèmes de production manufacturiers justifie pleinement une telle préoccupation qui ne pouvait être aussi évidente lors des premières études qui ont conduit le LAII à la version de CASPAIM présentée ici.

CHAPITRE II

Le projet CASPAIM : limites et évolution proposée

Introduction

La finalité de la méthodologie CASPAIM a été dès le départ, de fournir les moyens nécessaires à la conception, l'analyse et l'évaluation de la partie commande de cellules flexibles. Cette méthodologie permet également d'évaluer comparativement différentes solutions matérielles potentielles.

La démarche proposée se veut progressive et entièrement assistée. Elle propose à la fois :

- un outil méthodologique de conception de la commande de cellules flexibles

- un outil d'analyse de configurations matérielles existantes ou futures

- un outil d'évaluation de solutions complètes.

Cependant l'étude de cas complexes nous a montré les limites de la démarche telle qu'elle est exposée au chapitre I. Nous avons vu que, les cas suivants se sont révélés impossibles à modéliser; il s'agit notamment :

- des systèmes de transport évolués tels que des chariots comportant plusieurs emplacements,

- des stockages fixes complexes partagés par plusieurs moyens de production,

- des stockages mobiles complexes tels que stockeurs rotatifs, plateaux tournants (carrousels),...

- du traitement de lots de pièces,

- du traitement de pièces liées entre elles logiquement : par exemple des composants destinés à une même carte électronique.

L'analyse de ces divers points d'achoppement a permis de dégager les points faibles de la démarche :

- la description fonctionnelle (Prégraphe) est trop agrégée,

- le procédé est pris en compte de manière insuffisante,

- la description fonctionnelle est trop orientée : en effet la prise en compte des lieux de travail (opérationnel) dans la description fonctionnelle est privilégiée; les fonctions de transport ne sont considérées qu'en second lieu.

Une description fonctionnelle doit être, par nature, indépendante des moyens opérationnels nécessaires à la réalisation des fonctions, qu'il s'agisse aussi bien des moyens de transport que des lieux physiques de transformation, et en les considérant avec la même importance.

Commençons par détailler le domaine d'application de la démarche de conception dans sa nouvelle version que nous appelons désormais CASPAIM-2, ainsi qu'une extension possible du champ d'application. Puis, sur un exemple, nous illustrerons la démarche de conception proposée. Cette démarche consiste à prendre en compte de façon progressive et contrôlée l'aspect opérationnel du système réel sur la base d'une description fonctionnelle orientée produit.

II.1. Domaines d'application de la méthodologie

Par vocation, la méthodologie élaborée par le L.A.I.I. s'adresse aux productions manufacturières et flexibles. Il s'agit essentiellement de pièces de mécanique, d'électronique ou autre, qui peuvent être manipulées par toutes sortes de robots et préhenseurs.

Les moyens de production du Procédé sont conventionnels : centres d'usinage, de fraisage, de tournage, de soudage, d'assemblage etc...

Les moyens de transport le sont également : palettes, robots, chariots filoguidés ou non, convoyeurs à bandes débrayables etc...

Nous verrons au paragraphe II.2. que nous avons veillé à prendre le problème du transport dans sa globalité afin de pouvoir gérer la généralité des moyens de transport, en particulier des systèmes complexes tels que ascenseur, stockeur rotatif etc...

II.1.1. Champ d'application

Il s'avère que la méthodologie de conception doit permettre de prendre en compte tout système pour lequel :

- le produit manipulé peut être inscrit dans un volume indéformable : produit solide uniquement,

- toute transformation est réalisée en un temps limité et ne concerne qu'un nombre fini d'objets constituant le produit,

- les opérations relatives à un produit sont supposées exclusives entre elles,

- un système de commande maîtrise le lancement (par une commande) et la fin d'une opération (par une valeur d'une variable d'état du produit). En particulier aucune transformation ne peut être déclenchée de manière intempestive vis à vis du système de commande.

II.1.2. Cas ne relevant pas de notre champ d'application

Sont exclus, en particulier, les procédés pour lesquels :

- les produits subissent des transformations chimiques ou physiques non maîtrisées,

- les produits liquides et/ou gazeux, ou les solides qui le deviennent au cours d'une transformation,

ex: fabrication de culasse à partir de fonte liquide
liquéfaction de copeaux en vue d'un recyclage de matière

- il existe une transformation dont on ne maîtrise ni le déclenchement ni l'arrêt (corrosion, altération due au vieillissement).

De manière plus formelle, les hypothèses sur les produits et le procédé sont les suivantes :

- les objets manipulés sont indéformables (solides),

- les états des objets sont identifiables et discrétisables,

- les changements d'état des objets (position dans l'espace, degré d'achèvement...) sont la résultante d'une commande transmise à un moyen de production.

De manière plus synthétique nous supposons que :

- le système de commande maîtrise intégralement le système commandé (produit et procédé de fabrication),

- le système est à états discrets et commandes discrètes.

II.1.3. Les exceptions tolérées

Un produit déformable ne devra pas constituer un produit intermédiaire de fabrication. Il doit être considéré comme une ressource nécessaire à la production d'un produit solide. De plus, la quantité nécessaire par produit

solide doit être finie et isolable dans l'espace.

Par exemple, un pistolet à peinture manipule un produit liquide, donc à priori les volumes utilisés n'entrent pas dans le domaine d'application de la méthodologie. Cependant, si par un moyen quelconque il est possible d'isoler le volume de peinture nécessaire à la mise en couleur d'une pièce, alors on peut considérer la réserve de peinture comme un ensemble discret d'échantillons distincts.

Par cet artifice il devient possible d'assimiler le stock de peinture à un ensemble discret de produit, afin d'en assurer la gestion et le contrôle (par exemple garantir un stock suffisant pour la mise en couleur d'une pièce). Nous constatons que dans ce cas précis de produit déformable, il est possible de discrétiser l'état du stock. Le volume du stock est pris en compte avant une opération de peinture, puis après. Ainsi les valeurs prises par le volume du stock sont discrètes.

Les seules exceptions acceptables au caractère indéformable du produit concernent ainsi les adjonctions de produits nécessaires à la fabrication, que l'on assimile à un produit discret.

Le cas typique de procédé non modélisable est la fabrication de lingots métalliques à partir de métal fondu. Dans ce cas c'est le produit lui-même qui est liquide puis solide. La difficulté de la commande de ce type de process hybride (continu/discret) vient du fait que la production de métal et la production de lingots ne sont pas commandables de manière discrète : il existe un phénomène d'inertie important dans le cas d'un haut fourneau par exemple. Il est impossible de faire varier instantanément le débit de production de tels process.

A l'inverse c'est une des caractéristiques des systèmes à événements discrets : le système de commande doit pouvoir maîtriser la production du procédé commandé, en particulier en imposant des évolutions significatives de production, ce qui ne s'apparente en rien à une variation de consigne.

Nous constatons que dans le cas de l'atelier de peinture précédent, on se contente de gérer les stocks de peinture en tant que produit utilisable immédiatement (variation instantanée de la quantité utilisée). Par contre la gestion de la production de la peinture elle-même ne serait pas réalisable dans l'état actuel du projet CASPAIM. Il s'agit en effet d'une production hybride (continu/discret), dont la partie continue sort du cadre de notre champ d'application.

Il est à noter que les hypothèses décrites précédemment sont peu contraignantes. Nous verrons Chapitre III qu'elles permettent de considérer le produit et les moyens de production exactement de la même façon. Ainsi il

devient possible de gérer l'approvisionnement d'outils consommables (forets, lames de scies) de la même manière qu'un produit intermédiaire ou fini.

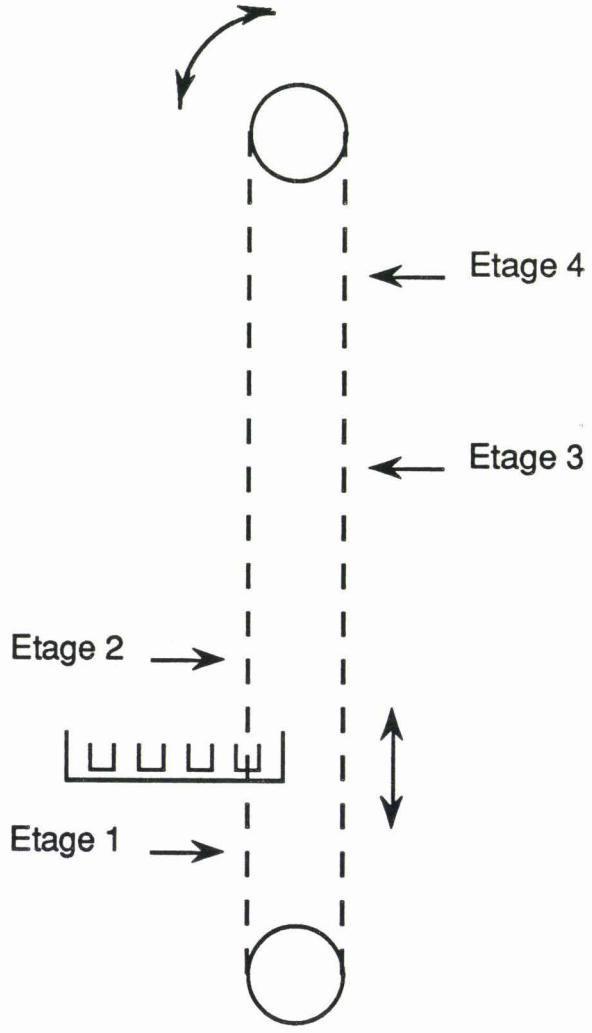
De plus les moyens de production et de manipulation sont analysés selon une méthodologie simple et originale (Chapitre III) qui autorise la description de moyens inédits, inexistants et cependant réalisables d'un point de vue mécanique et informatique.

L'exemple suivant permettra de détailler la façon d'appréhender un procédé comportant une complexité suffisante pour justifier l'approche proposée.

II.2. Exemple introductif : l'ascenseur

II.2.1. Présentation de l'exemple

Considérons un système de production réparti sur quatre étages pour lequel il est nécessaire de gérer les transferts inter-étages. L'ascenseur est composé d'une cage pouvant contenir simultanément 10 pièces. De plus nous supposerons dans cet exemple que les pièces de type Pièce1 doivent impérativement transiter de l'étage 1 à l'étage 3, et les pièces de type Pièce2 de l'étage 2 à l'étage 4.



Exemple de l'ascenseur
Figure 1

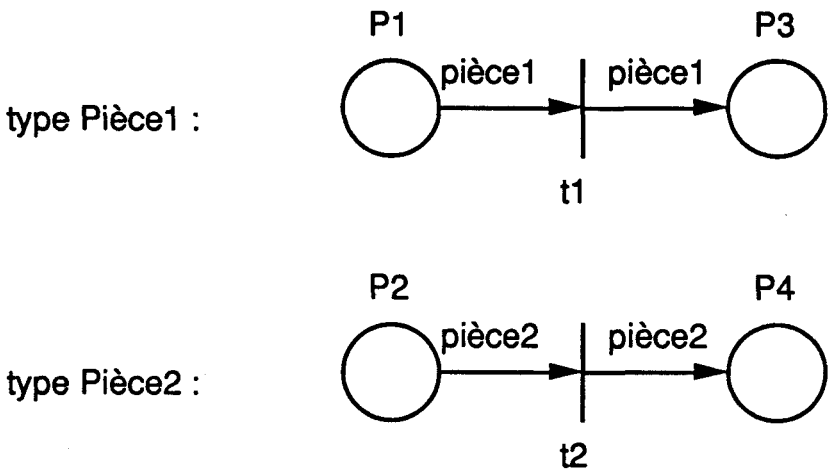
Notons que cette analyse est faite indépendamment de l'hypothèse de flexibilité (de routage ou de gamme) sur le système de production. Ainsi une production en ligne peut éventuellement utiliser ce type de transfert.

Nous allons, pour bien situer l'apport de notre démarche, tenter d'utiliser l'ancienne démarche CASPAIM sur cet exemple. Il convient dans ce sens d'élaborer le prégraphe (description fonctionnelle) de cette d'unité (KAP 88).

Il existe 4 zones de travail P1 à P4, associées aux 4 étages de l'ascenseur. Les 2 règles de production sont les suivantes :

(présence Pièce1 P1) => (présence Pièce1 P3)
et (présence Pièce2 P2) => (présence Pièce2 P4)

ce qui donne le RdPC (Prégraphe) suivant :



Description fonctionnelle de l'unité
Figure 2

Lors de la phase de structuration de ce Prégraphe (BOU 88), un processus est associé à chaque zone P1 à P4. De plus deux processus de transfert sont associés aux transitions t1 et t2. Ils réalisent la synchronisation nécessaire entre les processus amont et aval (hypothèse de parallélisme maximal). De plus t1 et t2 réalisent le séquencement des commandes de transfert.

Dans le cas de t1 :

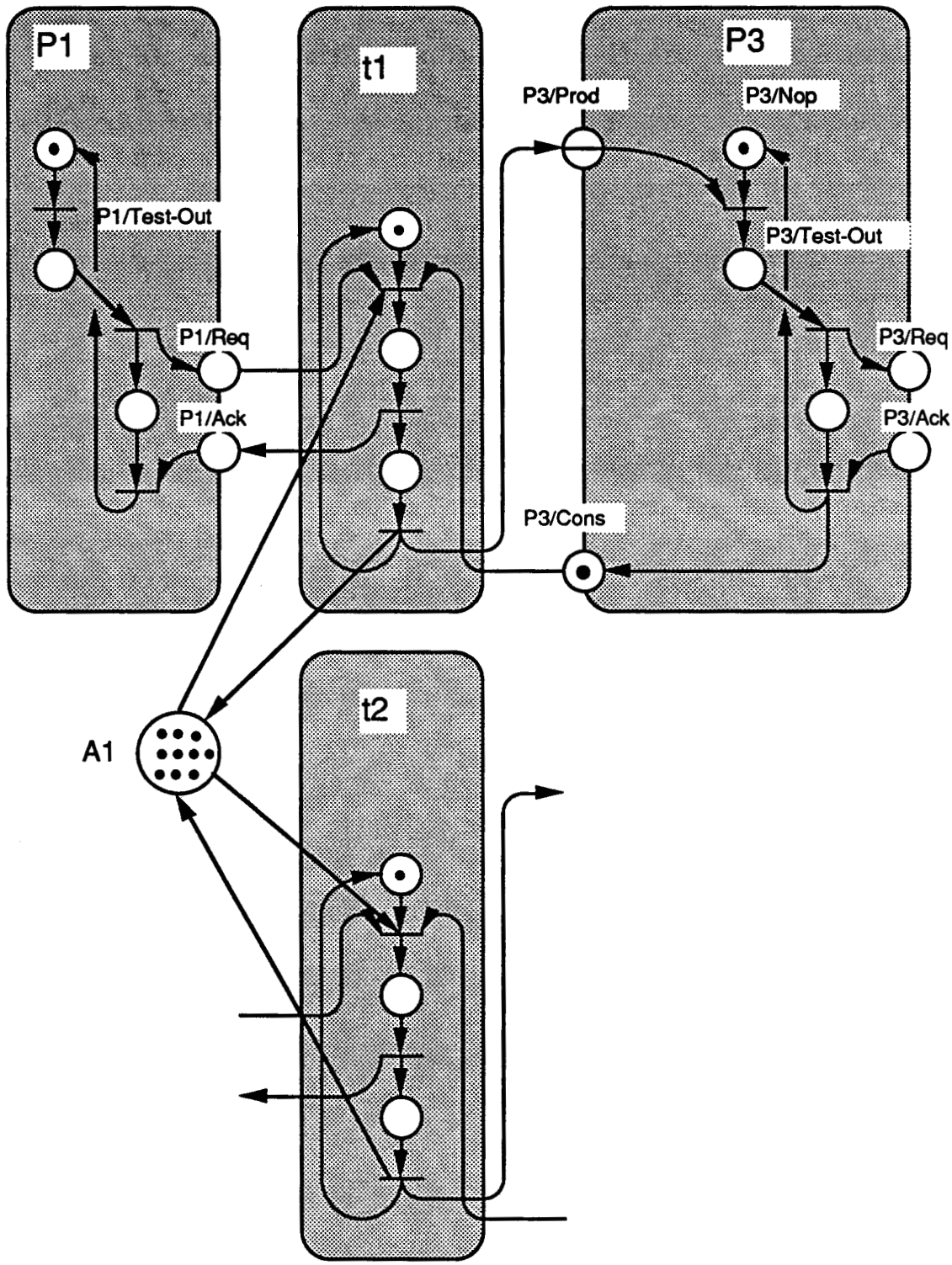
transfert du préhenseur vers P1
chargement du préhenseur en P1
transfert du préhenseur vers P3
déchargement du préhenseur en P3

La phase de structuration introduit de plus les contraintes d'accès aux ressources de transfert, induites par le fort degré de partage d'une même ressource.

II.2.2. Analyse des contraintes de synchronisation existantes

On suppose que les fonctions de transport réalisées par l'ascenseur sont réduites à celles que réaliserait un robot classique (nombre d'articulations quelconque, préhenseur unique). L'ascenseur est assimilé à une ressource critique qui ne peut réaliser qu'une seule tâche. Il est donc alloué avant le premier transfert et libéré à la fin du déchargement, en vue de son utilisation par une autre tâche de transfert. En somme, l'ascenseur exécute les transferts élémentaires de façon séquentielle. Il est clair que considérer l'ascenseur de cette manière restreint considérablement son degré de flexibilité. En particulier la possibilité de réaliser un grand nombre de transferts simultanés ne peut être exploitée. La mise œuvre de l'ancienne méthodologie restreint de fait la flexibilité potentielle du moyen de transport utilisé.

Une solution moins restrictive consisterait à ne voir de l'ascenseur que ses emplacements de stockage de pièces (10 au total). Faire un transfert de pièce via l'ascenseur reviendrait dans ce cas à allouer un des emplacements libres de la cage puis à réaliser la séquence de commandes de t1. Le RdP structuré résultant serait alors le suivant :



Synchronisation des processus
Figure 3

La place A1 est chargée de caractériser l'état d'allocation des emplacements.

Cela revient à supposer que les transferts des 10 emplacements de la cage sont totalement indépendants, comme si nous disposions de 10 moyens de transport autonomes. Ce n'est pas le cas de toute évidence puisqu'ils appartiennent tous à la même cage et donc se déplacent simultanément. Il est donc nécessaire de gérer l'entité ascenseur. En effet c'est l'ascenseur qui est effectivement commandé au sens de la montée ou de la descente.

II.2.3. Analyse du Procédé ascenseur vis à vis du champ d'application de la démarche

Notons à ce propos que le procédé ascenseur entre bien dans le champ d'application de notre démarche. Bien que son état (position courante de la cage dans l'espace) soit à évolution continue, il est aisé de discrétiser son comportement selon le modèle de la Figure 4 :

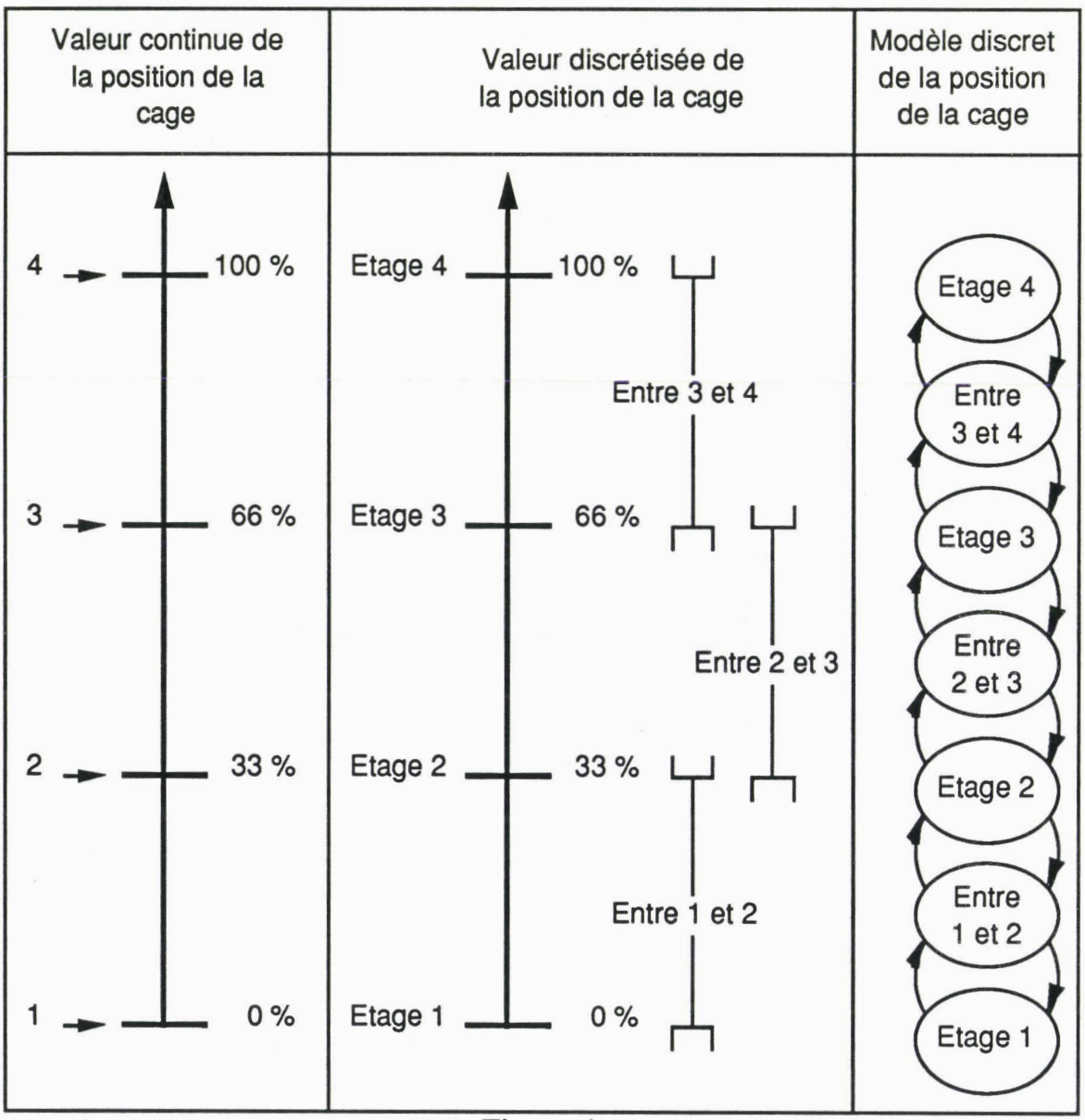


Figure 4

Bien que le comportement de l'ascenseur soit par essence continu, on peut, du point de vue qui nous intéresse ici, ne prendre en considération que 7 états distincts. Ces états suffisent à caractériser le comportement de l'ascenseur du point de vue de la commande.

Le déplacement de la cage ne peut résulter que de l'effet d'une commande appliquée à l'actionneur moteur électrique. En aucun cas la cage ne peut se déplacer de manière intempestive (surcharge, casse...). Il appartient donc bien à la catégorie des procédés concernés par notre étude.

Cet exemple est particulièrement significatif des caractères spécifiques complémentaires des approches méthodologiques proposées respectivement d'une part dans le cadre des systèmes à états continus (SEC) ou assimilés, et d'autre part dans le cadre des systèmes à événements discrets (SED).

En effet, si on considère l'ascenseur du strict point de vue du pilotage de la cage d'un étage à l'autre, il s'agit bien d'effectuer la synthèse d'un asservissement de position au sens des SEC. Si par contre on prend en considération la gestion globale des objets discrets transportés par l'ascenseur, il devient nécessaire de disposer d'une méthodologie de modélisation originale, relevant des SED. Dans ce contexte, le pilotage de la cage (SEC) devient esclave de la gestion discrète des pièces utilisant le moyen de transfert que constitue l'ascenseur (SED).

II.2.4. Identification des contraintes du Procédé

La difficulté rencontrée lorsque l'on veut commander le moteur vient du fait que pour des raisons de collision, il ne faut pas déplacer la cage lorsqu'on réalise un chargement ou un déchargement. Il s'agit ici de contraintes purement opérationnelles.

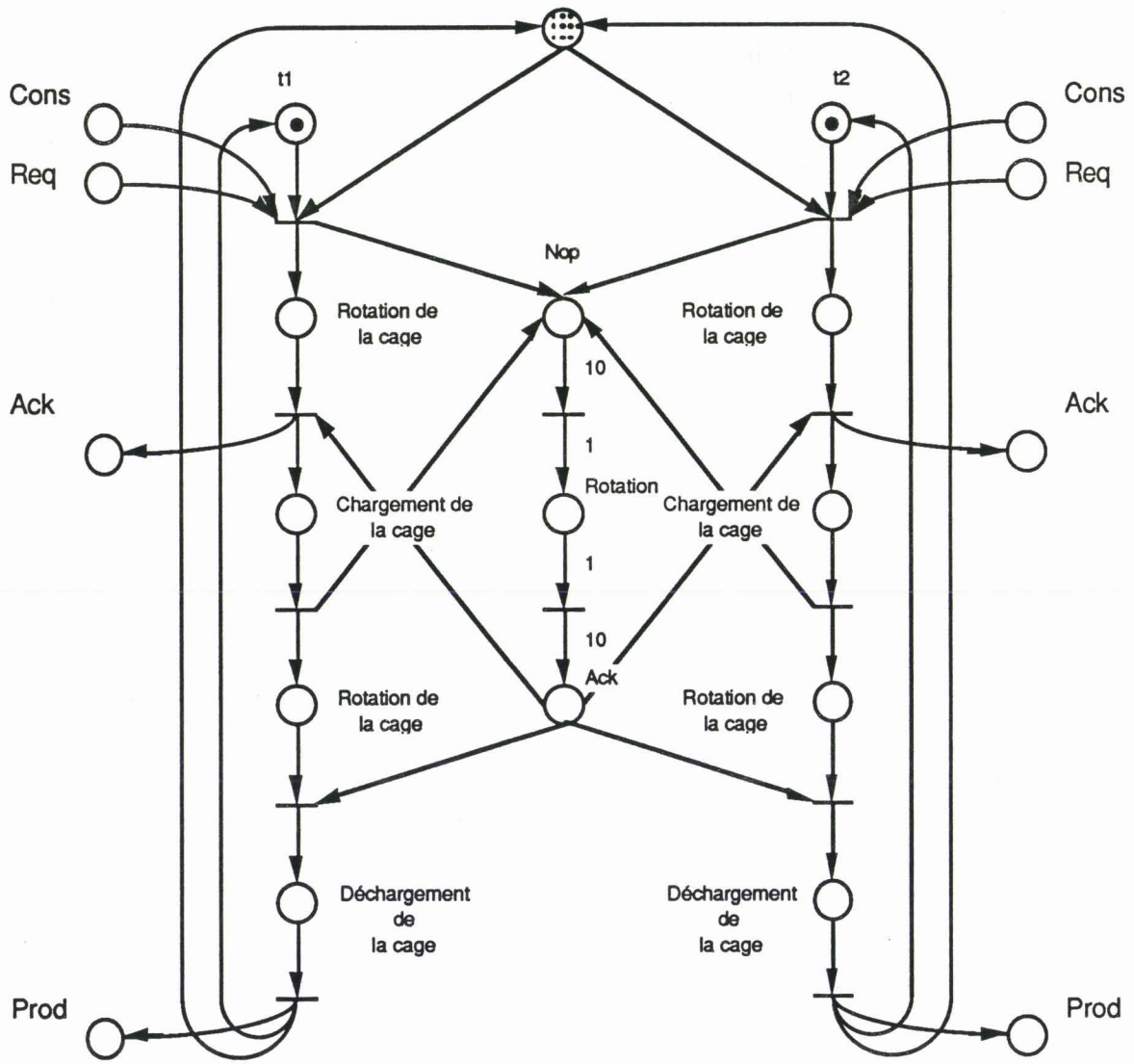
Une première solution qui satisfait cette contrainte consiste à réaliser en séquence :

1 chargement à l'étage 1
2 déplacement vers étage 2
3 chargement à l'étage 2
4 déplacement vers étage 3
5 déchargement à l'étage 3
6 déplacement vers étage 4
7 déchargement à l'étage 4
8 déplacement vers étage 1
9 retour au pas 1

Cette solution, bien que très simple à mettre en œuvre, est extrêmement

contraignante et rigide : l'arrêt à un étage se fait même s'il n'est pas nécessaire, une modification dans la destination d'une pièce entraîne la reprogrammation de la séquence, aucune possibilité d'optimisation de trajectoire n'est permise.

Une seconde solution consiste à synchroniser l'utilisation des 10 emplacements afin de satisfaire la condition précédente. On met en œuvre un mécanisme qui garantit que les 10 emplacements ne sont ni chargés ni déchargés lorsque l'opération de rotation est déclenchée. On obtient alors le graphe de commande suivant :



Synchronisation des processus

Figure 5

Comme nous le voyons sur ce RdP, la rotation du moteur ne peut se faire que si les 10 emplacements sont alloués. Il est sûr que ce type de solution devient excessivement lourde à gérer. De plus, réaliser la rotation du moteur alors qu'un nombre variable d'emplacements (0 à 10) sont alloués est

impossible à faire au moyen de l'outil RdP.

L'ancienne démarche CASPAIM parait donc inadaptée à cet exemple.

Dans ce cas précis nous cherchons à gérer deux catégories de ressources différentes, afin de ne pas restreindre la flexibilité potentielle du moyen de transport :

- les emplacements de stockage de l'ascenseur sont gérés de manière exclusive vis à vis des pièces : un emplacement réalise une fonction de stockage pour une pièce,

- la fonction de déplacement de la cage ne doit en aucun cas contraindre le nombre variable de requêtes de transfert de pièces.

Cet exemple d'illustration nous amène ainsi à mettre en évidence les difficultés de représentation du procédé selon un double objectif :

- en réalisant un découplage maximal entre ce qui concerne les pièces produites d'une part, et les moyens de production d'autre part,

- en générant un modèle du procédé totalement indépendant du modèle de la commande.

II.2.5. Nécessité de disposer d'un modèle fin du Procédé

L'un des points faibles de l'ancienne méthodologie CASPAIM concerne l'absence d'analyse du procédé. En effet, le procédé n'est pas pris en compte à priori. Or le choix d'un système physique peut avoir des répercussions très importantes sur les performances, les coûts de production, la flexibilité. Notons enfin que la maintenance est l'un des aspects essentiels de l'exploitation, ce qui justifie une élaboration rigoureuse et systématique d'un modèle spécifique du procédé. La surveillance de l'usure d'un outil ainsi que la maintenance préventive sont, de fait, réalisés indépendamment de l'utilisation de l'outil pour telle ou telle fabrication. Enfin des considérations sur la sécurité de fonctionnement du procédé font qu'il est impératif de modéliser finement tout système réel.

Ainsi nous verrons, au Chapitre III, la manière dont nous menons à bien l'analyse exhaustive des systèmes de production évoqués. La méthode est basée essentiellement sur les points suivants :

- l'analyse du système physique doit être indépendante de l'utilisation effective du système de production,

- la nécessaire identification de tous les Objets Physiques qui doivent être

commandés, et contrôlés (contrôle de comportement). Il s'agit par exemple pour l'ascenseur de pouvoir caractériser à la fois :

- l'état de la cage de l'ascenseur (position géographique)
- l'état des emplacements de stockage (vide, plein,...)

- une description arborescente du système de production considérant les Objets Physiques commandables,

- l'identification des contraintes existant entre ces objets,

- l'identification des relations d'accessibilité entre moyens de stockage.

Pour notre exemple, une contrainte entre les objets concerne l'interdiction de tout mouvement de la cage d'ascenseur pendant une opération de chargement ou de déchargement d'un des emplacements de stockage.

II.2.6. Approche fonctionnelle de l'exemple

La fonction de transport assurée par l'ascenseur peut ainsi être décomposée en deux tâches complémentaires :

- la fonction de stockage (temporaire) d'objets dans les emplacements prévus à cet effet,

- la fonction globale de déplacement de la cage contenant les emplacements.

Si l'on se place du point de vue d'une pièce transportée, il est impératif de réaliser en séquence les opérations suivantes :

allocation d'un emplacement
positionnement de la cage à l'étage de départ
chargement de l'emplacement de la cage
positionnement de la cage à l'étage d'arrivée
déchargement de l'emplacement de la cage
libération de l'emplacement

Si l'on suppose une stricte indépendance entre les pièces, il apparait que la fonction de déplacement ne doit en aucun cas être contrainte par un séquencement impératif (étage 1 puis étage 2 etc...) ou un traitement par lot (demi charge en P1, demi charge en P2 etc...).

De fait la gestion du déplacement doit être la plus souple possible, ne serait-ce que pour pouvoir envisager une optimisation à court terme de la trajectoire de la cage et doit préserver de ce fait au niveau du modèle descriptif tous les degrés de liberté du procédé.

Nous pouvons constater que du point de vue de la pièce transportée le trajet effectif de la cage importe peu. La seule contrainte, du point de vue de la pièce, est que la cage arrive, au bout d'un temps fini, à l'étage de départ ou d'arrivée. En somme la fonction de déplacement de la cage est considérée comme un service qui peut être rendu à n pièces simultanément (n entre 1 et 10).

Le deuxième point de vue du problème est celui de la cage elle-même. En effet, sa seule fonction est la fonction de déplacement. Cette fonction est assurée pour transporter les pièces, mais aussi lors des trajets à vide et même lors d'opérations de maintenance (manipulations de la cage par un opérateur). Dans la mesure où la fonction de déplacement de la cage satisfait, lors de l'arrivée à un étage, un nombre variable de positionnements du point de vue des pièces, il est possible d'envisager une optimisation de la trajectoire selon un ou plusieurs critères.

Ce système de transport comporte deux caractéristiques essentielles :

- sa capacité à contenir des objets,

- son fonctionnement interne, indépendamment de ce qu'il contient.

Ce type de constatation nous conduit à proposer une architecture de contrôle modulaire et fortement découplée.

II.2.7. Analogie avec le Génie Logiciel (MOR 88)

Nous rejoignons ainsi l'objectif du Génie Logiciel. Dans (BOO 88), l'auteur cite un article (ROS 75) de O.J. Ross, J.B. Goodenough et C.A. Irvine, pour qui les buts du Génie Logiciel sont les suivants :

"Il existe quatre propriétés suffisamment générales pour être acceptées comme buts pour toute la discipline du Génie Logiciel : la modifiabilité, l'efficacité, la fiabilité et l'intelligibilité".

Le Génie Logiciel est basé sur les principes de conception suivants :

- abstraction et dissimulation d'information : l'objectif est de voir le système manipulé (informatique, matériel) selon plusieurs niveaux d'abstraction croissant et en adéquation avec son utilisation,

- modularité et localisation : l'objectif est de concevoir une collection de modules logiciels à forte cohésion (concentration de moyens de calcul, de données), et à faible couplage (limitation du volume d'échange d'informations entre modules),

- uniformité, intégralité et validabilité : le système produit des modules similaires par leur formulation, il est complet vis à vis de sa spécification et validable par des tests appropriés.

De fait la démarche de conception CASPAIM respecte les principes de conception du Génie Logiciel. Durant la phase de redéfinition de la démarche de conception CASPAIM, nous nous sommes attachés à concevoir une architecture de commande et de contrôle modulaire et le plus proche possible de la réalité matérielle. Les principes du Génie Logiciel se trouvent donc satisfaits.

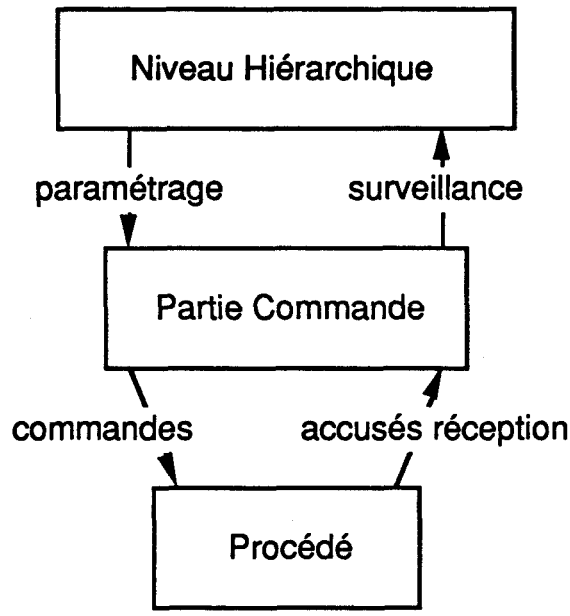
Les grandes lignes sont les suivantes :

- la Partie Commande (PC) assure le séquencement et la synchronisation des opérations effectuées sur chaque produit ou sous-produit, gère l'allocation des moyens (outils, machines, stocks) nécessaires, selon une stratégie d'allocation imposée par le Niveau Hiérarchique.

- les commandes transmises par la Partie Commande transitent par un bloc appelé Interface dont la fonction est d'assurer les commandes de la PC, tout en respectant les contraintes inter-objets inhérentes à tout système réel. C'est dans cette Interface que sont gérées les requêtes à destination du moteur de l'ascenseur. Là encore une fonction de choix des commandes à satisfaire parmi les commandes reçues est réalisée selon une stratégie de fonctionnement du Procédé qui est imposée par le Niveau Hiérarchique.

II.3. Le produit de la démarche de la chaîne de conception CASPAIM

Dans sa version initiale, la méthodologie CASPAIM conduit à une séparation du système de contrôle en trois niveaux :



Architecture du système de contrôle
Figure 6

Tout en respectant, dans les grandes lignes, cette structure, l'étude de cas réels nous a amenés à détailler cette décomposition.

II.3.1. La Partie Commande

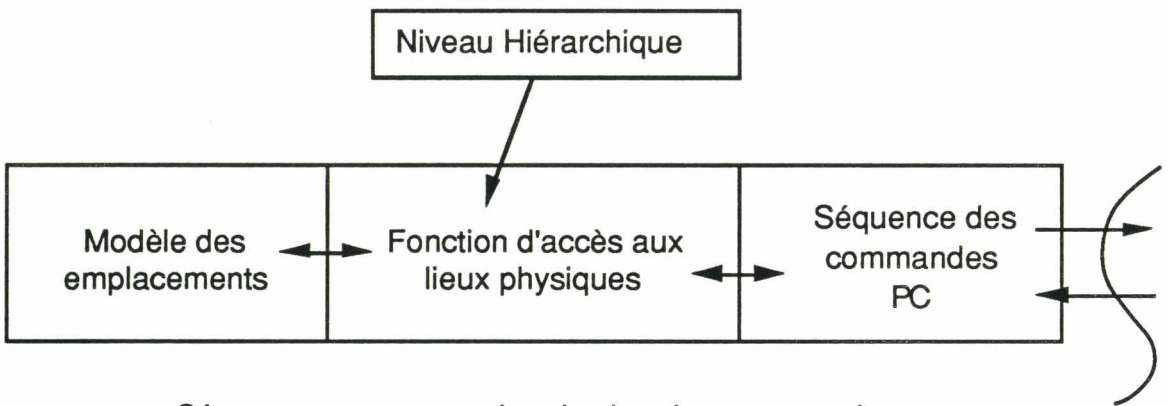
La finalité de tout système de production dans l'industrie manufacturière est de réaliser toutes les opérations permettant d'aboutir à un produit fini à partir de produits bruts ou semi ouvrés. Le rôle de la Partie Commande est de maîtriser le séquencement et la coordination des opérations de chaque pièce produite.

Elle utilise naturellement les moyens du Procédé. Le partage de ces moyens entre les diverses portions de la PC doit donc être géré. Nous avons pris le parti de ne gérer, dans la PC, que les emplacements de stockage nécessaires pour le stockage permanent (ou de longue durée) et le stockage temporaire (sur un robot, un ascenseur, un chariot). Ainsi, dans le cas de l'ascenseur par exemple, la PC gère l'allocation et l'utilisation des emplacements de la cage. Cette fonction d'allocation respecte une stratégie (ou

un critère de choix) qui lui est imposé (paramétrage) par le Niveau Hiérarchique. Notons que la fonction de déplacement de la cage est commune à tous les emplacements. Ce fait n'est pas pris en compte au niveau de la PC. La fonction de déplacement est vue comme un service qui peut être rendu par le procédé, sans se préoccuper des modalités de ce service.

De manière générale, le rôle de la PC est strictement limité à la gestion des emplacements de stockage ou des manipulateurs nécessaires aux pièces, et au séquencement des actions du point de vue des pièces. La gestion effective des moyens physiques est assurée par le module Interface que nous détaillerons dans la section suivante.

Fonctionnellement les interactions PC<->NH se schématisent comme suit :



Séquencement et synchronisation des commandes du point de vue des pièces

Figure 7

Pour l'exemple de l'ascenseur, la Partie Commande relative à la modélisation d'un déplacement de P1 à P3 sera la suivante :

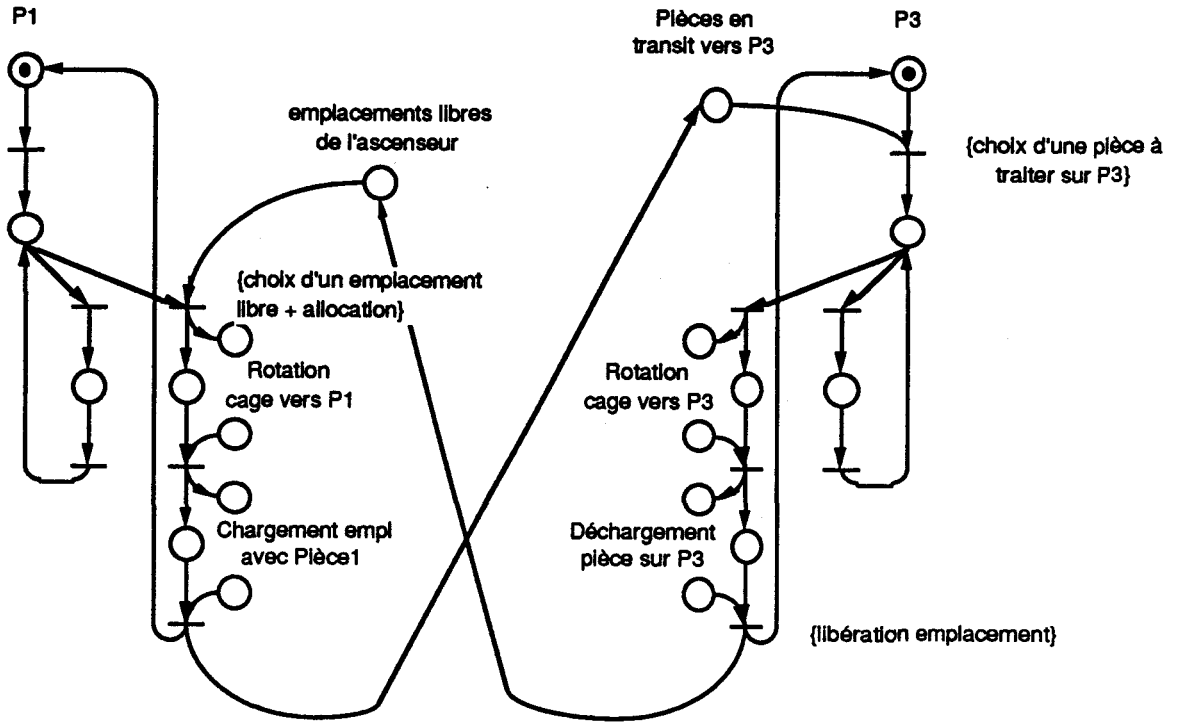


Figure 8

L'ascenseur est vu ici comme une zone de stockage active (les emplacements se déplacent). Notons à ce propos que la démarche permet de gérer indifféremment :

- un transfert d'un stockage fixe vers un stockage mobile : chargement d'un robot avec une pièce au sol,

- un transfert d'un stockage mobile vers un autre stockage mobile : déchargement d'un robot sur un ascenseur, d'un robot vers un autre, d'un chariot sur un robot, d'un plateau mobile sur un ascenseur...

Les moyens de transport sont vus tout d'abord comme des moyens de stockage. Cette approche permet d'appréhender de la même façon un magasin, une machine d'assemblage, de tournage et un robot de manipulation. A titre d'exemple, ce niveau de détail se justifie d'autant plus que les opérations de transformation sont courtes. Dans ce cas, les durées de transfert peuvent être prépondérantes par rapport aux durées des transformations.

Le chapitre IV détaille la méthode de conception itérative et hiérarchique de la Partie Commande, ainsi que sa validation par étapes.

La Partie Commande ne communique pas directement ses commandes au Procédé réel. La PC requiert l'accomplissement d'un service de la part du procédé. Le rôle de l'Interface, intercalé entre la PC et le Procédé réel est de réaliser effectivement ce service.

II.3.2. L'Interface

Dans l'exemple de l'ascenseur, la PC transmet des ordres de rotation de la cage vers les étages. Compte tenu du nombre d'emplacements de stockage de la cage, l'interface peut recevoir entre 1 et 10 commandes simultanées de déplacement de la cage sur requête d'accès émanant des pièces. En effet, lorsqu'un emplacement est alloué à une pièce, la portion de la PC qui gère la pièce envoie au maximum une commande de déplacement de la cage. Nous appellerons commande aveugle ce type d'ordre qui ne tient pas compte des contraintes liées à l'état actuel du process commandé.

II.3.2.1. La fonction de l'Interface.

Le rôle de l'Interface est de satisfaire ces différentes commandes. Notons à ce propos qu'une intervention opérateur, donc une commande supplémentaire doit pouvoir être prise en compte. Du point de vue de la PC, donc des pièces manipulées, cette commande ne modifie en rien l'état d'avancement et la terminaison des transferts engagés.

Ainsi, l'Interface doit satisfaire les commandes provenant de la PC mais également toute autre commande opérateur. Une opération quelconque peut être menée de manière transparente par rapport à la PC.

Schématiquement la liaison PC/Interface est réalisée comme suit :

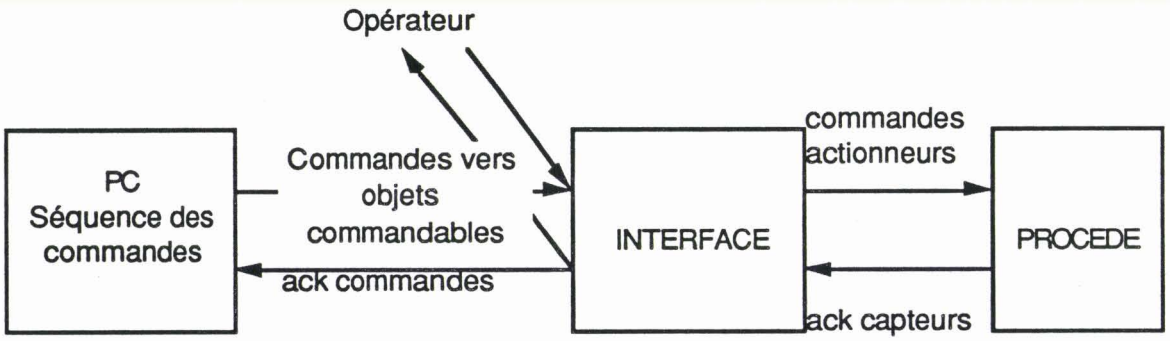


Figure 9

L'Interface possède en interne une image de l'état du Procédé. Nous verrons Chapitre III que nous pouvons construire une image de cet état au moyen d'automates à états finis associés à chaque objet élémentaire commandé. Ce concept d'Interface permet de prendre en compte les contraintes statiques et dynamiques inhérentes à tout système réel. Les commandes transmises au Procédé sont appelées valides (par rapport à l'état courant du procédé quand elles ont été transmises).

Au vu de l'exemple, l'Interface réalise un ordonnancement à court terme des commandes qui lui sont transmises. Pour notre exemple d'illustration, en

fonction de l'étage courant qui appartient à l'image du procédé de l'Interface, la décision du prochain étage destination est évaluée à partir des commandes courantes et d'une stratégie de fonctionnement imposée par le Niveau Hiérarchique. L'Interface optimise donc à court terme le fonctionnement du procédé. Dans notre exemple, le choix de la prochaine destination la plus proche dans le sens de la marche peut paraître intéressant. Cependant tout critère est à priori valable à condition toutefois que toute commande soit satisfaite, et en un temps fini.

Le modèle de l'ascenseur doit donc pouvoir prendre en compte les deux points de vue suivants :

- le point de vue des emplacements de stockage, pour une optimisation de l'allocation des emplacements,

- le point de vue purement opérationnel, pour une optimisation de fonctionnement du procédé.

Le produit de la démarche est alors schématisé de la manière suivante:

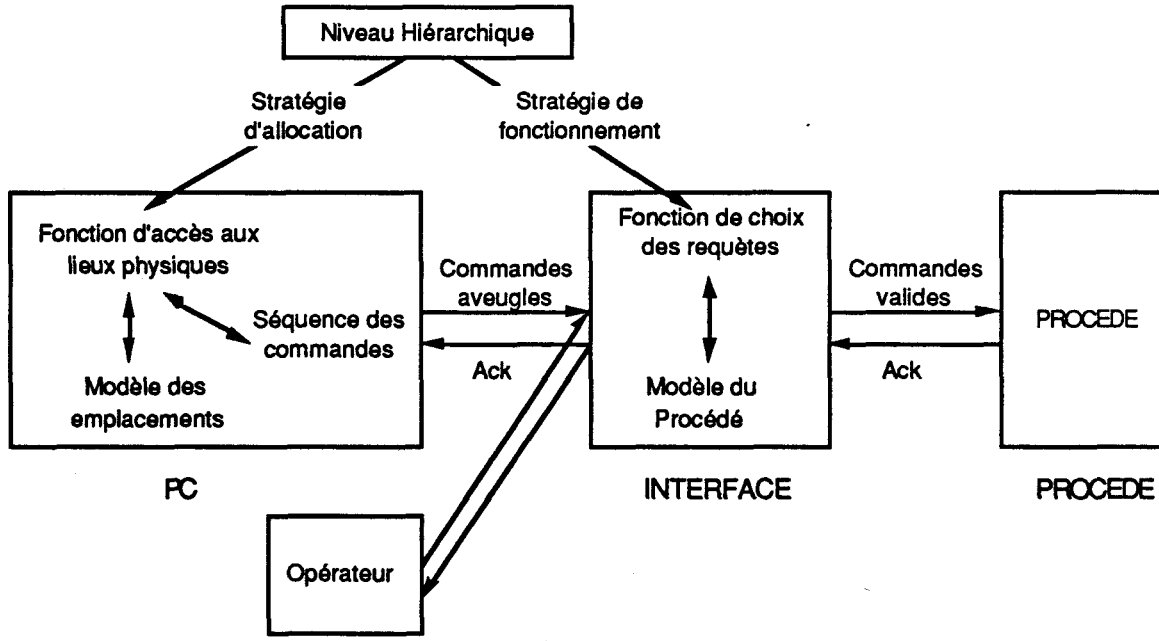


Figure 10

Il est clair que les deux fonctions d'optimisation locales gèrent le même système réel mais selon deux points de vue différents. Elles sont donc corrélées très fortement. Dans l'état actuel de l'avancement de nos travaux nous considérons cependant qu'elles sont totalement découplées.

L'allocation d'un emplacement est faite indépendamment de l'étage courant.

Le déplacement de la cage est fait indépendamment d'une désallocation

éventuelle.

II.3.2.2. Un exemple d'Interface : le gestionnaire de disque dur (BEA 90)

De la même manière, évoquons pour illustrer notre propos qu'un gestionnaire de disque dur, appelé aussi driver, constitue un bon exemple d'Interface. En effet, une image de l'occupation des secteurs est présente en mémoire vive de l'ordinateur. Une lecture de secteur consiste à déplacer la tête de lecture vers le secteur, puis réaliser la lecture tout en déplaçant la tête. Couramment les disques durs possèdent une tête de lecture par face de plateau. Il devient alors possible, soit de réaliser plusieurs lectures simultanées ou, à défaut, d'optimiser le déplacement des têtes si une seule lecture à la fois est possible. Ici, de même que dans le cas de l'ascenseur, nous avons un double point de vue pour caractériser la gestion du disque dur :

- le disque peut être vu comme un support mobile de secteurs (entité élémentaire de stockage de l'information, accessible de manière exclusive en lecture ou en écriture),

- c'est aussi un système mécanique : rotation du disque, déplacement des têtes de lecture et écriture.

La gestion d'un tel système se fait à deux niveaux :

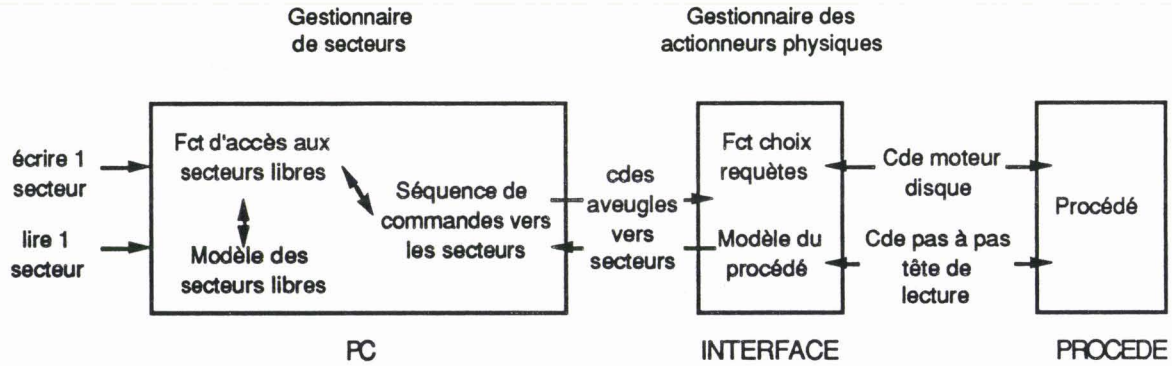


Figure 11

La "PC" envoie des commandes de lecture ou écriture simultanées à l'Interface, indépendamment de l'état réel du lecteur de disque.

La lecture d'un secteur (service rendu par le driver) suppose le déplacement de la tête vers le secteur, puis la lecture du secteur (éventuellement simultanée avec la lecture d'autres secteurs d'autres plateaux). Le service est alors rendu. L'intérêt d'une analyse aussi fine est qu'elle permet de dissocier au maximum les fonctions de décision et d'optimisation locales.

Notons au passage que notre démarche s'adapte aisément à des processus non typiquement manufacturiers. Les hypothèses décrites au paragraphe II.1. sont ici satisfaites. L'analogie entre les deux exemples peut concrètement s'exprimer sous la forme suivante :

Point de vue PC dans le cas de l'ascenseur :
- support : emplacement de stockage
- entité stockée : pièce
- chargement du support : positionnement de la cage puis stockage
- déchargement du support : positionnement de la cage puis déstockage

Point de vue PC dans le cas du disque dur :
- support : secteur
- entité stockée : 2^n octets
- chargement du support : positionnement de la tête puis écriture
- déchargement du support : positionnement de la tête puis lecture

Point de vue Interface dans le cas de l'ascenseur :
- services : positionnement de la cage, chargement, déchargement
- contraintes :
  cage à l'arrêt durant chargement ou déchargement
  nombre quelconque de chargement et déchargements simultanés
  pour un emplacement : exclusion mutuelle entre chargement et déchargement

Point de vue Interface dans le cas du disque dur :
- services : déplacement de la tête, lecture, écriture
- contraintes :
  tête fixe durant lecture ou écriture d'un secteur
  une écriture seule en exclusion avec des lectures multiples simultanées pour un secteur : exclusion entre lecture et écriture
  rotation des plateaux lors du déplacement de la tête, des lectures, des écritures

L'Interface, dont l'étude est en cours au laboratoire, transmet, par construction, des commandes aux actionneurs physiques qui sont dépendantes de l'état courant du procédé. Elles satisfont aux contraintes inter-objets : elles sont valides.

II.3.3. Le Procédé

Le procédé physique est décomposé en Objets Physiques commandables. Du fait des contraintes inter-objets, nous faisons transiter les commandes en provenance de l'Interface par un filtre de commande. Il s'agit d'un organe de sécurité visant à palier toute défaillance éventuelle du système de commande :

incomplétude de l'Interface, mauvaise gestion des emplacements dans la PC etc...

Nous obtenons le schéma suivant :

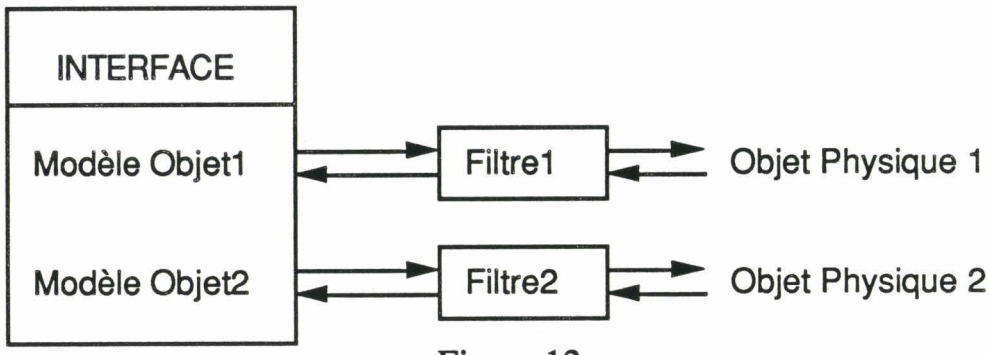


Figure 12

Le filtre de commande est un système de contrôle temps réel d'un Objet Physique. Nous verrons au chapitre III que cette notion est analogue à celle de Boîte Fonctionnelle utilisée au Laboratoire d'Automatique et de Commande Numérique (LACN) du Centre de Recherche en Automatique de Nancy. Un filtre de commande possède donc une image dynamique. Il peut détecter tout écart par rapport à son évolution attendue, et possède une double fonction :

- détection de commandes erronées par rapport à l'état courant,

- détection de toute valeur de capteur erronée par rapport à l'état courant.

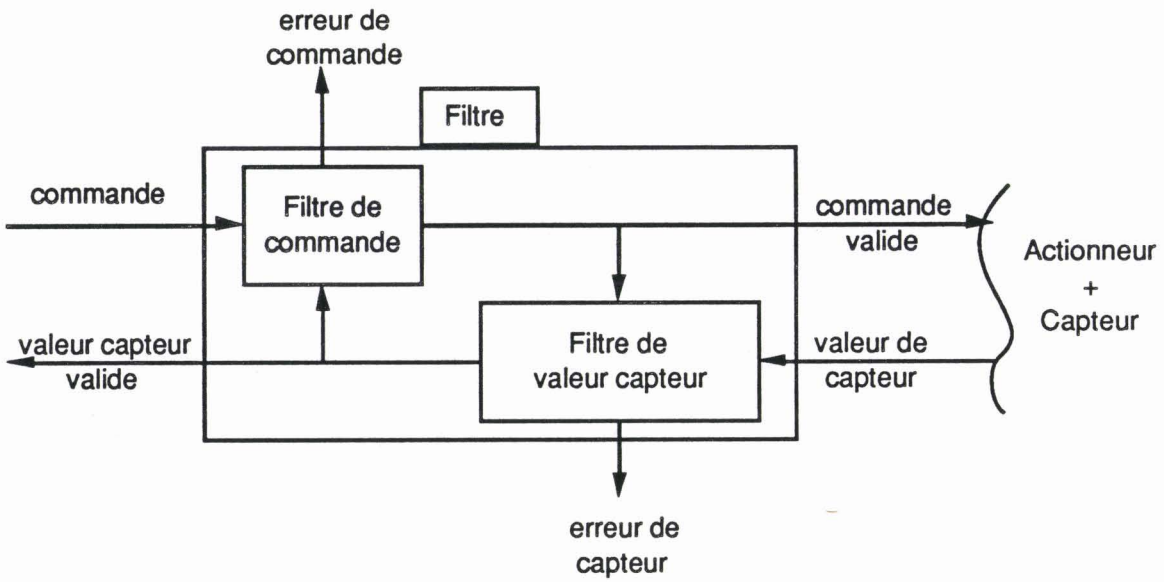


Figure 13

Nous rejoignons ainsi en partie les notions d'Elément de Partie Opérative du LACN (LHO 88, TIX 89, MOR 88).

Par construction, les commandes en provenance de l'Interface sont valides. Cependant une intervention opérateur à ce niveau bas de commande peut être nécessaire (maintenance, réparation ...). Un contrôle de cohérence des commandes opérateur est donc indispensable. En effet un opérateur n'a qu'une vision partielle du procédé qu'il manipule. Il risque ainsi de provoquer involontairement une défaillance du système.

Le schéma complet de l'architecture de contrôle produite par la nouvelle démarche que nous proposons est la suivante :

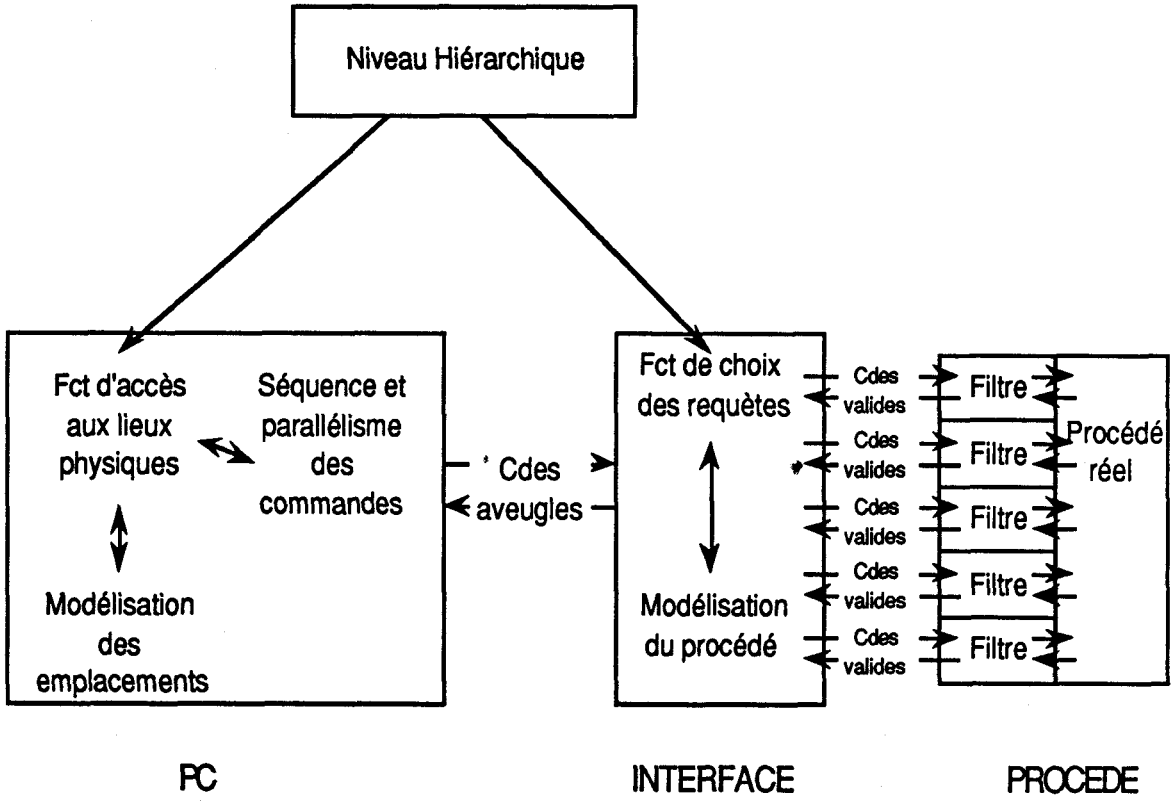


Figure 14

Notons pour conclure ce paragraphe que l'Interface et les filtres de commande sont spécifiques de la démarche de conception proposée. Ils sont justifiés par le nécessité de résoudre des cas réels complexes à très haut degré de parallélisme. Nous recherchons de ce fait le degré de parallélisme de commande maximal (quels sont les Objets Réels commandés) et une spécification des contraintes opérationnelles aussi peu restrictive que possible. De même une conception modulaire est d'autant plus intéressante que la portée d'une modification est faible, conformément aux principes du Génie Logiciel.

II.4. Proposition d'un cycle de vie d'un projet (CASPAIM-2)

II.4.1. La méthodologie de conception (CRU 90a, CRU 90b)

Par essence, une méthodologie est un ensemble de méthodes permettant, dans le contexte de l'étude, à un concepteur de mener à bien la spécification, la programmation et l'implantation du système de contrôle d'un atelier ou cellule flexible. Le caractère flexible de la production n'est pas déterminant dans la démarche. Cependant nous verrons dans quelle mesure la flexibilité peut influencer le choix du modèle de la Partie Commande.

Il s'est avéré que le modèle produit permet également d'effectuer une évaluation de la solution informatique et matérielle choisie.

Une méthodologie présente bien sûr l'avantage de guider le concepteur, notamment lors de la spécification et dans la phase de conception du système. Elle permet aussi une automatisation très poussée des phases de génération de programme et donc en principe réduit le risque d'erreurs de transcription bien connu de tout programmeur. Le résultat standardisé de la démarche facilite considérablement les phases, peu abordées, de la maintenance, notamment informatique, du système produit, ainsi que la mise à jour immédiate de la documentation technique.

La démarche, sous sa forme actuelle, bénéficie pour une large part de l'expérience pratique acquise par le laboratoire lors de l'étude de cas complexes. Toutefois la démarche de conception présente peu d'analogies avec la version antérieure de CASPAIM.

Le processus de fabrication d'un produit est modélisé sous la forme d'un réseau de Petri (RdP) dans lequel :

- une place représente un état d'avancement du produit,

- une transition modélise une fonction de transformation physique qui modifie l'état d'avancement du produit.

Ce modèle, appelé gamme logique, est donc un modèle purement fonctionnel (indépendant des moyens) et spécifique à chaque produit fabriqué.

Il est ensuite enrichi en prenant en compte l'aspect opérationnel (moyens de production et de manipulation) par étapes pour former une gamme opératoire.

De nouvelles places et de nouvelles transitions sont introduites afin de prendre en compte les différents lieux de support des pièces. Dans une gamme opératoire, une place modélise la présence d'une pièce dans un état d'avancement donné et sur un lieu donné.

Les lieux de support ayant une capacité finie, la gamme opératoire est

ensuite contrainte en ajoutant autant de places que de lieux de stockage de pièces. Ces places contiennent autant de jetons qu'il existe d'emplacements de stockage (ES) sur le lieu considéré. Une pièce devant être stockée sur un lieu de stockage doit s'allouer au préalable un ES. Ce mécanisme d'allocation et de restitution de ressources ES est traduit sur le RdP de la gamme opératoire de la pièce concernée, et constitue ce que l'on appelle le protocole d'accès à ce lieu.

Un certain nombre d'avantages résultent de la nouvelle approche proposée :

- la remise en cause du choix matériel est facilitée, elle était très difficile dans la démarche antérieure,

- l'indépendance entre la spécification des gammes logiques et la spécification du procédé est assurée,

- l'hypothèse de flexibilité maximale est prise en compte,

- la modularité de la démarche et du produit est maximale,

- la compatibilité totale avec la version antérieure est garantie.

Le synoptique de la méthodologie relevant de CASPAIM-2 est le suivant :

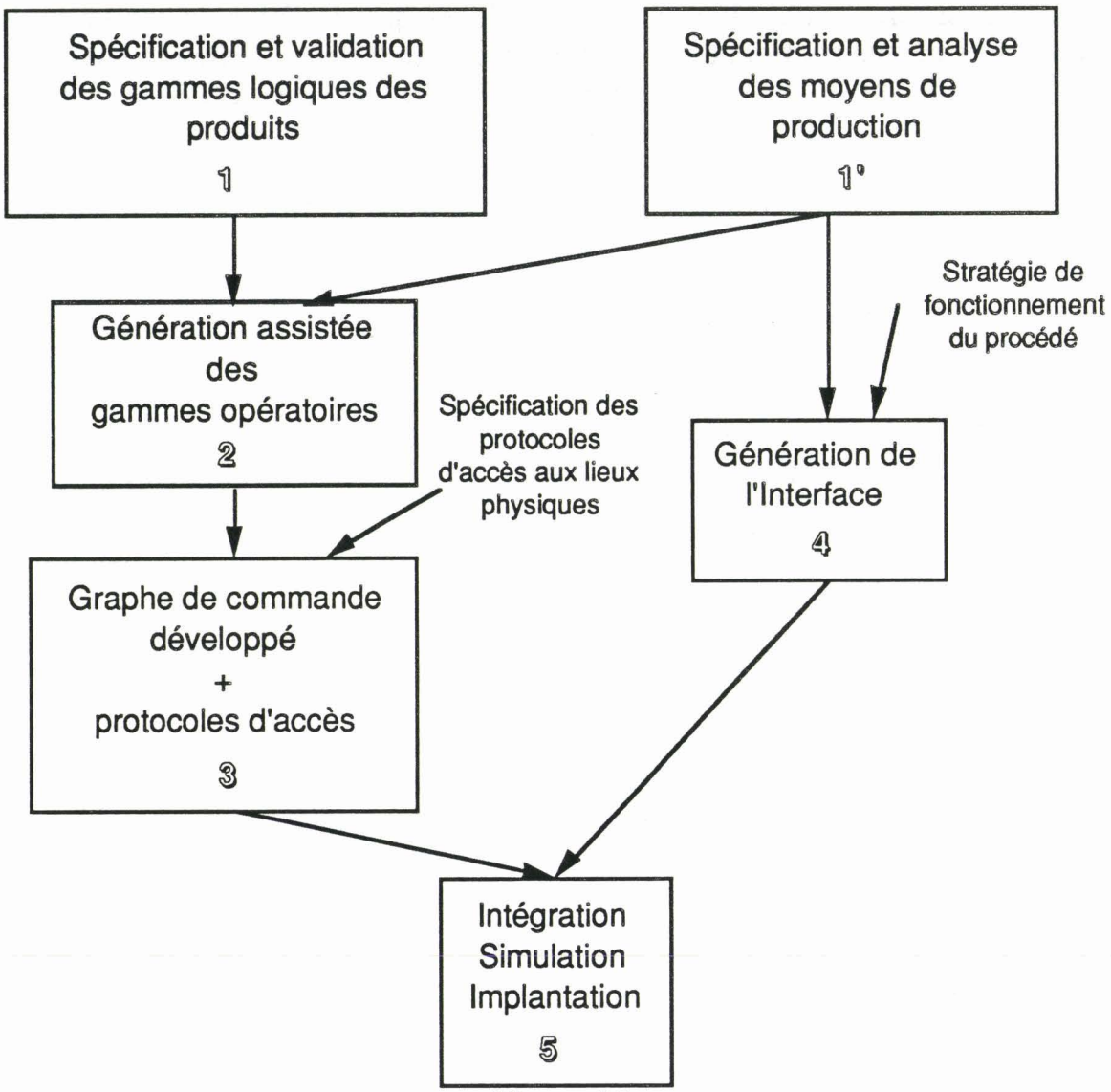


Figure 15

Les retours arrière ne sont pas représentés.

Ce schéma met en relief les points fondamentaux suivants:

- la spécification fonctionnelle des produits (gammes logiques) et la spécification opérationnelle (moyens de production) sont réalisées en parallèle et de manière indépendante l'une de l'autre,

- une même gamme logique peut être exploitée sur plusieurs spécifications opérationnelles : ce cas correspond à une démarche de conception ou d'extension de l'architecture matérielle selon plusieurs variantes, alors que la gamme logique est imposée. Il s'agit le plus souvent d'augmenter la flexibilité de routage des pièces;

- une spécification opérationnelle peut être prise en compte pour plusieurs

gammes logiques : ce cas correspond à la cohabitation de plusieurs gammes logiques (que l'on peut agréger ou non) qui induisent des temps opératoires différents sur un même système physique (esprit du Prégraphe de l'ancien CASPAIM), ou éventuellement l'ajout de nouvelles gammes logiques. Il s'agit souvent d'augmenter la flexibilité des gammes logiques ;

- deux spécifications existantes peuvent être combinées afin de vérifier qu'il est possible de créer une gamme opératoire à partir d'une gamme logique et d'une spécification opérationnelle (atelier) données.

Les effets des modifications des gammes logiques et du procédé qui viennent d'être exposés sont synthétisés sur le tableau qui suit. On retrouve verticalement la spécification fonctionnelle (gammes logiques), et horizontalement la spécification opérationnelle (Procédé). Nous pouvons remarquer qu'une modification de la spécification fonctionnelle ne peut être réalisée qu'à spécification opérationnelle constante, et réciproquement.

Gammes logiques	Spécification opérationnelle (Procédé)			
	Création	Valide	Extension	Réduction
Création		Contrôler l'adéquation des GL aux moyens		
Valide	Contrôler l'adéquation des moyens aux GL	GL et Procédé compatibles	Augmente la flexibilité de routage	Diminue la flexibilité de routage
Extension		Augmente la flexibilité des GL		
Réduction		Diminue la flexibilité des GL		

Figure 16

II.4.2. L'ossature du modèle de commande produit

Une démarche de conception n'est pas une fin en soi. En effet, le produit de la démarche peut être remis en cause pour différentes raisons : modification des caractéristiques du produit (occasionnel), ou modification des caractéristiques de l'unité de production (remplacement d'un appareil, augmentation de la capacité de production).

Dans la pratique, les gammes logiques ne sont modifiées qu'exceptionnellement tandis que le procédé peut changer très fréquemment

de configuration. En effet la robustesse d'une unité de production se mesure pour une grande part par sa capacité à s'adapter aux pannes matérielles de courte durée. Dès lors, la mise en service ou hors service d'une machine peut s'assimiler à une extension ou une réduction du procédé sans modification des gammes logiques.

La fréquence et le caractère inévitable des défaillances de courte durée font que l'étude du fonctionnement en modes dégradés du procédé doit être réalisée dès la phase de conception. Nous devons prendre en compte ce type de contraintes. Les modifications des gammes logiques étant supposées rares, elles entraînent le plus souvent une reprogrammation totale du système de commande et donc un arrêt momentané de la production. L'idéal serait bien sûr d'éviter cet arrêt. Cela suppose que l'on puisse reprogrammer le système de commande en maintenant la production. Cette hypothèse ne peut être aujourd'hui prise en compte, étant donnée la très grande complexité du problème.

II.4.3. Spécification des gammes logiques

Ces considérations d'ordre pratique nous ont amenés à bâtir une ossature de la Partie Commande à partir des gammes logiques. C'est le modèle RdP construit à partir des gammes logiques qui sera expansé et enrichi par étapes (CRA 89) pour aboutir au "code" effectivement implanté (sous la forme par exemple d'un Grafcet réparti). De ce fait une modification mineure dans une gamme logique peut entraîner une reprogrammation importante. La spécification et la validation des gammes logiques sont donc réalisées avec soin. L'outil RdP permet une validation formelle de propriétés importantes (notamment la terminaison propre) qu'il est vital de vérifier avant tout développement.

II.4.4. Génération des gammes opératoires

C'est seulement à l'issue de cette étape que nous prenons en compte la description opérationnelle des moyens de production. La génération des gammes opératoires est faite et validée par étapes en prenant en compte le procédé de plus en plus finement.

L'exemple qui nous servira de support tout au long de notre exposé illustre cette approche décomposée en plusieurs phases. Nous allons présenter rapidement, et de manière informelle, la manière dont nous abordons la prise en compte de ce moyen de transport.

L'exemple comporte en effet un convoyeur complexe. Le moyen convoyeur est vu comme une collection de zones de stockage communicantes. Par nature, un convoyeur réalise une fonction de transport de pièces. Du point de vue d'une pièce cette fonction peut être réalisée de n'importe quelle

manière. Lors de la construction des gammes opératoires, le convoyeur est vu de façon macroscopique : il doit être capable de transférer une pièce d'un lieu à un autre. Schématiquement le convoyeur possède des emplacements pour loger les pièces et réalise le transfert de A vers B :

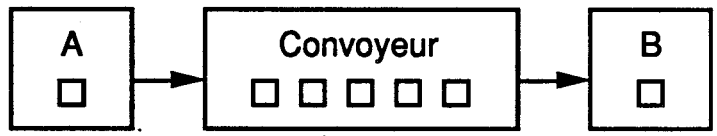


Figure 17

A l'étape suivante de la définition des gammes opératoires, le convoyeur est vu comme un ensemble de sections communicantes.

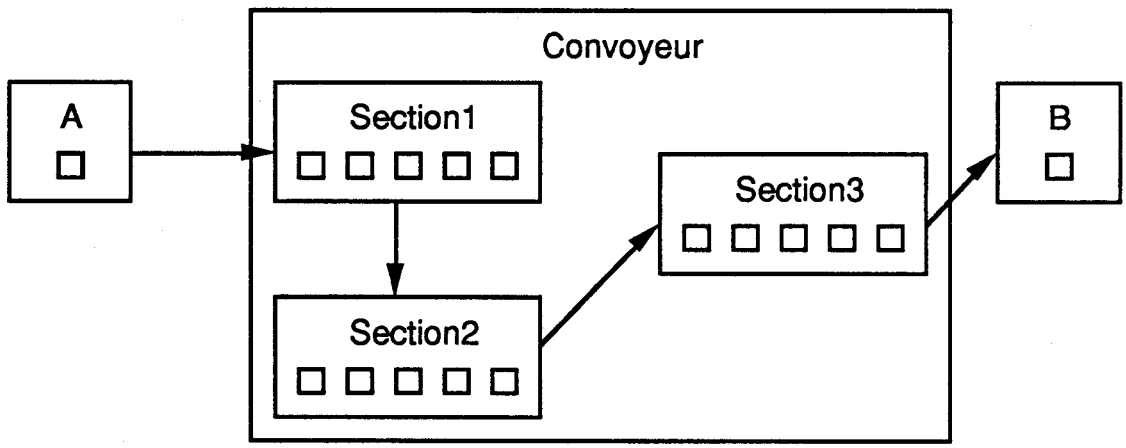


Figure 18

Puis à la troisième étape, chaque section est à nouveau détaillée : une section contient des rails et des aiguillages :

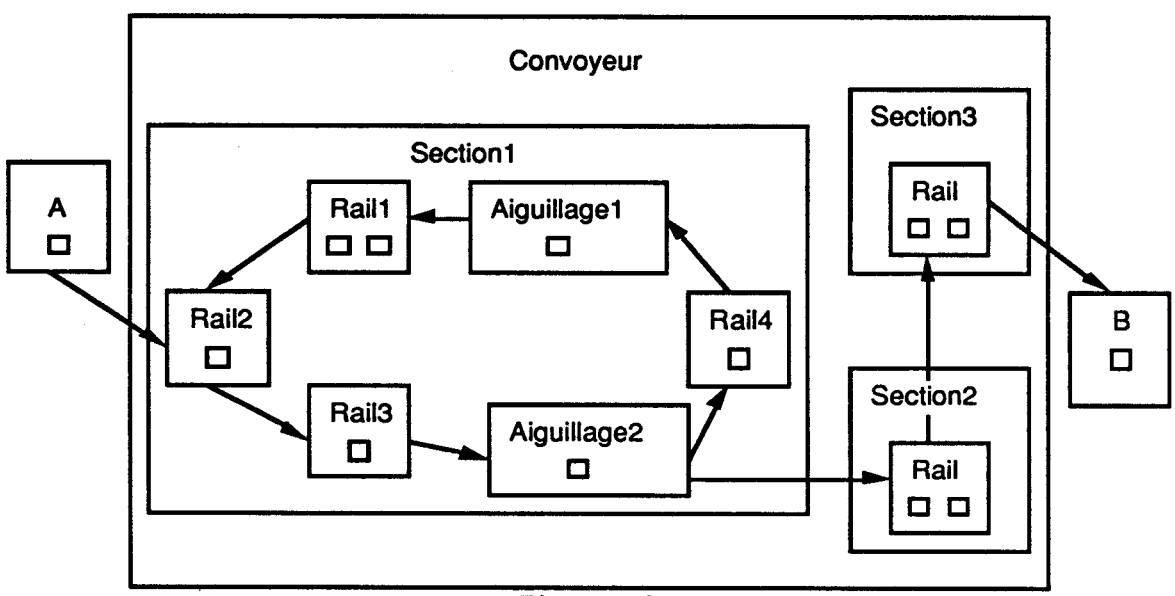


Figure 19

A la quatrième étape, soit les pièces se déplacent seules sur les rails, soit elles sont posées sur un support mobile (une palette). Les palettes peuvent être spécifiques à un type de pièce, ou polyvalentes. De même elles peuvent contenir un ou plusieurs objets simultanément.

Nous voyons donc sur cette présentation rapide, qu'à chaque étape de spécification des gammes opératoires nous prenons en compte un modèle du procédé affiné d'un niveau. Le procédé est donc décrit de manière arborescente.

II.4.5. Spécification des protocoles d'accès

Tout moyen physique de stockage a nécessairement une capacité limitée. La gestion de l'accès aux lieux physiques doit être explicitée. L'accès à un emplacement de stockage est réalisé en deux étapes : allocation puis accès réel. La procédure de retrait suit une séquence analogue : évacuation puis libération de l'emplacement. Ce mécanisme que nous appelons protocole d'accès à un lieu physique, est décrit sur le RdP modélisant les gammes opératoires. Notons que l'assignation d'un moyen (qui n'est pas un emplacement de stockage) à une pièce se fait de manière analogue (pince de soudage...). Ce modèle RdP est ensuite développé automatiquement pour fournir le graphe de commande du point de vue des pièces. C'est ce graphe qui sera effectivement implanté sur A.P.I.

II.4.6. Conception de l'Interface

Le graphe précédent n'est pas suffisant. En effet l'Interface, dont l'architecture et le fonctionnement sont en cours d'étude au laboratoire, doit être construit. Cependant sur des exemples courants sa définition ne pose pas de difficultés majeures. Nous en verrons un exemple au cours de notre exposé. Il est construit à partir de la définition du procédé et de la définition des stratégies de fonctionnement du procédé.

II.4.7. L'implantation

La dernière étape de la démarche concerne l'intégration, la simulation et l'implantation des modèles produits sur un support informatique et la répartition sur un ou plusieurs réseaux locaux informatiques. L'architecture informatique comporte en général :

- un organe de pilotage (micro ou mini ordinateur),

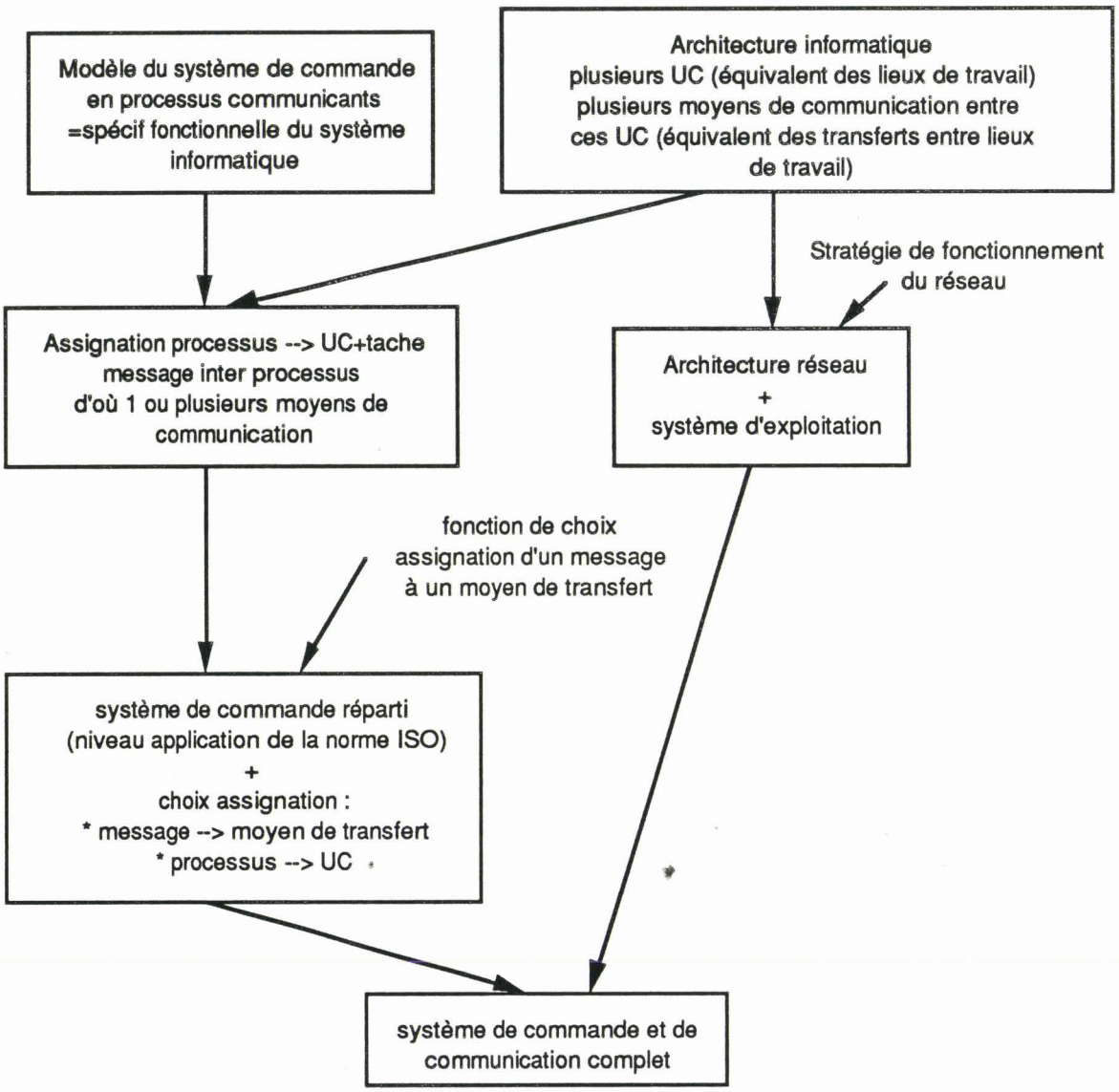
- un ou plusieurs réseaux locaux industriels,

- plusieurs automates programmables et armoires de commande numérique.

II.4.8. Le problème de la répartition

La répartition d'une commande sur un système réparti justifie à elle seule une étude approfondie. Elle remet en cause les travaux antérieurs établis au L.A.I.I. Elle présente des analogies importantes avec la synthèse des gammes opératoires à partir des gammes logiques et des moyens de production.

Le produit de la démarche CASPAIM-2 est un ensemble de processus de commande et de gestion communicants. Ce système est donc la spécification fonctionnelle du système informatique cible. On peut considérer un message atomique (une requête, un octet) comme l'analogue d'une pièce, et le réseau comme le support (partagé) du message. Implanter le système de commande revient à choisir les moyens de calcul et de communication physiques capables de recevoir le code produit. Il s'agit donc de la conception du système informatique cible selon le schéma suivant :



Conception de l'architecture de contrôle
et de commande informatique répartie hétérogène
Figure 20

Il est clair que la stratégie de fonctionnement du réseau (l'équivalent de la stratégie du fonctionnement du procédé) est immuable et rarement paramétrable. Elle est imposée par le choix du matériel. Notons que la communication par réseau est, dans la pratique le goulet d'étranglement des systèmes de contrôle répartis. Ce constat justifie qu'une étude de performance soit entreprise afin de vérifier que la dégradation des performances du système commandé induite par les réseaux reste admissible.

Conclusion

La démarche de conception présentée précédemment se veut pragmatique et très proche de la réalité industrielle. Il s'avère que la conception d'ateliers flexibles de production constitue un problème où les degrés de liberté

(spécifications, choix matériels) sont très nombreux. La démarche proposée a pour but principal de n'introduire les spécifications et les contraintes qu'au moment opportun, et par conséquent de ne réduire en aucune manière la flexibilité potentielle du processus de production.

CHAPITRE III

ANALYSE DU PROCEDE

Introduction

Le procédé, qui recouvre les moyens de production et de manutention et les objets produits, peut être d'une grande variété et d'un grand degré de complexité.

Dans le cadre de notre étude de conception, il conviendra de prendre en compte les caractéristiques du système de production. C'est pourquoi il nous semble important d'en effectuer une analyse approfondie. Nous proposons à cette fin une démarche visant à construire le modèle du procédé de manière simple et systématique. Ce modèle est exploité dans de multiples contextes et dans un grand nombre de phases de la démarche de conception de la Partie Commande d'un atelier flexible dans CASPAIM-2. C'est pourquoi son analyse doit être réalisée avec soin afin que le même modèle d'origine puisse être utilisé en conception, en simulation et en exploitation (maintenance, modes de marche), selon plusieurs formes dérivées (AMA 90).

III.1. Pourquoi son analyse ?

Nous avons vu au chapitre II que les principales limitations de la démarche CASPAIM-1 proviennent essentiellement d'une prise en compte peu systématique de l'aspect opératoire du processus commandé. L'exemple de l'ascenseur en est une illustration : il y a amalgame entre la notion de zone opératoire fixe, lieu de stockage fixe, moyen de transport, moyen de déplacement etc...

Le principal objectif de l'analyse d'un procédé réel est d'aboutir à un modèle fin et détaillé ne permettant aucune confusion de cet ordre.

Le second objectif est d'obtenir un modèle autonome dans le sens où il est strictement indépendant de l'exploitation qui en sera faite réellement lors de la phase de conception de la PC. Il est de ce fait réutilisable dans plusieurs contextes de commande, et souvent générique.

Le troisième objectif est d'aboutir à un modèle qui n'intègre que les contraintes opérationnelles strictement nécessaires (encombrement, collision, sécurité, zones à accès exclusif), et qui traduit donc au mieux la puissance et la flexibilité potentielle du moyen réel. En effet, un moyen, même de technologie simple, peut être d'une très grande flexibilité et d'une importante facilité d'utilisation.

Le modèle obtenu est naturellement exploité en phase de simulation afin de

vérifier le comportement global de l'installation.

Enfin le modèle que nous construisons sert de guide lors de la génération des gammes opératoires. En effet, celles-ci décrivent l'ensemble des opérations appliquées sur les zones opératoires. Il est donc logique de retrouver la description du procédé à ce moment-là.

De manière annexe, une base de modèles (génériques éventuellement) de procédés divers peut être très utile en phase de prototypage d'une installation, ou pour comparer les performances pures (ou théoriquement atteignables) de plusieurs moyens accomplissant les mêmes fonctionnalités.

L'idée maîtresse de cette démarche a été de se placer du point de vue du système réel, et de chercher à identifier l'ensemble de toutes les opérations élémentaires que peut réaliser simultanément un procédé donné, sans chercher à imposer un séquencement dans les opérations. Il s'agit donc de dresser le bilan de toutes les commandes élémentaires pouvant être réalisées en parallèle et indépendamment d'un séquencement logique, qui serait imposé par le traitement d'un objet (une pièce) par exemple.

III.2. La méthode de décomposition arborescente du procédé (CRU 90)

Le but de la démarche de conception CASPAIM est de concevoir la Partie Commande d'une unité de production. Le procédé doit donc être vu uniquement selon le point de vue commande.

Nous cherchons à identifier toutes les entités commandables du Procédé. Les premiers exemples sont immédiats : moteurs électriques, vérins hydrauliques, moteurs pas à pas etc ...

Il ne s'agit que d'actionneurs mécaniques. Leur fonction, pour ces exemples précis, est d'assurer un déplacement (longitudinal ou angulaire) à un objet sur lequel ils sont solidairement liés.

Prenons l'exemple d'un robot. Il possède une base qui est solidaire du sol. Le tronc est lié à la base par une liaison de type pivot. L'actionneur moteur pas à pas placé entre les deux (solidaire de la base ou du tronc selon la configuration) réalise une fonction de rotation verticale du tronc sur la base. Il est associé à un capteur de type incrémental afin d'assurer un asservissement en position angulaire correct.

De plus, le tronc est lui-même le support d'un autre axe : le bras. Ce bras est mu en rotation également au moyen d'un autre système moteur pas à pas

associé à un capteur de position angulaire.

Le tronc est donc vu selon deux angles :

- en tant qu'objet commandable en position par rapport à la base,

- en tant que support d'un autre axe de même type, donc d'un autre objet commandable.

III.2.1. Notion d'Objet Physique Commandable

Le but de l'analyse du procédé est d'identifier l'ensemble de ces objets, que nous appelons **Objets Physiques Commandables.**

De prime abord, les Objets Physiques Commandables sont tels que :

- soit ils sont mobiles sur leur objet support,

- soit ils sont le support permanent d'autres objets,

- soit ils sont le support occasionnel d'objets (pièces par exemple),

- soit toute combinaison des trois caractéristiques précédentes.

Cette notion de support d'objet permet de construire de proche en proche une **description arborescente** d'objets. Nous prenons la convention qu'un objet support est le père de l'ensemble des objets qu'il supporte. Ainsi le tronc du robot est le père du bras qu'il supporte.

Nous pouvons remarquer de plus que les objets peuvent être le support permanent d'autres objets (le bras du tronc), et également un support temporaire (une pince contient temporairement une pièce).

Notre objectif est de faciliter la définition de la commande.

Le fait qu'un objet soit en permanence sur son support signifie qu'il est inutile de gérer le chargement et le déchargement de cet objet. Ces opérations correspondent aux opérations de montage et de démontage de l'objet. En revanche, le fait qu'un objet supporte occasionnellement d'autres objets signifie qu'une commande est indispensable et qu'il faut la gérer.

Nous avons choisi de définir un objet particulier, appelé emplacement de stockage (ES), qui n'a pas de matérialité, mais qui englobe le lieu de l'espace dans lequel est placé l'objet stocké. L'objet ES est effectivement un objet commandé puisqu'il contient ou non un objet. Un ES est donc bien un Objet

Physique Commandable.

Tous les objets en transit sont donc logés dans des emplacements de stockage.

III.2.2. Arborescence d'Objets Monofonctionnels

Nous cherchons enfin à identifier des objets monofonctionnels de telle sorte qu'ils ne soient concernés que par un seul actionneur à tout instant.

Le modèle de description du procédé est donc une arborescence d'objets mono-fonctionnels supportant plusieurs objets.

Quelques fonctions d'objets ainsi que les moyens et applications sont donnés ci-après :

- stockage élémentaire (ES) temporaire d'un objet,

- positionnement linéaire (vérin, piston) pour robot portique, chariot,

- positionnement angulaire (moteur pas à pas) pour axe robot,

- asservissement en vitesse (moteur électrique) pour tournage, filetage, taraudage,

- asservissement en pression et position (vérin) pour serrage mandrin, presse hydraulique,

- asservissement en volume pour peinture, graissage,

- asservissement en débit (moteur électrique) pour liquide de lubrification et de refroidissement de tour.

La liste des fonctions élémentaires est ouverte.

III.2.3. Application à un exemple de chariot

Prenons un exemple simple. Il s'agit d'un chariot comportant 3 ES. Un des ES contient une pièce. Le chariot se déplace sur deux rails qui contiennent un ES chacun.

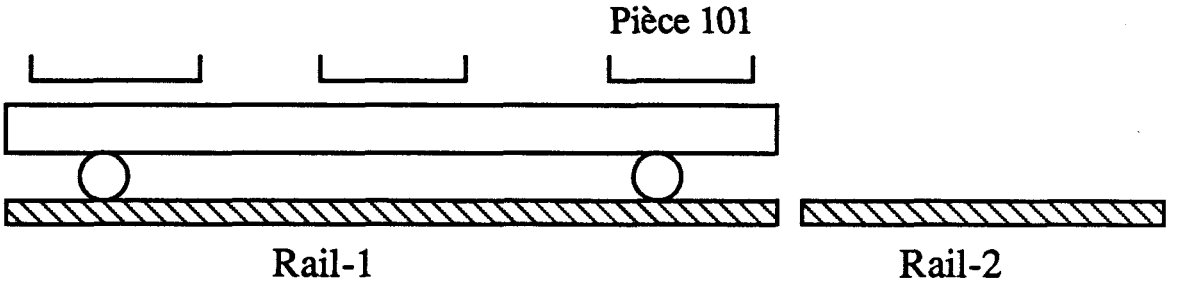


Figure 1

Le modèle arborescent de ce système est donc le suivant :

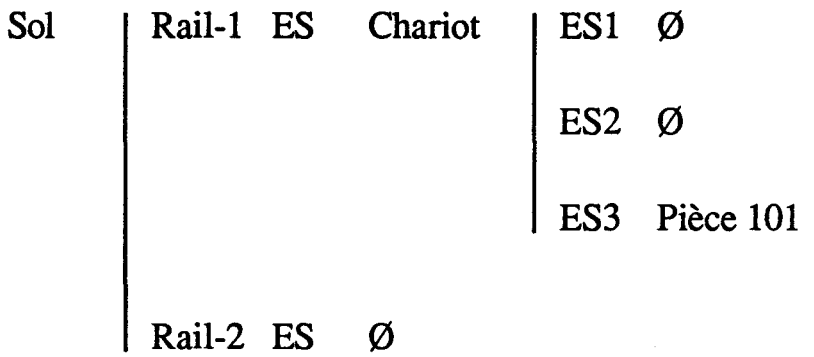


Figure 2

Chacun des rails ne possède qu'un ES. Ils sont donc gérés comme des ressources critiques : un seul objet peut se trouver sur un rail à un instant donné.

Cependant l'objet chariot, qui possède en permanence 3 ES, se déplace sur le rail sur lequel il est situé. Les ES du chariot sont quant à eux fixes sur le chariot, et supportent occasionnellement des pièces ou tout autre objet (un autre chariot ?).

De manière générale, l'identification des Objets Physiques Commandables doit mener à des objets qui :

- sont mobiles sur leur support, et contiennent en permanence le même nombre d'objets (le chariot contient en permanence 3 ES et rien d'autre),

- sont fixes sur leur support et contiennent 0 ou 1 objet (les ES).

Les objets accomplissent donc exclusivement une fonction de positionnement longitudinal (le chariot) ou une fonction de stockage (ES). La fonction de stockage permanent est de ce fait ignorée puisqu'elle n'induit pas de gestion de commande.

La méthode d'analyse d'un procédé aboutit donc à rechercher l'ensemble

des objets monofonctionnels constitutifs.

III.2.4. Application à un tour à commande numérique (CRU 90)

Considérons un deuxième exemple. Il s'agit d'un Tour à commande numérique. Bien que ces systèmes possèdent leur propre logique de commande (NUM 760), ils sont intéressants à analyser car complexes. Ils sont composés d'un grand nombre d'actionneurs et de capteurs. Il s'agit d'un tour Ernault-Somua HES 400.

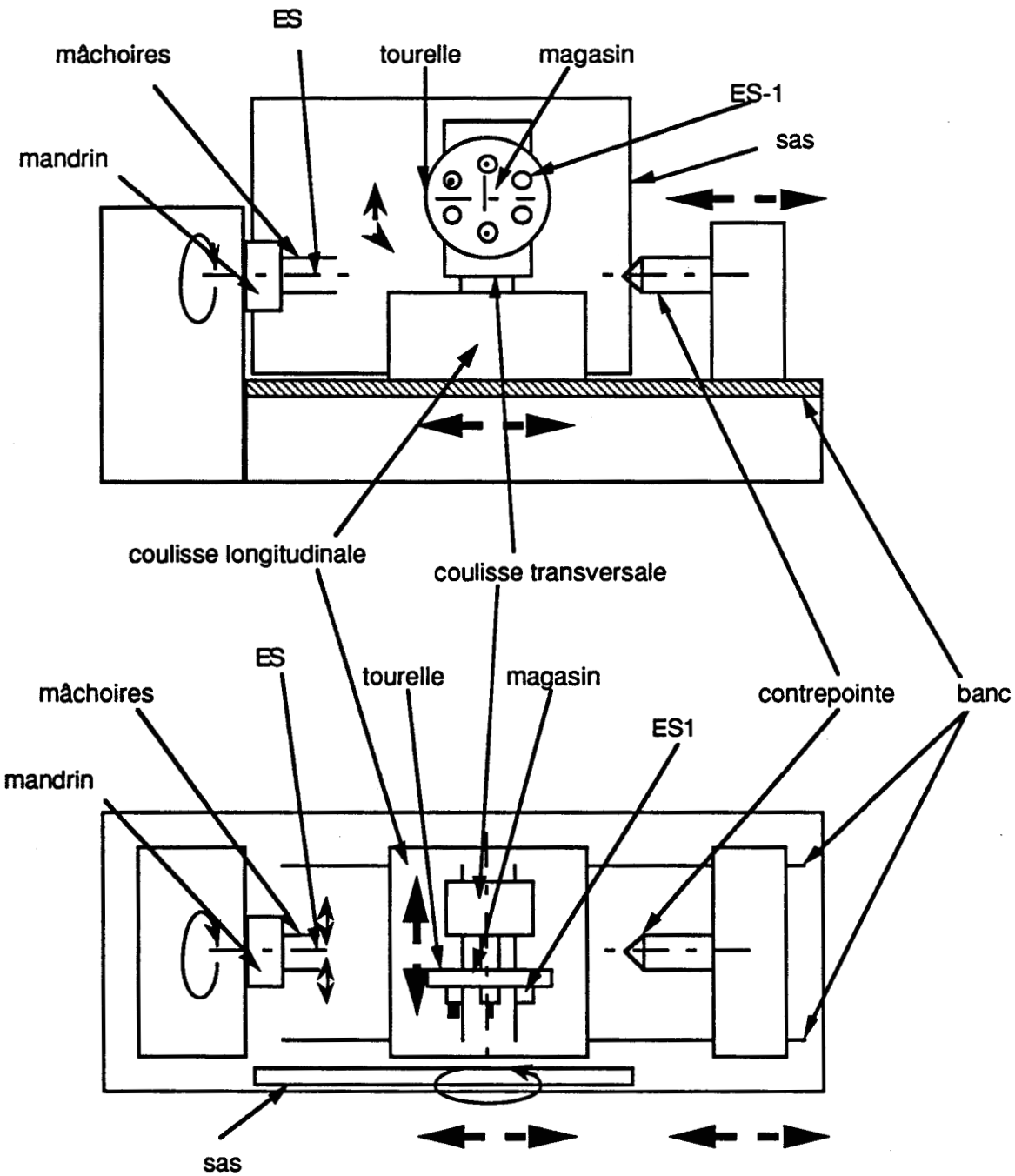


Figure 3

La décomposition structurelle arborescente de ce tour permet de créer le modèle statique suivant :

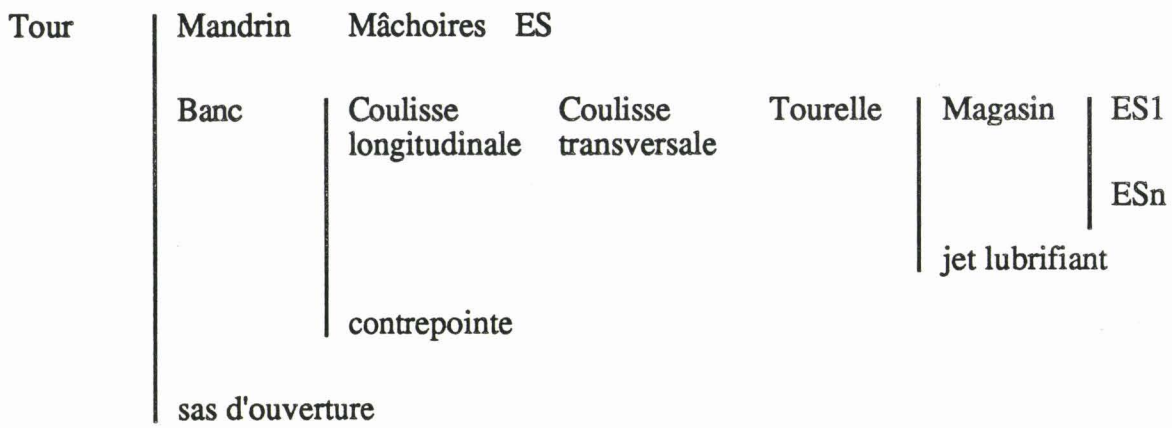


Figure 4

Chacun des objets que nous avons identifiés répond à la définition précédente d'un Objet Physique Commandable.

. Nous précisons ensuite **la** fonction de chaque objet, ainsi que le moyen associé pour réaliser la fonction (actionneur + capteur).

Objet	Fonction	Moyens
Tour	/	/
Mandrin	Asserv en vitesse / Tour	Moteur + Tachymètre
Mâchoires	Asserv en pression et position	Vérin + Manomètre
Banc	/	/
Coulisse longitudinale	Asserv en position / Banc	Vérin + capteur position
Coulisse transversale	Asserv en position / Coulisse long	Vérin + capteur position
Tourelle	Asserv en position angulaire / Coulisse transv	Vérin + capteur position
Magasin	/	/
Jet lubrifiant	Asserv en débit	Pompe électrique + débit mètre
Contrepointe	Asserv en position / Banc	Vérin + capteur position
sas d'ouverture	Asserv en position / Tour	Vérin + capteur position

Figure 5

A partir de ce tableau, nous pouvons faire plusieurs remarques :

- chaque objet évolue indépendamment de son support et de ce qu'il contient. Ainsi la coulisse transversale a un mouvement totalement indépendant de la coulisse longitudinale (son père) et de la tourelle (son fils). Elle possède son propre système de commande et d'asservissement. Du point de vue commande, ce tour est donc vu comme une collection d'objets commandés séparément et indépendamment l'un de l'autre ;

- la fonction de tournage n'est pas une fonction élémentaire. Elle n'est que la synthèse de toutes les fonctions des objets pris individuellement, et qui évoluent de manière cohérente. De même nous faisons totalement abstraction de la logique de fonctionnement du tour, à savoir du séquencement normal des opérations : ouverture du sas et du mandrin, chargement du mandrin, fermeture du sas, tournage etc ... ;

- on se limite au strict point de vue des constituants matériels et de leur fonction afin de ne réduire en rien le potentiel de flexibilité latent du système. Un exemple de flexibilité concerne le magasin d'outils. En effet, prenons l'hypothèse que le sas est ouvert et le mandrin arrêté (obligatoire si le sas est ouvert). Du fait que tous les ES du magasin de pièces sont accessibles (voir le magasin Fig 3), il est possible de les manipuler simultanément (par plusieurs robots par exemple). Les mâchoires sont également accessibles dans les mêmes conditions. Cet exemple montre bien qu'il faut partir de cette description matérielle et du potentiel de parallélisme qui lui est attaché afin de ne pas se laisser influencer par l'image purement séquentielle que l'on se fait de sa manipulation.

Le rôle du concepteur est d'exprimer au mieux le potentiel de commande d'un système. Nous avons donc pris le parti d'analyser le procédé en envisageant le traitement potentiel de parallélisme maximal. C'est pour cette raison qu'il est indispensable de descendre à un niveau d'analyse aussi fin. En effet, le parallélisme maximal correspond exactement à l'évolution simultanée et indépendante de **tous** les actionneurs identifiés par l'analyse du procédé sans contrainte à priori.

III.2.5. Modèle arborescent de l'exemple de la cellule

L'exemple de la cellule que nous avons choisi est composé des éléments suivants :

- une zone de stockage fixe,

- un tour à commande numérique,

- une taraudeuse,

- un robot à un préhenseur appelé Robot-1,

- un robot à deux préhenseurs indépendants appelé Robot-2,

- un robot graisseur muni d'un pistolet à graisse,

- une machine d'assemblage,

- un convoyeur muni d'un grand nombre d'aiguillages et de rails, sur lesquels transitent des palettes à plusieurs emplacements de stockage. Une zone du convoyeur est utilisée comme stock de palettes vides.

Un exemple industriel de robot multibras est présenté dans ALA 90.

Les positions relatives sont grossièrement les suivantes :

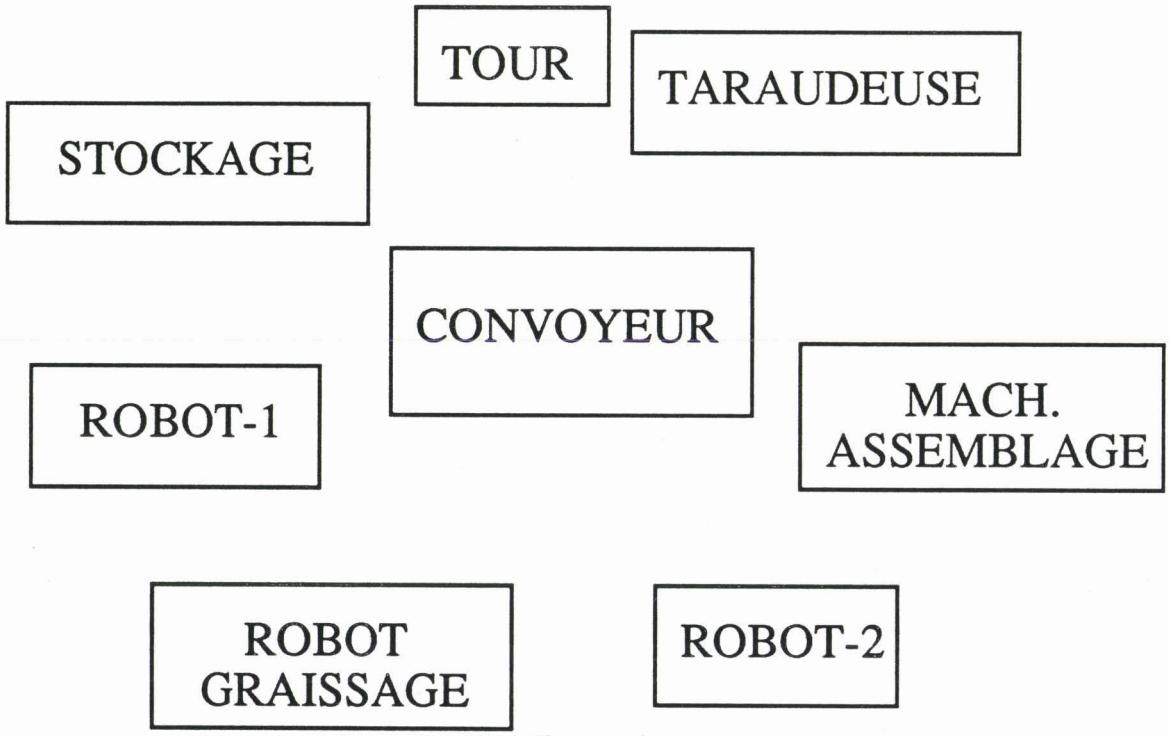


Figure 6

Chacun de ces moyens est ensuite détaillé par étapes pour aboutir à l'arborescence suivante :

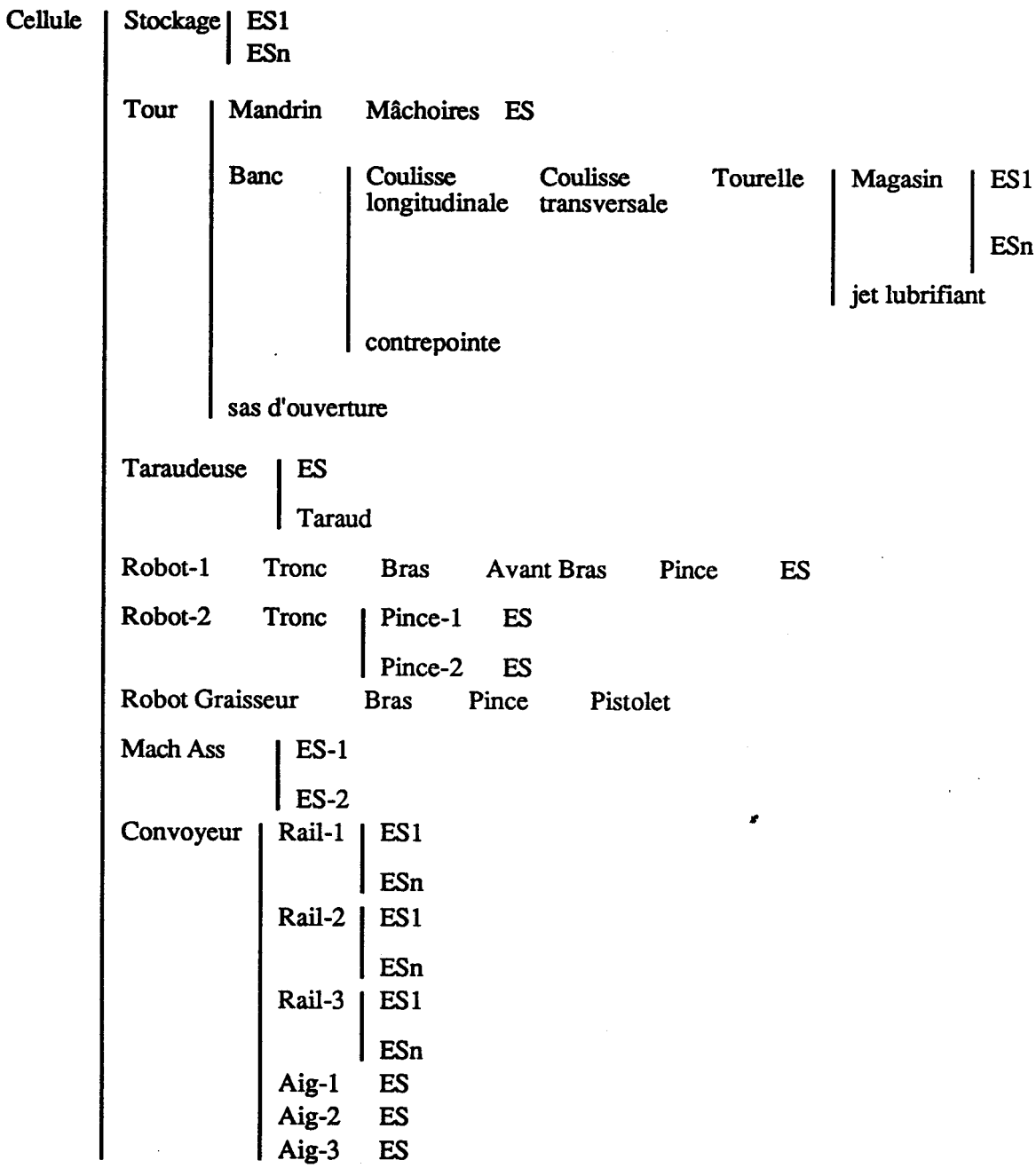


Figure 7

Cette arborescence est présentée sous une autre forme sur la figure suivante :

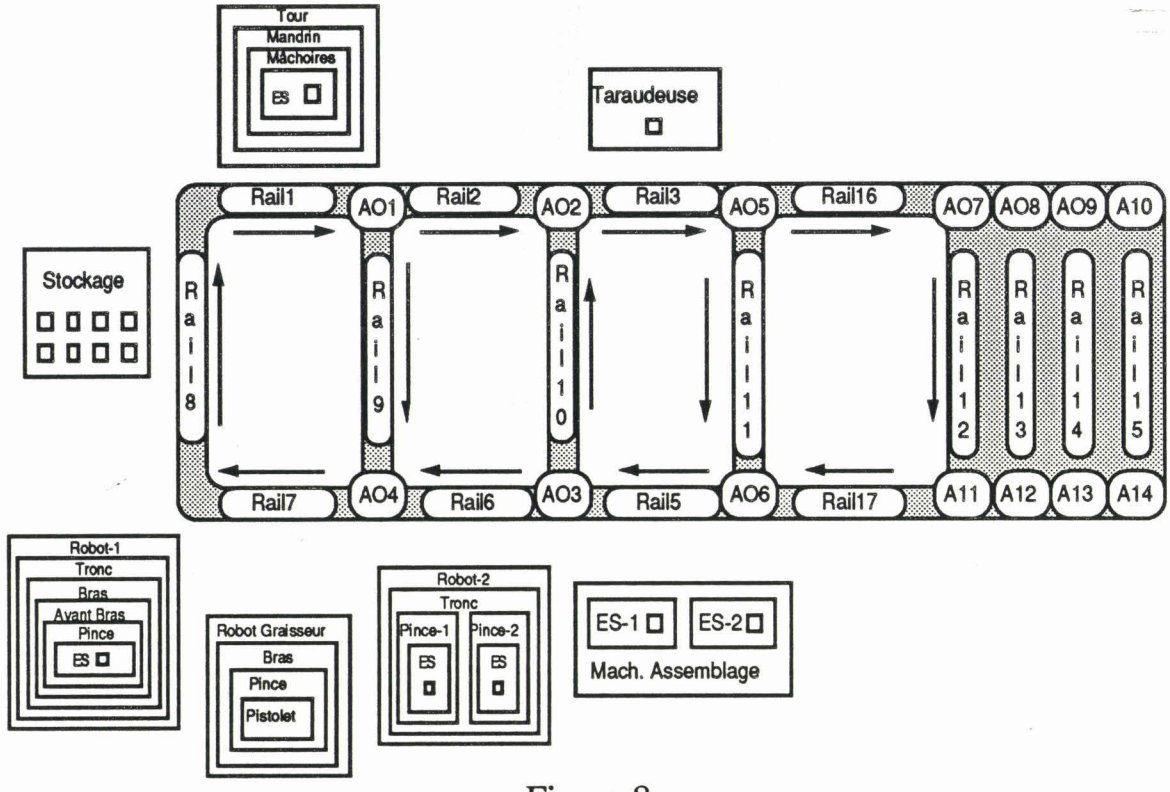


Figure 8

Le tour à commande numérique a été détaillé en début de section.

La taraudeuse est composée d'un support de pièce et d'un taraud assurant l'opération de taraudage.

La machine d'assemblage est composée de deux ES servant à loger les deux pièces à assembler.

Les fonctions de ces objets sont principalement :

- asservissement en position ou en vitesse,

- asservissement en volume (pistolet à graisse),

- stockage élémentaire.

Ce modèle statique va être utilisé au chapitre IV pour concevoir étape par étape le modèle de la commande de cet atelier, dans le cadre d'une production de boulons à partir de vis et écrous bruts de fonderie.

Nous allons maintenant préciser la dynamique du modèle créé ainsi que la notion de contrainte opérationnelle.

III.3. Filtres de commande mutuellement contraints

Chaque objet physique commandable évolue en fonction des commandes que reçoit l'actionneur qui lui est associé. Considérons par exemple le mandrin du tour. Il est mis en rotation par un moteur électrique. La coupe se fait à vitesse constante (vitesse qui dépend du matériau), par conséquent la vitesse de rotation du mandrin change en fonction du diamètre de la passe.

Du point de vue commande, il s'agit donc de réaliser un asservissement en vitesse du mandrin.

Nous avons pris comme hypothèses au chapitre II que le procédé est totalement assujetti aux ordres transmis par la PC (via l'Interface). De plus nous assimilons tout procédé à un système discret, soit pour lequel on ne considère que l'état discrétisé, même si l'état réel est par essence continu. Pour l'exemple du mandrin, nous devons donc expliciter ses états discrets ainsi que les commandes qui font évoluer l'image discrète de son état.

III.3.1. Notion de filtre de commande

L'image procédé dont nous avons besoin doit nous renseigner sur l'état courant du moyen moteur. Nous supposons que le module d'asservissement en vitesse (électrique par rhéostat, électronique par triac, numérique, PID, etc...) existe et qu'il est capable de répondre à la consigne en vitesse. Du fait de l'inertie mécanique, le changement de consigne induit un transitoire et donc des états transitoires.

Nous cherchons à concevoir un modèle discret du système continu composé du moteur et du tachymètre. Ce modèle, appelé Filtre de commande, se situe entre le module Interface et le procédé réel (commande de PID ...).

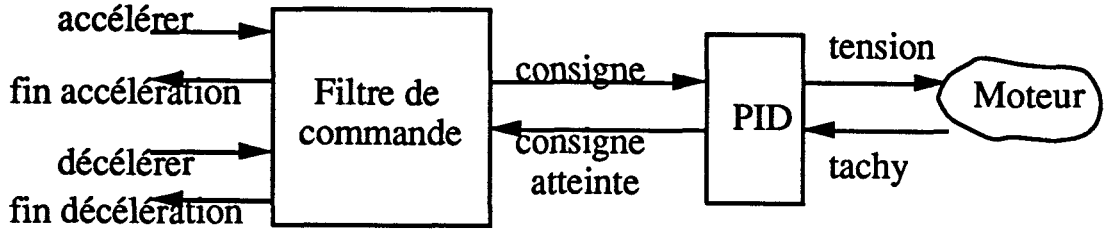


Figure 9

Son rôle est de garantir que les commandes transmises au procédé sont autorisées vis à vis du procédé (par exemple, ne changer de consigne que si la vitesse est stabilisée), mais aussi vis à vis des autres objets commandés (notion de contrainte explicitée plus loin).

Considérons le cas simple où le moteur passe de l'arrêt à une vitesse constante, puis à nouveau à l'arrêt.

Nous commençons par décrire le comportement normal du moteur. Le moteur possède 4 états discrets : A l'arrêt, En accélération, En rotation (à vitesse constante), En décélération.

L'image du procédé moteur dans le filtre de commande évolue en fonction des signaux reçus par le filtre, à savoir accélérer, décélérer jusqu'à l'arrêt et consigne atteinte que l'on interprète comme une fin d'accélération ou une fin de décélération.

L'automate traduisant l'évolution de l'image du procédé est donc le suivant :

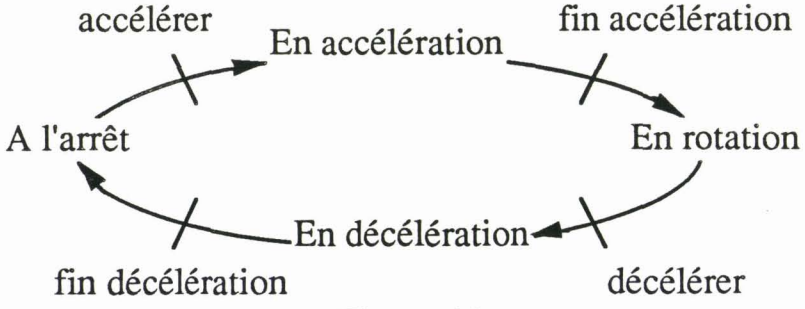


Figure 10

Cet automate est exploité en tant que filtre de commande dans le sens où les commandes reçues ne sont transmises que si l'état (discret) le permet. De ce fait, il est également possible d'identifier les commandes erronées pour chaque état. Une erreur de commande déclenche un signal erreur_commande et l'automate n'évolue pas. L'automate de départ est rendu déterministe par ce biais.

Le filtre de commande obtenu est donc le suivant :

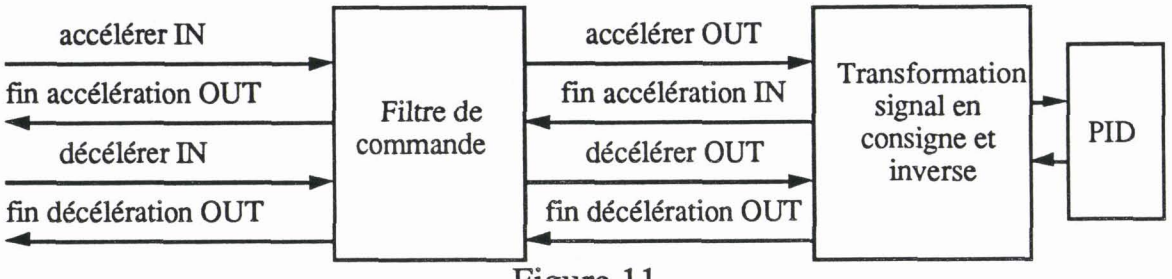


Figure 11

La transformation d'ordre en consigne consiste à traduire un ordre discret (valué éventuellement) en une consigne physique.

Le filtre de commande possède en interne une image discrète de l'état continu du moteur et de sa logique de commande associée (PID).

Il est constitué de l'automate déterministe suivant :

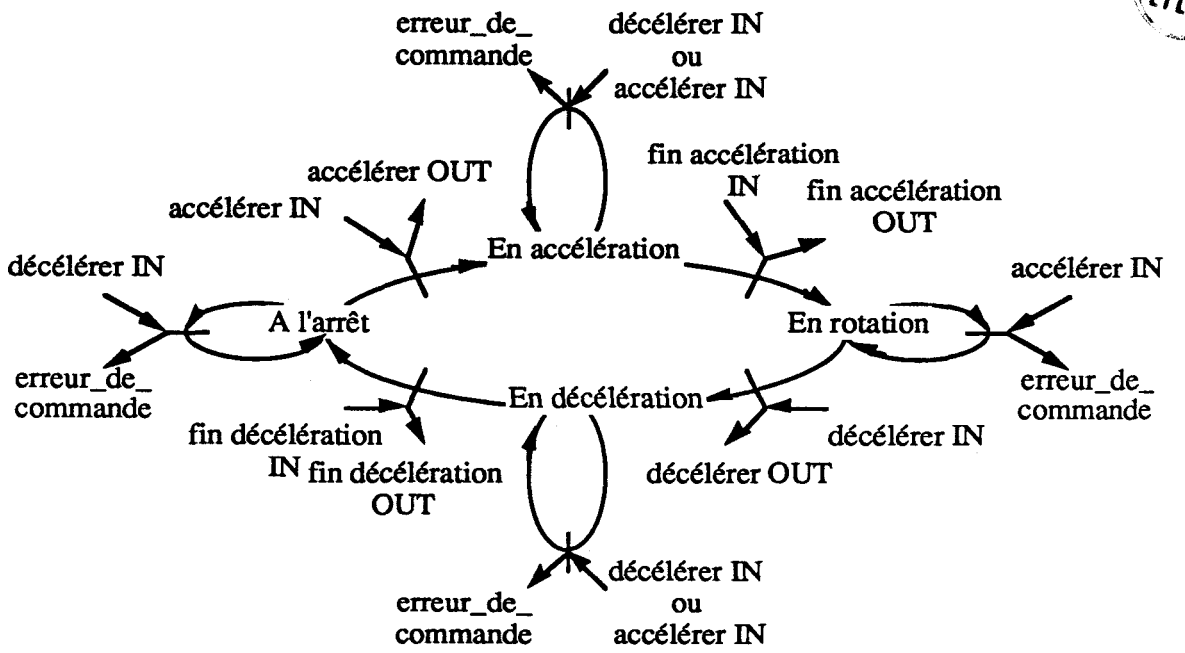


Figure 12

Un filtre de commande est donc un système de sécurité temps réel dans le sens où la commande est contrôlée en temps réel par rapport à l'état courant du système commandé.

La gestion des erreurs de commande consiste donc simplement à rendre déterministe l'automate de fonctionnement normal (Figure 10). Nous ne considérons donc à présent que cet automate de fonctionnement normal puisqu'il permet de générer automatiquement le filtre de la Figure 12.

Notons de plus que la notion de filtre utilisée au CRAN de Nancy est proche de la notion de Boîte Fonctionnelle (TIX 89, MOR 88, LHO 88).

Une Boîte Fonctionnelle (BF) est composée de :

- une interface, comprenant des variables d'entrée et de sortie,

- un ensemble de variables locales, rémanentes,

- un corps exécutable qui calcule les sorties en fonction des variables d'entrée et des variables rémanentes.

La BF est, de même que nos filtres, l'élément logiciel qui contrôle

effectivement les actionneurs et capteurs physiques (commandes d'axes, sortie TOR, entrées binaires etc).

De plus une BF possède en interne une image des pièces manipulées par les actionneurs.

Les BF servent donc à la fois au contrôle temps réel des actionneurs et d'organe de contrôle des flux d'informations relatives aux objets manipulés (plateaux, pièces etc).

Ce que nous appelons Partie Commande, qui gère le séquencement des opérations sur chaque pièce, est mélangé au contrôle commande temps réel dans la même notion de BF.

Pour notre part, nous avons choisi de modéliser chaque partie du contrôle au moyen de l'outil le plus approprié, à savoir des automates mutuellement contraints au niveau de contrôle temps réel, assujettis à une commande de coordination et de synchronisation à base de RdP.

De la sorte, le modèle de commande fine (automates) est **indépendant** du niveau de coordination, et pour une large part générique et réutilisable. Ainsi, le filtre du moteur que nous venons de détailler est utilisé pour chaque moteur contrôlé, et est décrit indépendamment de l'objet qui est mis en rotation. En effet, il est évident que le comportement du mandrin (la rotation) ne dépend pas de son chargement. Il évolue de la même façon à l'état chargé ou vide.

Encore une fois, on se place dans l'hypothèse selon laquelle on ne connait pas l'utilisation effective des moyens et qu'il convient de conserver le maximum de leur flexibilité potentielle.

Le modèle de comportement d'un emplacement de stockage tel que celui qui est logé dans les mâchoires du mandrin est le suivant :

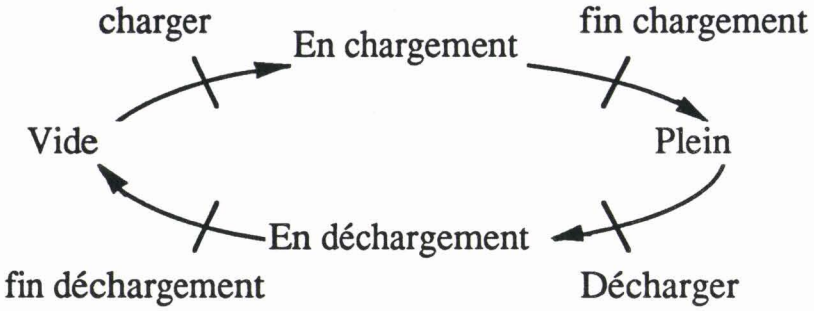


Figure 13

Tous les ES ont le même type de comportement.

Ainsi, la zone de stockage de notre exemple d'atelier est composé de n ES.

Ils sont tous indépendants entre eux. En effet, le chargement de l'un n'empêche en rien le déchargement de son voisin. Cette zone de stockage fixe a donc un potentiel de flexibilité très important : les n ES peuvent évoluer en parallèle et de manière totalement indépendante. Nous appelons ce type de stockage un VRAC par opposition aux PILES et aux FILES pour lesquels seuls 1 ou 2 ES peuvent être manipulés à la fois.

III.3.2. Contrainte opérationnelle

Considérons à nouveau le mandrin du tour. Si l'on suppose que le moteur et l'ES évoluent librement, il est possible que le moteur soit en rotation alors que l'ES est en cours de déchargement. Dans ce cas, bien sûr, il y a risque de projection et d'accident.

De manière générale, si l'on suppose que tous les objets évoluent en parallèle et librement, il est fort probable que l'on atteigne des configurations de collision, de projection, de casse.

Dans ce sens les contraintes opérationnelles permettent de garantir un comportement sain du procédé. L'hypothèse de parallélisme maximal, qui consiste à laisser évoluer simultanément tous les Objets Physiques Commandables, est en fait contrainte et ne prend en compte que le strict minimum de contraintes opérationnelles. Ces contraintes ne sont issues que de considérations technologiques de type collision, encombrement, sécurité etc... et ne concerne en aucun cas le séquencement réel des opérations sur une pièce par exemple. Là encore on souhaite conserver le maximum de flexibilité potentielle.

Dans le cas du tour, quelques exemples de contraintes entre objets sont les suivantes :

1. mandrin à l'arrêt lors du chargement, déchargement des mâchoires,

2. mandrin, porte outil et contre-pointe à l'arrêt lors de l'ouverture du sas,

3. mandrin et porte outil à l'arrêt lors de la manœuvre de la contre-pointe,

4. contre-pointe à l'arrêt lors de la manœuvre du mandrin et du porte-outil,

5. porte-outil en position de repli lors du changement d'outil,

6. le porte-outil et la contre-pointe doivent être distants de x cm au moins.

Ces contraintes sont donc de nature très variée et font intervenir un grand

nombre d'objets physiques à la fois. En effet, imposer que le porte-outil soit à l'arrêt implique que l'on contrôle simultanément :

la coulisse transversale,

la coulisse longitudinale,

et la tourelle.

Tout en sachant que ce contrôle temps réel est géré directement par la commande numérique, nous allons étudier en détail la première contrainte :

le mandrin doit être à l'arrêt lors du chargement et du déchargement des mâchoires.

En effet, il est courant qu'un concepteur ait à gérer les commandes d'un procédé axe par axe. La définition de ce contrôle temps réel est alors à sa charge. Il est indispensable de lui fournir les outils de conception de ce contrôle.

III.3.3. Expression des contraintes inter objets sur chaque filtre

Le fait que le mandrin et l'ES des mâchoires soient liés par une contrainte signifie qu'il est indispensable de contrôler simultanément leurs états et leurs commandes. Le premier objet ne peut évoluer que si le second est dans l'état adéquat.

Considérons un objet fictif Proc composé des deux objets ES et moteur. L'état de Proc est donc le couple *(état ES, état moteur)*, et ses commandes sont les couples *(commande ES, commande moteur)*. Pour l'objet Proc, la commande *(charger, Ø)* traduit le fait que le moteur n'est pas commandé, et que l'ES reçoit la commande *charger*.

Le graphe d'état de Proc est le produit cartésien des deux graphes d'état du moteur et de l'ES. Il s'agit donc du graphe suivant :

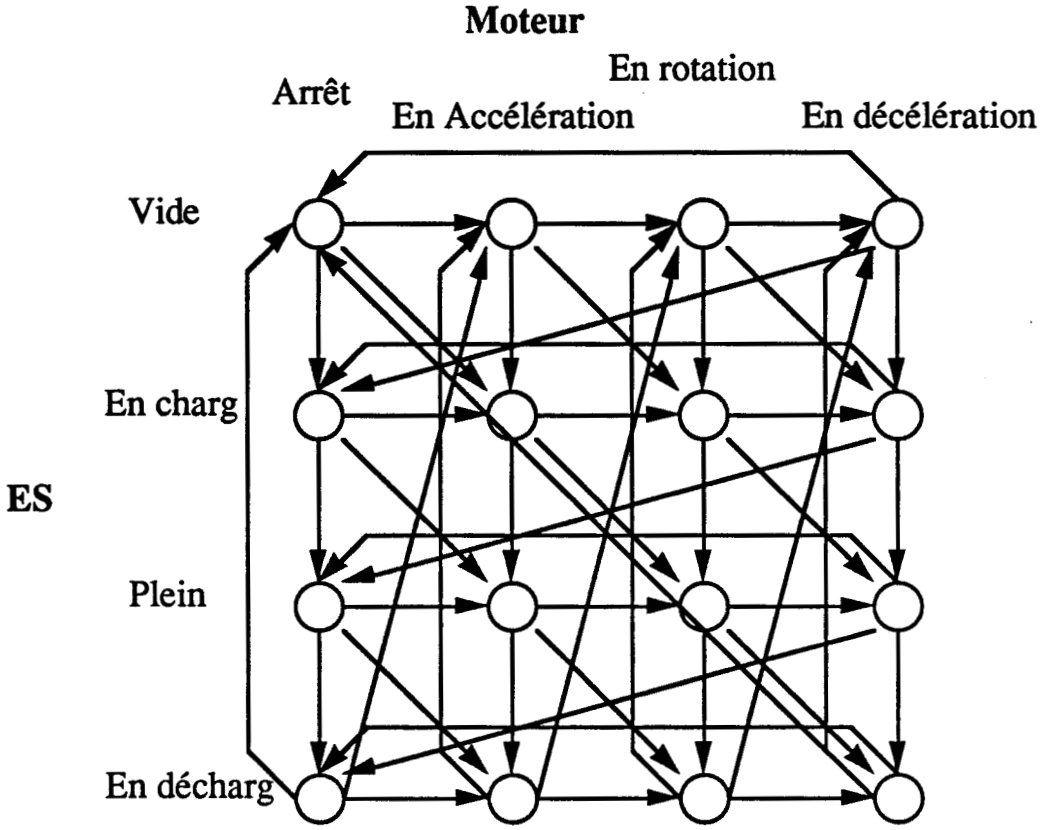


Figure 14

L'état de Proc *(Vide, Arrêt)* peut donc conduire à l'état

(Vide, En accélération) par la commande *(Ø, Accélérer)*,
(En chargement, Arrêt) par la commande *(Charger, Ø)*,
(En chargement, En accélération) par la commande *(Charger, Accélérer)*.

La contrainte entre l'ES et le moteur se traduit par le fait qu'un certain nombre d'états de l'objet fictif Proc ne doivent jamais être atteints. Il s'agit des 6 états :

(En chargement, En accélération),
(En chargement, En rotation),
(En chargement, En décélération),
(En déchargement, En accélération),
(En déchargement, En rotation),
(En déchargement, En décélération).

On élimine dans ce sens les commandes susceptibles de permettre d'atteindre ces états.

Les seules évolutions autorisées de l'objet fictif Proc sont donc les suivantes :

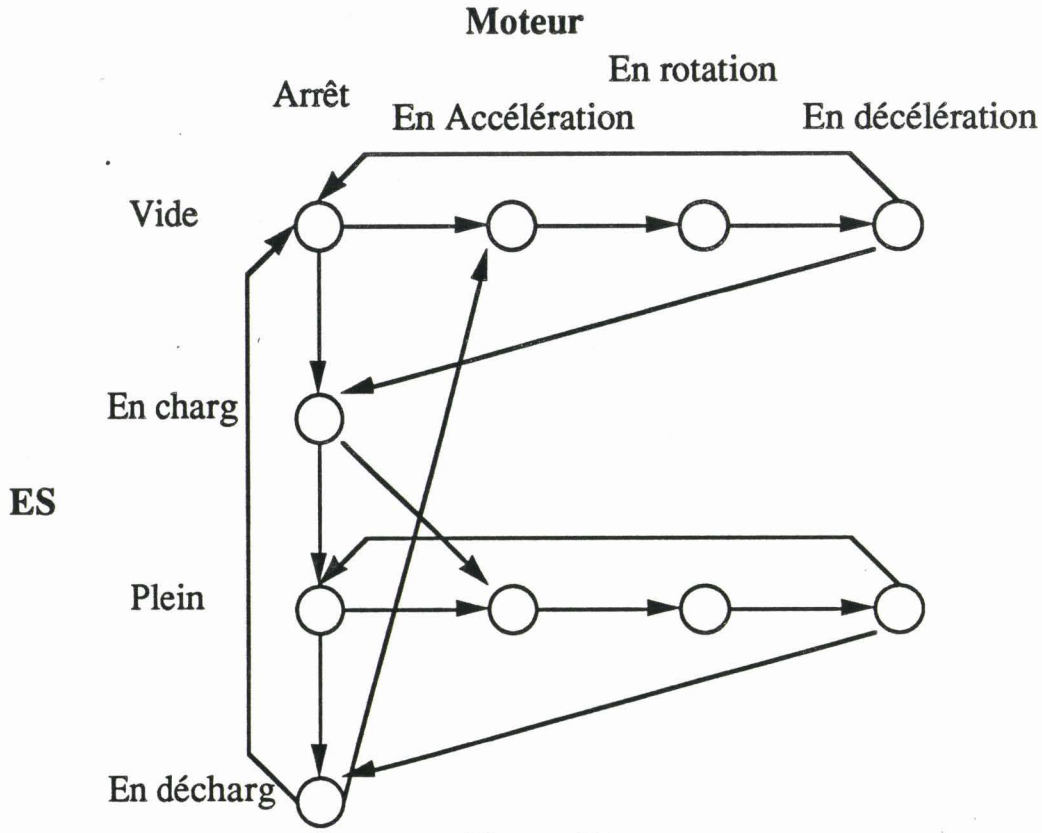


Figure 15

Ce modèle est suffisant pour garantir un comportement satisfaisant des deux objets moteur et ES. Cependant du point de vue de l'implantation, il est beaucoup plus intéressant et pratique de conserver une certaine autonomie de chacun des objets. En effet, à un graphe d'état élémentaire est associé un actionneur et non deux comme sur le modèle généré.

Nous cherchons donc maintenant à exprimer les contraintes sur le graphe d'évolution de chacun des objets.

Pour cela il suffit d'ajouter des conditions de franchissement c_i à chaque transition de chaque automate élémentaire. Les conditions c_i d'un objet j feront référence à l'état et aux commandes reçues par tous les objets autres que j et qui sont liés à l'objet j par au moins une contrainte.

Les contraintes sont localisées de la manière suivante :

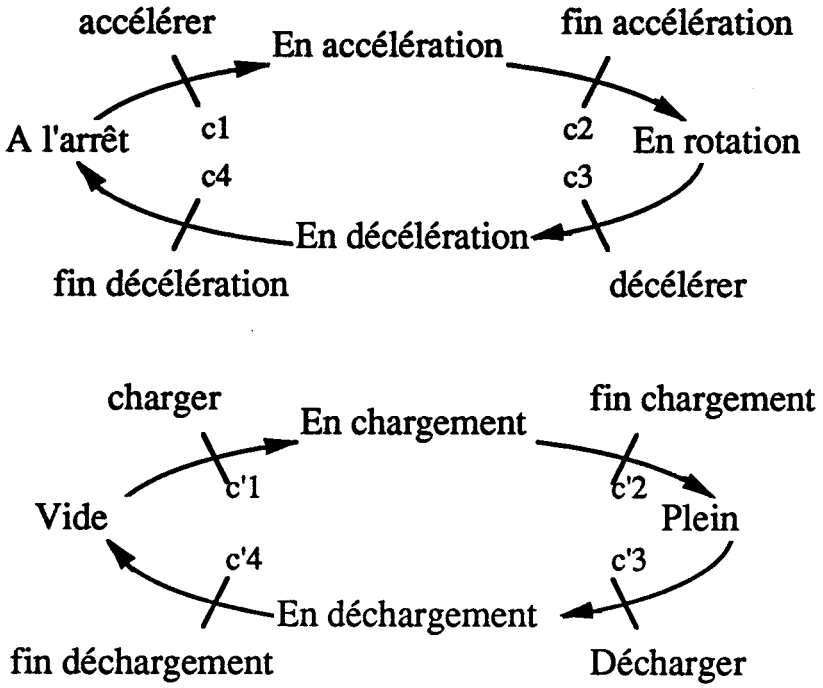


Figure 16

La contrainte c1 doit être satisfaite pour que l'ES puisse recevoir la commande *Charger*.

En consultant l'automate de l'objet fictif Proc, l'ES peut passer de l'état *Vide* à l'état *En Chargement* à condition que :

le moteur soit à *l'arrêt* et ne reçoive aucune commande,
ou que le moteur soit dans l'état *En décélération* et reçoive la commande *Fin décélération*.

Toutes ces contraintes sont construites en consultant l'automate de l'objet fictif Proc.

Nous obtenons les contraintes suivantes :

concernant l'**EMPLACEMENT**:

c1 = (etat-moteur=*arrêt*,cde-moteur=Ø)
v (etat-moteur=*en décé*,cde-moteur=*fin décé*)
c2 = (etat-moteur=*arrêt*,cde-moteur=Ø)
v (etat-moteur=*arrêt*,cde-moteur=*démarrer*)
c3 =(etat-moteur=*arrêt*,cde-moteur=Ø)
v (etat-moteur=*en décé*,cde-moteur=*fin décé*)
c4 =(etat-moteur=*arrêt*,cde-moteur=*démarrer*)
v (etat-moteur=*arrêt*,cde-moteur=Ø)

concernant le **MOTEUR:**

c'1 = (etat-empl=*plein*,cde-empl=Ø)
v (etat-empl=*en déch*,cde-empl=*fin déch*)
v (etat-empl=*en ch*,cde-empl=*fin ch*)
v (etat-empl=*vide*,cde-empl=Ø)
c'2 = (etat-empl=*vide*,cde-empl=Ø)
v (etat-empl=*plein*,cde-empl=Ø)
c'3 = (etat-empl=*vide*,cde-empl=Ø)
v (etat-empl=*plein*,cde-empl=Ø)
c'4 = (etat-empl=*vide*,cde-empl=Ø)
v (etat-empl=*plein*,cde-empl=Ø)
v (etat-empl=*vide*,cde-empl=*charger*)
v (etat-empl=*plein*,cde-empl=*déch*)

Les filtres de commandes initiaux ont donc été enrichis afin de prendre en compte leurs contraintes mutuelles.

Nous contribuons ainsi de manière conséquente à la sécurité de fonctionnement du moyen, aussi bien du point de vue mécanique que du point de vue de la sécurité des manipulateurs humains.

Nous avons présenté une méthode systématique conduisant à une commande fine sûre. Sa conception procède d'une démarche simple, rigoureuse et facilement automatisable notamment en ce qui concerne la définition des expressions des contraintes c_i de chaque objet. Seule la spécification des états interdits est laissée au soin du concepteur.

Le seul handicap reste la taille des expressions booléenne c_i lorsque ne nombre d'objets contraints est important. Nous considérons cependant qu'il ne s'agit pas d'un obstacle significatif, compte tenu de la sécurité apportée.

III.3.4. Critère de répartition des filtres de commande sur API

. Dans la mesure où deux objets sont contraints, leurs filtres de commande doivent évoluer de manière synchrone. Il faut donc en particulier que les automates (traduits en GRAFCET) soient implantés sur le même calculateur ou API.

Ce critère de synchronicité induit un critère de répartition du contrôle temps réel.

Ainsi, les contraintes que nous avons énumérées pour le tour à commande numérique (cf §III.3.2.) nous permettent de dire qu'il est nécessaire pour de simples raisons d'efficacité de regrouper les filtres de commande des objets suivants :

1. mandrin, mâchoires

2. mandrin, coulisse transversale, coulisse longitudinale, tourelle, contre-pointe, sas

3. mandrin, coulisse transversale, coulisse longitudinale, tourelle, contre-pointe

4. id

5. coulisse transversale, coulisse longitudinale, tourelle

6. contre-pointe, coulisse transversale.

Nous constatons sur cet exemple que tous les objets qui constituent le tour sont contraints entre eux par transitivité. Il est donc peu judicieux de répartir le traitement temps réel de ces objets. Ils doivent être commandés de manière cohérente par un seul processeur de contrôle.

De manière générale, la définition des contraintes inter objets est d'une grande importance puisqu'elle permet de regrouper l'ensemble des objets fortement dépendants.

A l'inverse, deux ensembles d'objets qui ne possèdent aucune contrainte entre eux peuvent être répartis sur deux systèmes de contrôle informatique distincts.

Ce critère est donc à prendre en compte lors de la répartition informatique que nous avons présenté au paragraphe II.4.8.

Nous allons identifier ces contraintes sur l'exemple d'atelier flexible présenté en début de chapitre.

III.3.5. Application à l'exemple

Nous proposons tout d'abord d'analyser les contraintes internes à chaque machine, puis dans un second temps, nous étudierons les contraintes entre machines.

III.3.5.1. Contraintes internes aux machines

Le **Stockage** est composé d'ES strictement indépendants entre eux (VRAC d'ES). Il n'existe donc pas de contraintes mutuelles entre eux.

Le **tour** a été présenté en détail dans la section III.3.2. Tous les objets qui le composent doivent être gérés de manière synchrone.

La **taraudeuse** est composée d'un ES et d'un taraud qui doivent également être commandés de manière synchrone.

Le **robot-1** possède une pince qui ne peut être manipulée que si l'ensemble des axes est à l'arrêt. Tous les objets du Robot-1 sont donc gérés de manière synchrone.

Le **Robot-2** est analogue au Robot-1. Cependant les deux pinces sont strictement indépendantes. Elles sont commandées indépendamment l'une de l'autre mais chacune d'elle est contrainte du fait de sa liaison avec l'objet Tronc.

Le **robot graisseur** : cas analogue au Robot-1.

La **machine d'assemblage** est composée de 2 ES indépendants (VRAC) donc non contraints.

Le **convoyeur** est composé de rails. Si l'on suppose que chaque rail possède son propre système de déplacement de palettes (chaîne, convoyeur débrayable), alors tous les objets du convoyeur sont indépendants. Cependant dans la majorité des cas, les chaînes de convoyage couvrent un nombre important de portions de rails. Dans notre cas par exemple, la même chaîne déplace l'ensemble des palettes des rails Rail8, Rail1, Aiguillage1, Rail9, Aiguillage4 et Rail7.

Le moteur associé à la chaîne est donc l'actionneur commun de tous les Objets Physiques Commandables.

Cet exemple illustre donc le fait qu'un même actionneur peut être associé à plusieurs objets monofonctionnels (dont la fonction est l'asservissement en

position dans ce cas).

Chaque chaîne de transport définit donc un ensemble d'objets qui évoluent de manière synchrone.

III.3.5.2. Contraintes entre les machines

Les robots Robot-1 et Robot-2 manipulent des pièces

- entre le Stockage et le Convoyeur,

- entre le Convoyeur et le Tour, la Taraudeuse, la Machine d'Assemblage.

Le robot graisseur opère directement sur les palettes.

A partir de ces renseignements, nous pouvons affirmer que l'ensemble des machines doit être géré de manière synchrone. En effet, le chargement d'un objet du Convoyeur sur un robot induit une contrainte entre le convoyeur et le robot : la palette doit être immobile lors du chargement du robot.

Il est préférable que l'ensemble de toutes ces machines soit commandé par le même automate programmable industriel (API).

Il est clair que cette solution n'est pas envisageable. Afin de lever cette contrainte de synchronicité il faut s'assurer que les ordres qui sont effectivement transmis au procédé satisfont les contraintes inter machine. De la sorte il n'est plus nécessaire de les contrôler en temps réel.

Les machines sont contrôlées séparément (par exemple un API par machine constitue de fait l'approche la plus courante et la plus naturelle).

La Partie Commande doit être validée afin de vérifier qu'à tout instant les contraintes entre les machines, qui apparaîtront du fait des flux de pièces, sont vérifiées par les commandes transmises.

Prenons l'exemple du chargement du tour par le Robot-1.

La PC synchronise temporairement le tour et le Robot-1 selon le schéma suivant :

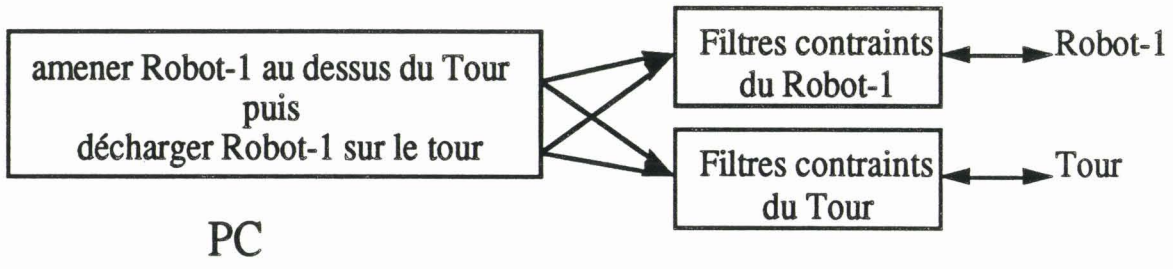


Figure 17

Nous devons donc valider formellement qu'au niveau de la PC, le tour est inactif à chaque positionnement du Robot-1 au dessus du tour. Dans ces conditions seulement on est autorisé à découpler totalement les filtres associés au Robot-1 d'une part et au tour d'autre part.

Nous avons donc été conduits à définir un nouveau critère de validation de l'ensemble de contrôle commande sur la base du problème de la répartition des processeurs de traitement.

III.4. Relations d'accessibilité

Dans un système de production, une grande part du traitement est associé au contrôle et du suivi des flux de pièces et de supports de pièces que sont des palettes, des caisses, des plateaux etc...

Nous verrons au chapitre IV, qu'il est indispensable de définir explicitement tous les trajets possibles, pour une pièce par exemple, au sein d'une cellule de production.

Il est donc nécessaire de disposer d'un modèle traduisant simplement les possibilités de transfert entre les différents lieux de stockage.

Nous appelons relation d'accessibilité entre deux lieux de stockage (fixes ou mobiles), la possibilité pour le premier lieu de transférer un objet quelconque vers le deuxième lieu.

Nous obtenons donc un schéma regroupant toutes les relations binaires entre tous les lieux de l'unité considérée.

Le schéma est réalisé à chaque niveau d'analyse (profondeur variable) de la description du procédé.

Au premier niveau de la description arborescente de notre exemple, nous obtenons le schéma des relations d'accessibilité suivant :

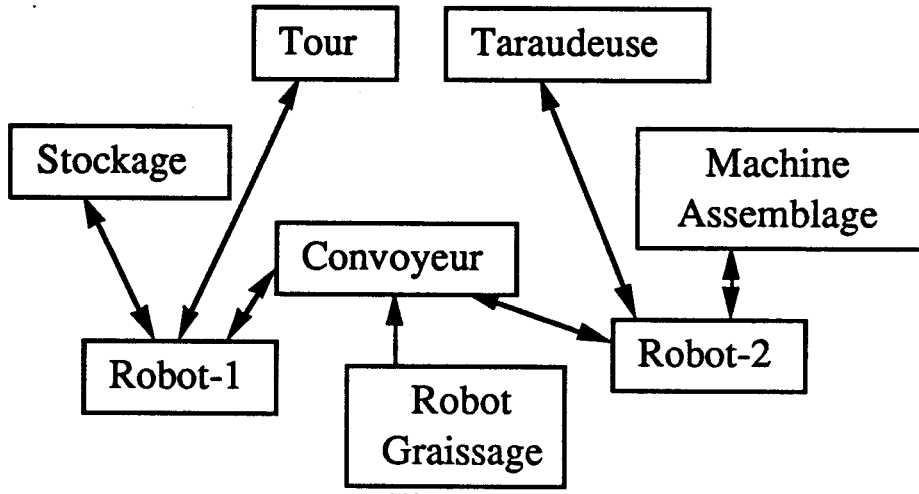


Figure 18

Ce schéma traduit les possibilités d'échange entre les robots de manipulation et les lieux de stockage et de traitement. Le robot graisseur peut accéder à la zone convoyeur pour l'opération de graissage.

A l'étape de niveau 2, les lieux sont affinés. Les relations d'accessibilité qui sont définies à ce niveau doivent bien sûr être compatibles (non contradictoires) avec celles de niveau 1.

Le cas particulier du Robot-2, qui possède deux préhenseurs, donne le schéma suivant :

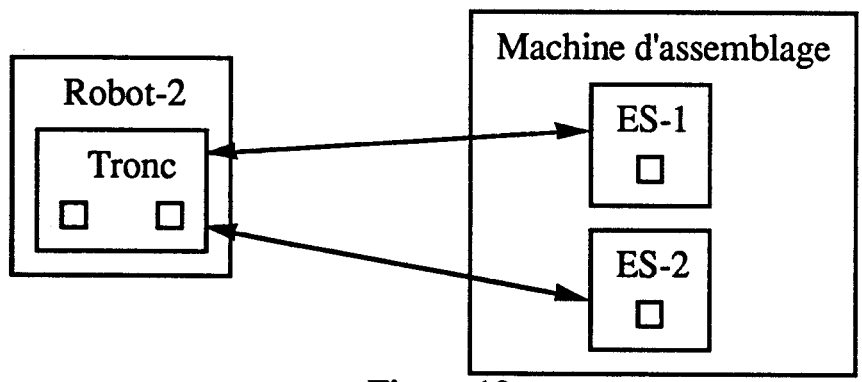


Figure 19

Les deux ES du Robot-2 peuvent échanger des pièces avec les deux ES de la machine d'assemblage. Dans ce cas, nous avons choisi de ne faire figurer qu'une seule flèche pointée en son centre. Elle schématise le fait que tous les ES d'un premier lieu peuvent échanger des objets avec tous les ES du second lieu.

Nous obtenons donc le schéma :

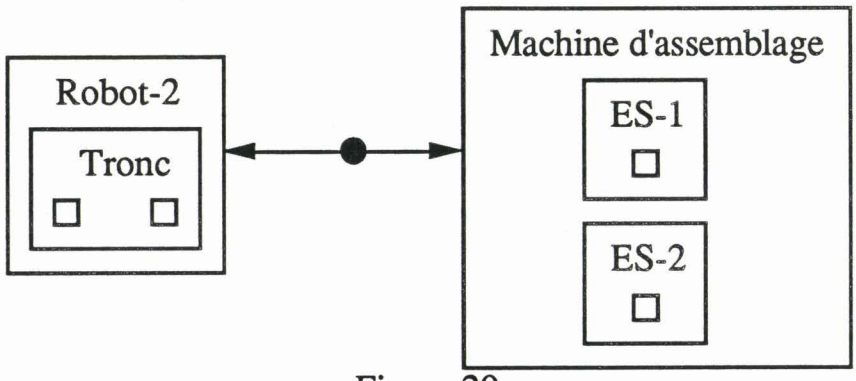


Figure 20

Il serait également possible de restreindre les relations d'accessibilité en fonction de la nature des objets transportés (des pièces de tel type, des plateaux, etc...) ou de la nature de lieux de stockage vis à vis des pièces transportées.

Les relations sont définies pour chaque niveau du procédé. Le schéma final (associé au modèle de profondeur huit du procédé) est le suivant :

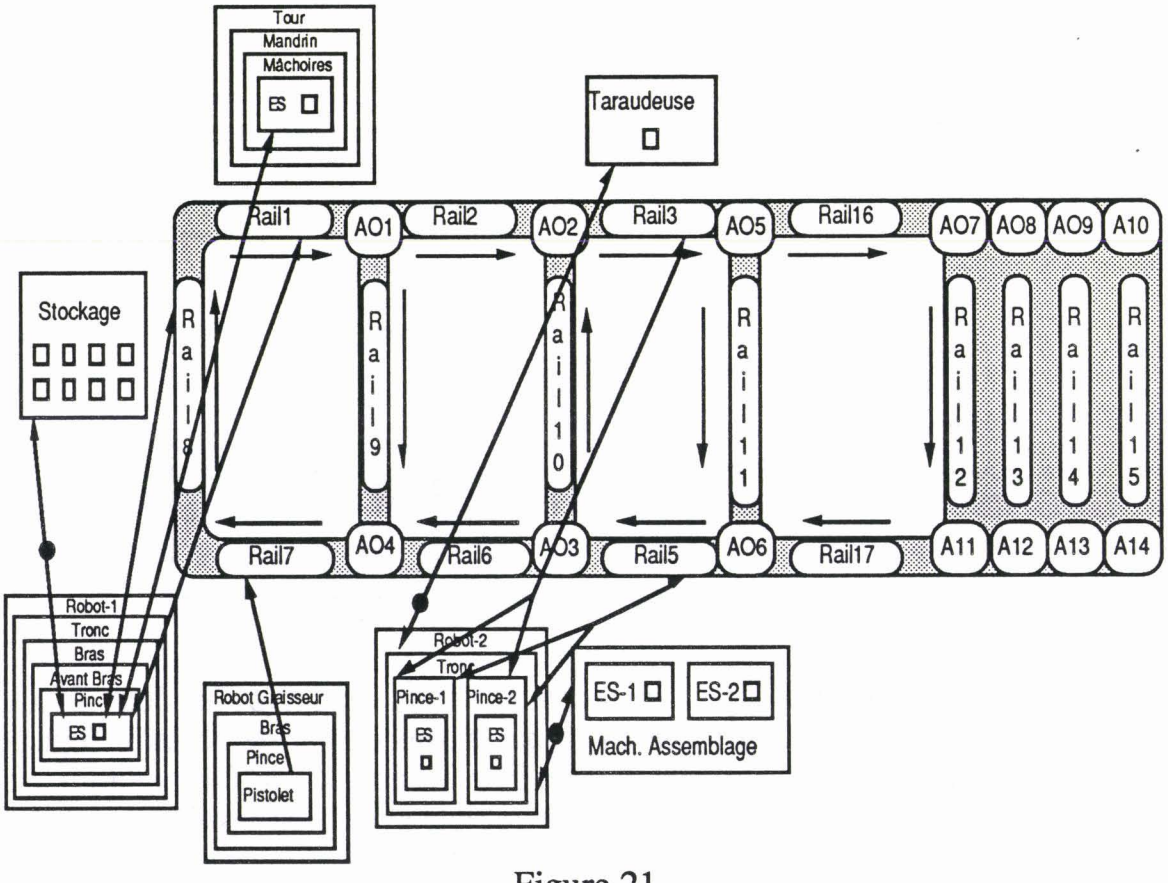


Figure 21

Il servira de modèle de base pour la génération des gammes opératoires de l'installation au chapitre IV.

Conclusion

Dans ce chapitre, nous nous sommes attachés à analyser le procédé sans contraindre à priori son utilisation en exploitation.

Cette démarche nous a permis de construire un ensemble de modèles du procédé satisfaisant un ensemble de liaisons ou dépendances (les filtres de commande), qui sont exploités aussi bien au niveau du contrôle temps réel sur API, qu'au niveau du module Interface.

De nombreuses études sont actuellement engagées par le L.A.I.I., notamment en ce qui concerne l'aspect générique des modèles, et une possibilité d'intégrer un système expert temps réel de surveillance.

Enfin, le modèle des relations d'accessibilité est exploité afin d'assister le concepteur dans la tâche de spécification des gammes opératoires des produits traités dans l'unité de production.

CHAPITRE IV

GENERATION ASSISTEE DES GAMMES OPERATOIRES

INTRODUCTION

Au cours du chapitre II nous avons soutenu l'idée d'une conception d'un système de production ne faisant aucune hypothèse restrictive sur les différentes entités à considérer, en fait il conviendra de réaliser une synthèse cohérente entre :

- d'une part le processus de fabrication qui explicite les différentes fonctions qui doivent être réalisées sur un exemplaire du produit fini et les contraintes d'ordre, et que nous appelerons gamme logique,

- et d'autre part le système de production, c'est à dire l'ensemble des moyens matériels capables de réaliser effectivement les fonctions de traitement des produits.

La synthèse entre la gamme logique (description fonctionnelle des opérations subies par le produit indépendante des moyens) et la description des moyens de production constitue la gamme opératoire.

La première partie de ce chapitre concerne la définition du modèle de description et d'analyse des gammes logiques.

Nous verrons en deuxième partie comment se réalise la prise en compte progressive et contrôlée de l'aspect opérationnel du système de production.

IV.1 - Génération des gammes logiques

IV.1.1. Introduction

La fonction essentielle d'une unité ou d'un système de production est de réaliser un produit dit fini à partir d'un produit initial et de plusieurs composants. Ainsi, créer un produit fini revient à agencer et traiter un ensemble de produits intermédiaires.

Il convient tout d'abord d'expliciter quelles sont la ou les procédures de fabrication possibles du produit. Décrire une gamme logique d'un produit revient à identifier quels sont les produits nécessaires, quels sont les traitements à appliquer sur les produits, dans quel ordre, et avec quelles contraintes d'antériorité.

Nous pouvons ainsi schématiser la fonction d'une gamme logique :

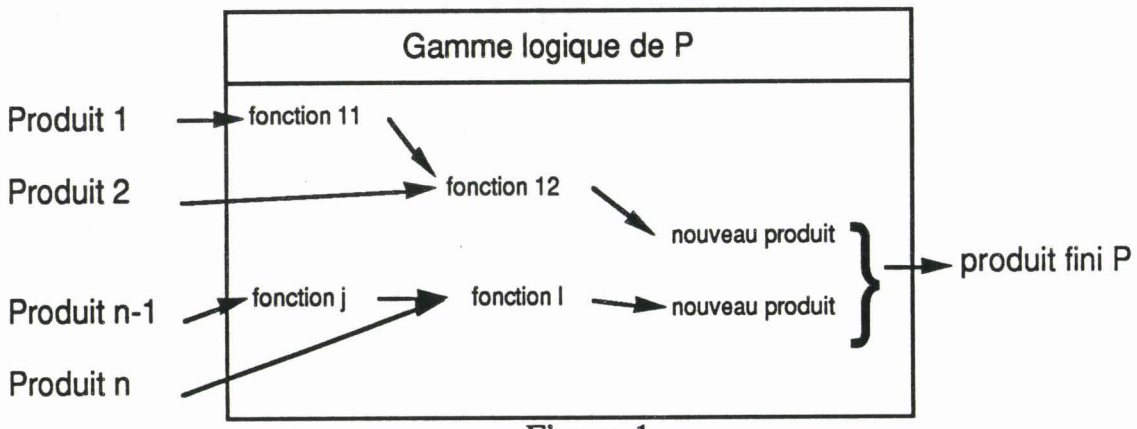


Figure 1

Dans la méthodologie CASPAIM, le résultat de la phase d'élaboration des gammes logiques sert d'ossature au graphe de commande qui sera effectivement implanté. C'est pourquoi son élaboration est menée et validée avec minutie.

De nombreux travaux ont traité le problème de l'élaboration des gammes d'usinage et d'assemblage (BOU 87b, BOU 90a, BOU 90b, BOU 90c, CAM 89, CHA 88, HEN 90).

Nous ne citerons que la méthode développée dans la thèse d'Alain BOURJAULT (BOU 84) sur une approche méthodologique de l'assemblage automatisé pour l'élaboration des séquences opératoires.

Notons que le problème de la construction des gammes logiques ne fait pas partie des axes de recherches du L.A.I.I. Le travail présenté par l'équipe d'Alain BOURJAULT du LAB est de ce fait complémentaire au nôtre.

Nous considérons que le processus de fabrication d'un produit composite à partir de produits dissociés est connu, éventuellement selon plusieurs variantes (flexibilité de la gamme logique). Notre propos n'est pas de valider cette gamme logique en vérifiant que le produit généré satisfait à son cahier des charges, mais d'exploiter cette gamme en vue d'une implantation de la commande du système de production de ce produit.

La méthode de conception des gammes logiques d'assemblage et d'usinage présentée par A. Bourjault est rapidement détaillée dans le paragraphe suivant, nous verrons ensuite sur un exemple d'illustration la manière dont nous appréhendons la description d'une gamme logique.

IV.1.1.1. La méthode présentée par Alain Bourjault (BOU 84)

Le principe de la méthode est de décomposer les assemblages en liaisons fonctionnelles Li BINAIRES entre deux composants de l'assemblage final.

Pour l'exemple du stylo à bille de type Bic, décrit par la Figure 2 :

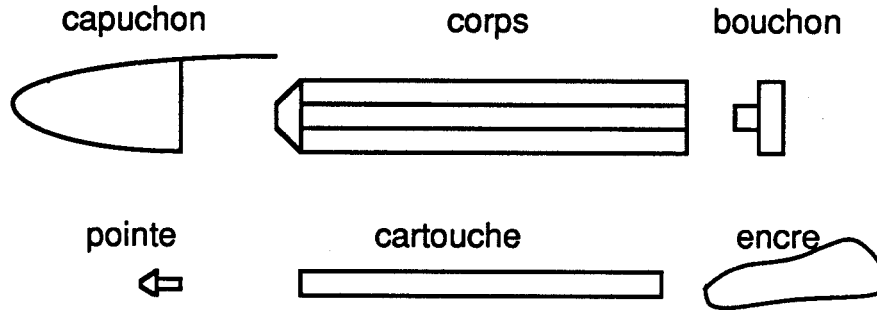


Figure 2

On obtient le graphe des liaisons fonctionnelles :

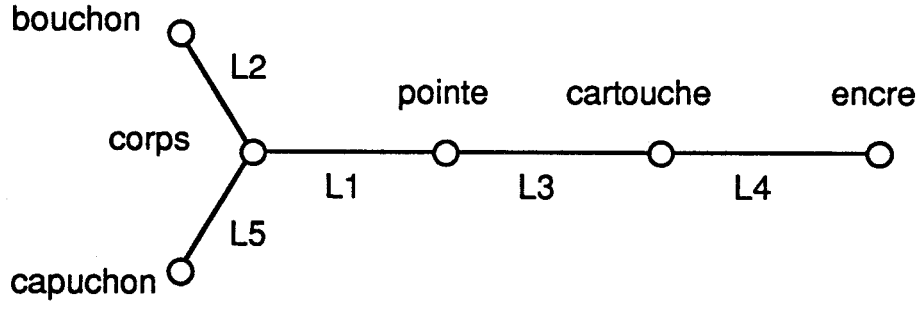


Figure 3

A chaque liaison Li on associe la condition de réalisabilité Ci construite à partie d'une expression booléenne traduisant l'existence ou l'absence d'autres liaisons Lk.

ex : Ci=L1.Not(L2) signifie que Li ne peut être réalisée que si L1 est réalisée et L2 ne l'est pas encore.

La méthode de conception a été développée et permet, grâce à l'utilisation d'expressions booléennes intermédiaires, de déterminer les expressions minimales des Ci.

Dans le cas du stylo à bille nous obtenons :

C1=Not(1).(Not(2).3.Not(5)+Not(2).Not(4).Not(5)+3.4.Not(5))
C2=Not(2).(Not(1).3.Not(5)+3.4.Not(5)+Not(1).Not(4).Not(5)+1.3.4)
C3=Not(3).(1.Not(2).Not(4)+Not(2).Not(4).Not(5)+Not(1).Not(4).Not(5))
C4=Not(4).(1.Not(2).3+Not(1).3.Not(5)+Not(2).3.Not(5))
C5=Not(5).(1.3.4+1.Not(2).3+1.Not(2).Not(4))

où 1 signifie la présence de la liaison L1 et Not(1) son absence.

L'analyse de ces contraintes permet d'identifier tous les chemins qui permettent de conduire à l'assemblage final, comme par exemple L3 puis L4 puis L2 puis L1 puis L5.

L'étude a été étendue aux problèmes des actions d'assemblage simultanées, aux actions indifférentes et aux sous-assemblages.

La thèse de M.Kapusta (KAP 88) présente une adaptation de la démarche basée sur un ensemble de contraintes physiques de réalisation des assemblages. Une inférence de type système expert permet d'identifier les séquences d'assemblage admissibles et menant au produit final assemblé.

IV.1.1.2. Approche fonctionnelle de l'assemblage

La méthode exposée précédemment impose que l'on décrive l'assemblage final sous la forme d'un ensemble d'assemblages binaires. Le même exemple du stylo à bille est exposé dans la thèse d'état de Mr Bourrières (BOU 90a). Cependant, l'assemblage de la bille dans la tête a été ajouté, et l'assemblage de la cartouche et de l'encre a été supprimé. Deux gammes logiques (appelées processus d'assemblage) sont présentés. Elles sont également basées sur une description sous forme d'assemblages binaires :

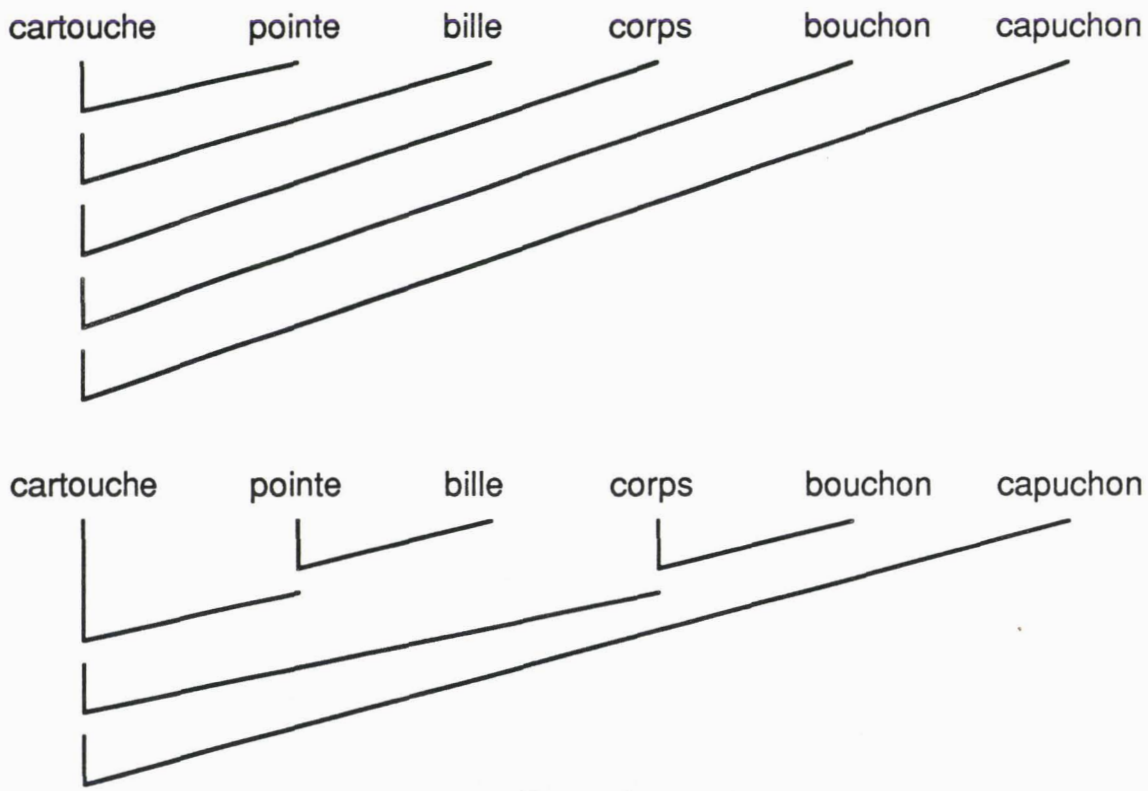


Figure 4

Considérons l'assemblage Cartouche, Pointe, Bille.

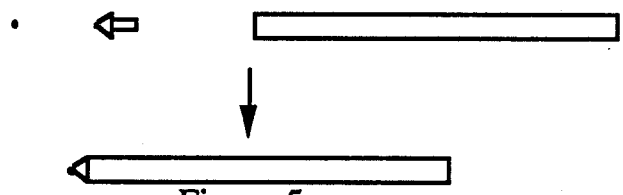


Figure 5

Il s'agit ici de réaliser deux assemblages ayant une pièce en commun :
bille + pointe et pointe + cartouche

Si les assemblages ne peuvent être que binaires non simultanés, alors les deux séquences possibles sont :

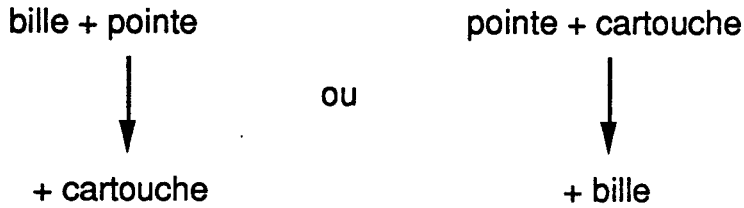


Figure 6

Or il est plus intéressant, du point de vue des performances du procédé de fabrication, de considérer que l'on réalise deux assemblages binaires indépendants donc non séquentialisés, bien que mettant en jeu un composant commun (la pointe) :

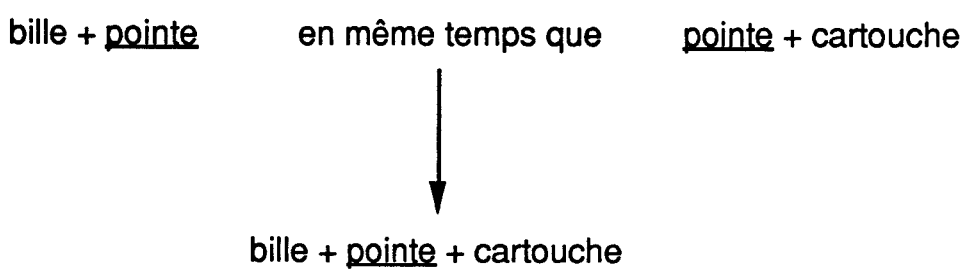


Figure 7

La durée de cet assemblage est alors la durée maximale des deux durées et non la somme comme dans le cas précédent.

Nous verrons dans la suite du chapitre qu'il s'agit ici de deux assemblages binaires indépendants sur une même pièce conduisant, par synchronisation par rendez-vous, à la création d'une nouvelle pièce.

L'assemblage ternaire bille+pointe+cartouche a été décomposé en deux

assemblages binaires indépendants.

La démarche de conception des gammes logiques est basée sur une spécification purement fonctionnelle du processus de fabrication. Elle consiste à identifier, pour un produit fini, quels sont les produits nécessaires, et quelles sont les fonctions à appliquer sur ces produits, et dans quel ordre. Aucune spécification opérationnelle n'est prise en compte dans cette conception.

Ainsi, pour l'exemple précédent, créer le stylo revient à réaliser une fonction d'assemblage 6-aire :

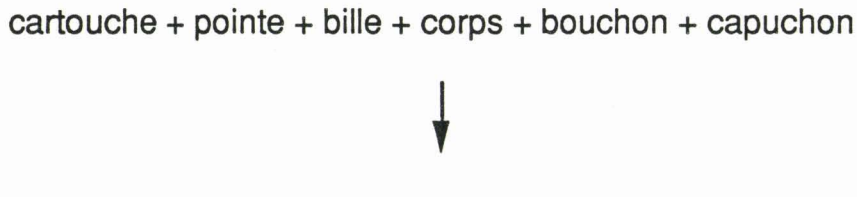


Figure 8

Or, d'après les contraintes Ci exposées précédemment, ou par simple bon sens, il est nécessaire de décomposer cette fonction en au moins deux fonctions d'assemblage consécutives car ne pouvant pas être simultanées. En effet, la fonction d'assemblage cartouche+pointe+bille+corps+bouchon doit être réalisée avant que la liaison capuchon+corps ne soit commencée :

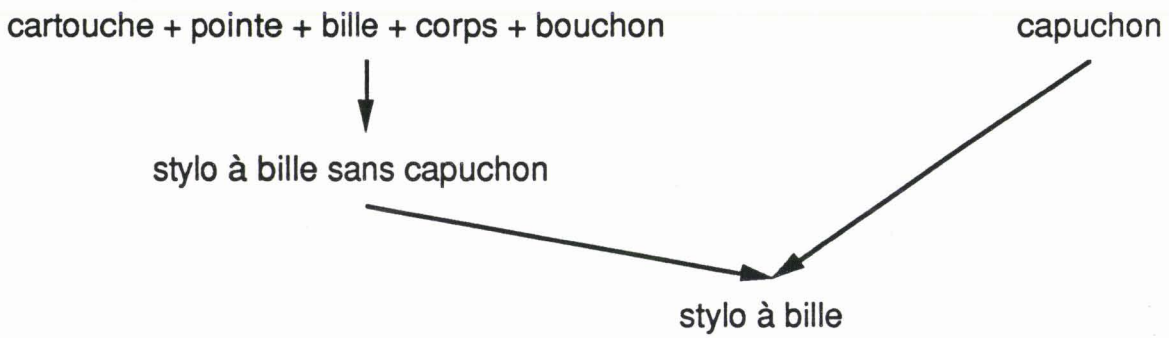


Figure 9

Le modèle que nous utilisons pour modéliser une gamme logique est le réseau de Petri (RdP) (BRA 83). Le RdP ordinaire suffit pour modéliser l'exemple précédent, pour lequel une place modélise l'état d'un produit et une transition une fonction qui fait disparaître les produits amont et crée un nouveau produit aval. Cette introduction informelle aux gammes logiques ne traite que des problèmes d'assemblage. Nous verrons dans la section suivante une définition formelle d'une gamme logique et du modèle sous-jacent. Le modèle RdP de la gamme logique précédente est donc :

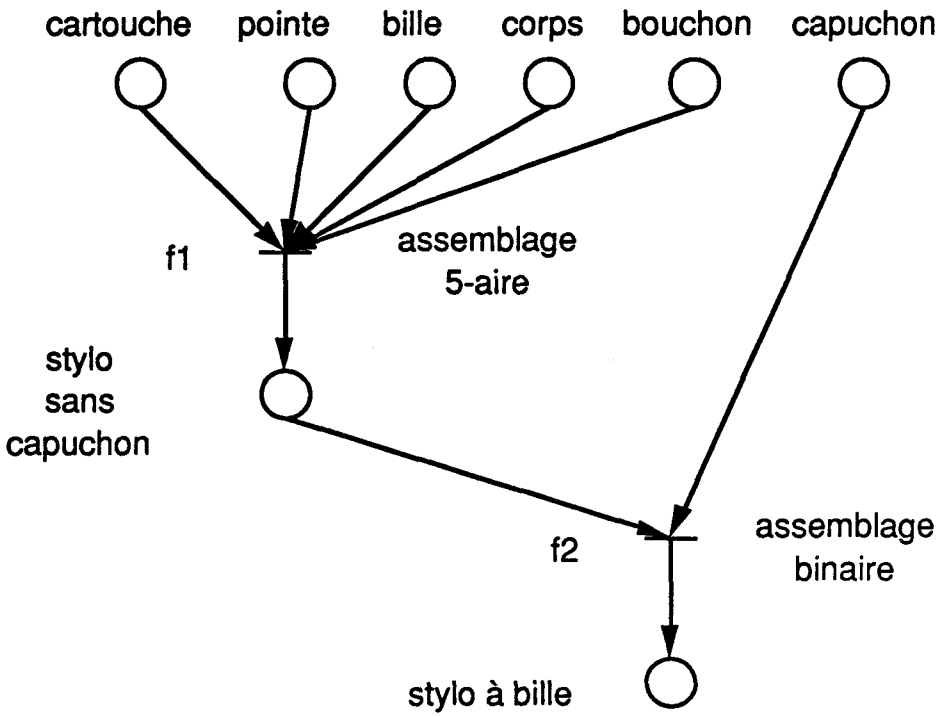


Figure 10

Le composant intermédiaire "stylo sans capuchon" est considéré comme n'importe quel constituant, et est assemblé avec le capuchon pour former le produit final.

Ce RdP traduit, de par sa structure, la précédence de f1 sur f2, pour un exemplaire de stylo donné.

Le macro-assemblage f1 doit également être décomposé. En effet l'assemblage bille + pointe + cartouche doit être réalisé avant de réaliser le positionnement de la pointe sur le corps, pour des raisons d'accessibilité. Par contre l'assemblage ternaire cartouche + pointe + bille est terminal dans le sens où aucune contrainte d'accessibilité n'impose que l'un des deux assemblages binaires soit réalisé avant l'autre. Du strict point de vue fonctionnel, la fonction d'assemblage ternaire f11 des trois pièces cartouche + pointe + bille n'est pas décomposable :

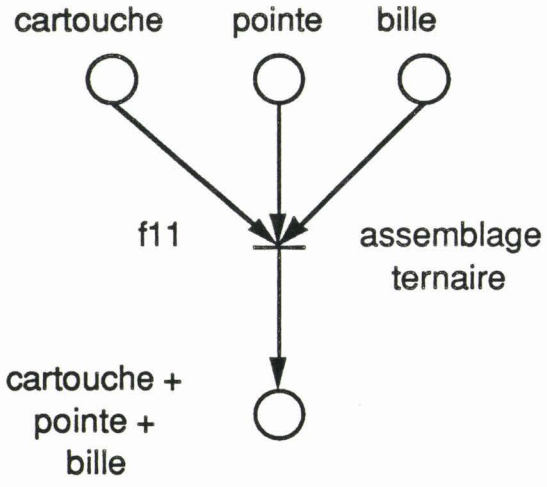
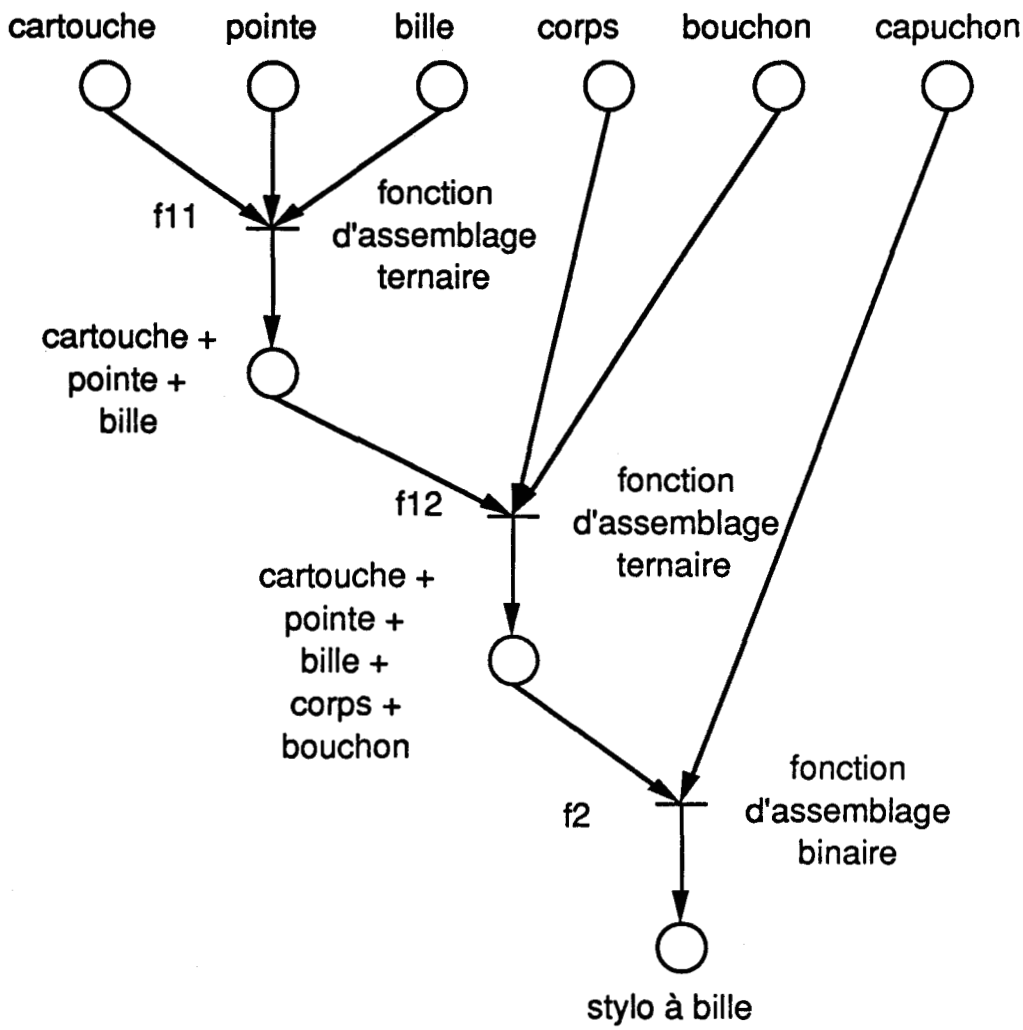


Figure 11

Dans le même esprit, l'assemblage de l'ensemble cartouche + pointe + bille avec le corps est libre par rapport à l'assemblage du corps et du bouchon. Ils peuvent être réalisés en même temps ou dans n'importe quel ordre. Il s'agit encore d'un assemblage ternaire non décomposable du strict point de vue fonctionnel. Le RdP traduisant le séquencement et les contraintes d'antériorité des fonctions d'assemblage est donc le suivant :



Gamme logique du stylo à bille
Figure 12

Une fonction d'assemblage n-aire est considérée comme terminale dans la phase de conception de la gamme logique si elle décomposable en fonctions d'assemblage d'arité inférieure, et si ces fonctions ne sont contraintes par AUCUNE relation de précédence. Nous conservons ainsi le degré de parallélisme de réalisation maximal (assemblages binaires simultanés et non contraints entre eux).

Le cas de l'assemblage d'un nombre important de composants électroniques sur un circuit imprimé est une bonne illustration de cette approche fonctionnelle basée sur le produit. Un circuit imprimé est composé de pistes et de trous dans lesquels se logent les broches des circuits intégrés et des composants.

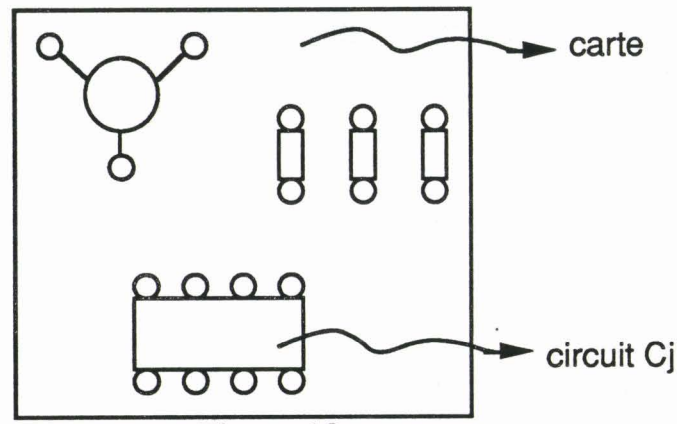


Figure 13

De fait, il est tout à fait possible de positionner simultanément tous les composants sur la carte puisque tous les emplacements sont accessibles. Les assemblages binaires carte+Ci sont donc totalement indépendants et ne sont contraints par aucune relation de précédence. La gamme logique est donc la suivante :

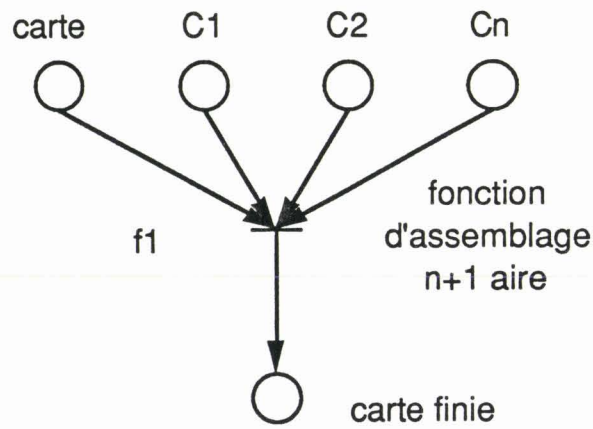


Figure 14

IV.1.1.3. Définition formelle d'une gamme logique

Cette approche fonctionnelle repose donc sur les points suivants :

- hypothèse du parallélisme maximal (assemblage n-aire)

- le parallélisme est restreint par la prise en compte de contraintes spécifiques au produit fabriqué (accessibilité des composants, encombrement...) d'où une séquentialisation des fonctions d'assemblage n-aires au moyen de fonctions d'ordre plus faible,

- une fonction d'assemblage est terminale et non décomposable si elle ne contient pas de fonctions d'assemblage en relation de précédence.

Une gamme logique décrit également, pour un produit donné, la séquence des fonctions unaires qu'il subit (usinage, fraisage, mesure...). Pour cela, on associe une place RdP par état d'avancement atteint et une transition par fonction.

Définition :

La gamme logique d'un produit fini P décrit la séquence et la synchronisation des fonctions qui sont appliquées à chaque produit qui entre dans la composition du produit fini P. Le modèle utilisé est le RdP où une place modélise l'état d'avancement d'un produit, et une transition modélise une fonction qui modifie l'état d'avancement d'un produit, ou crée de nouveaux produits à partir de produits existants.

L'outil de modélisation d'une gamme logique est le RdP car il permet d'exprimer facilement les relations de précédence et le parallélisme d'exécution des fonctions appliquées aux produits. Nous verrons dans la suite du chapitre que des extensions des RdP ordinaires sont nécessaires dans certains cas. Ainsi, les RdP colorés, dont la définition est présentée dans (BOU 88a, BRA 83), sont nécessaires lors de l'agrégation de gammes logiques de produits distincts mais ayant la même ossature.

Nous verrons aussi que les RdP à Objets (RdPO) que nous allons définir au paragraphe suivant, sont indispensables dès lors que l'on traite de pièces liées logiquement entre elles. Ainsi, l'assemblage d'une mémoire ROM personnalisée sur une carte unité centrale (UC) impose l'utilisation des RdPO dans la mesure où il faut individualiser la ROM et la carte pour réaliser l'assemblage. Bien d'autres exemples nécessitent l'utilisation des RdPO.

IV.1.2. Les outils de modélisation

L'outil RdP permet d'exprimer le parallélisme et la concurrence. Cependant dans les RdP ordinaires, les jetons sont tous identiques et indifférenciés. La flexibilité des gammes ainsi modélisées peut conduire à des réseaux très importants et très complexes. C'est pourquoi des extensions telles que les RdP colorés, les RdP à prédicats et enfin les RdP à Objets ont été peu à peu introduits.

Considérons l'exemple suivant :
On dispose de deux postes d'usinage M1 et M2 qui ne peuvent traiter qu'une seule pièce à la fois, et de trois types de pièces :

le type1 qui doit subir l'usinage1 sur M1 puis l'usinage2 sur M2,
le type2 qui doit subir l'usinage2 sur M2 puis l'usinage1 sur M1,
le type3 qui doit subir l'usinage1 sur M1 puis l'usinage1 sur M1.

IV.1.2.1. Modélisation par RdP ordinaire

La modélisation de cet atelier consiste à décrire les interactions entre les opérations que peuvent réaliser les machines et les opérations qui doivent être réalisées sur les pièces. Le RdP ordinaire associé à cet atelier est le suivant :

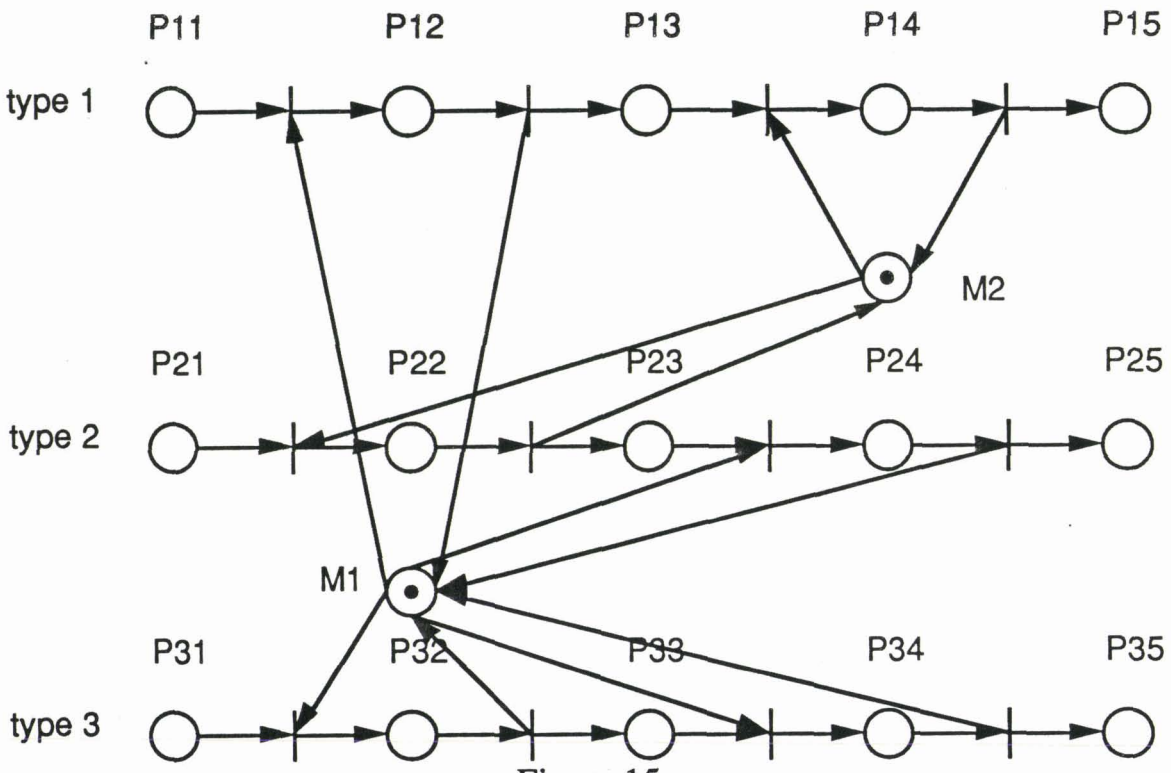


Figure 15

Ce RdP traduit effectivement la séquence des opérations sur chaque type de pièce et la concurrence d'accès aux machines qui sont considérées comme des ressources partagées. La place P12 correspond à la machine M1 en train de réaliser l'usinage1 sur une pièce de type 1. La place P13 correspond à une pièce de type 1 en attente de la machine M2 pour réaliser l'usinage2. A partir du marquage, il est possible de connaître l'état d'occupation des machines ainsi que les opérations en cours sur chaque type de pièce. Cependant, l'ajout d'une gamme accroît considérablement la taille du réseau.

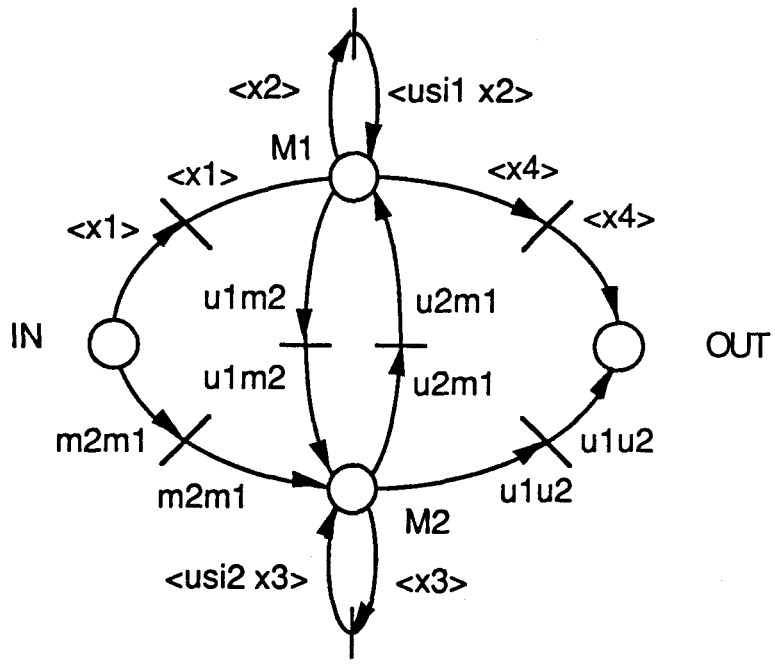
IV.1.2.2. Modélisation par RdP coloré

Dans le modèle réseau de Petri coloré (BRA 83, GEN 87, BOU 88b), une couleur est associée à chaque type de pièce. Nous avons en effet choisi au L.A.I.I. de ne pas agréger les machines. Cette couleur représente une information. Ainsi, à chaque place est associé un ensemble de couleurs. Plusieurs jetons de même couleur sont donc indiscernables et transportent la même information.

Pour notre exemple :

les pièces de type1 ont la couleur m1m2 puis u1m2 puis u1u2,
les pièces de type2 ont la couleur m2m1 puis u2m1 puis u2u1,
les pièces de type3 ont la couleur m1m1 puis u1m1 puis u1u1.

La couleur traduit donc explicitement l'état d'avancement d'un type de pièce donné. Le RdP coloré issu du précédent RdP ordinaire est le suivant (BOU 87a, CAS 87) :



dom(x1) = {m1m2, m2m1}
dom(x2) = {m1m2, m1m1, u2m1, u1m1}
dom(x3) = {m2m1, u1m2}
dom(x4) = {u2u1, u1u1}

USI1(x2) :
x2 → usi1(x2)
m1m2 → u1m2
m1m1 → u1m1
u2m1 → u2u1
u1m1 → u1u1

USI2(x3) :
x3 → usi2(x3)
m2m1 → u2m1
u1m2 → u1u2

Figure 16

Ce modèle permet d'effectuer des regroupements fonctionnels de telle sorte que les commandes d'usinages divers effectués sur la même machine soient regroupées sous le même système de contrôle.

L'utilisation de marques colorées permet une agrégation de graphes similaires en un seul modèle, plus concis.

Considérons maintenant un exemple illustrant les limites du modèle RdP coloré. Nous supposons que les transferts entre les machines sont réalisés par des palettes spécifiques à chaque type de pièce :

pal1 pour les pièces type1
pal2 pour les pièces type2
pal3 pour les pièces type3

Le transfert IN→M1 est donc modélisé par le RdP coloré suivant :

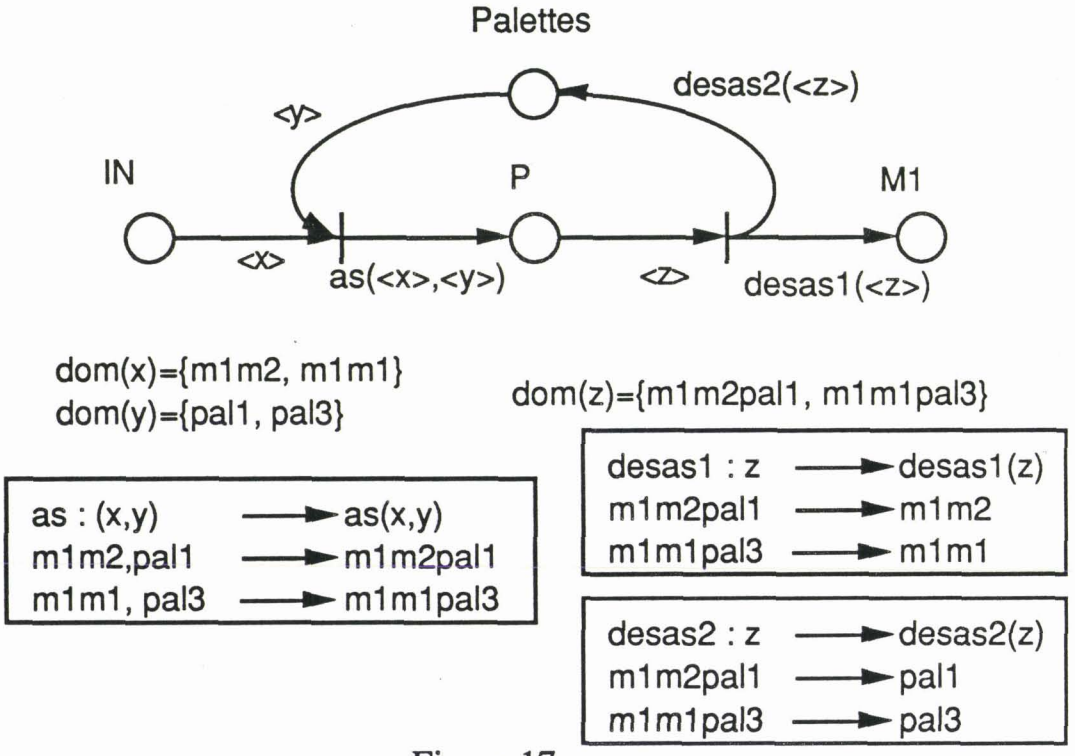


Figure 17

Cet exemple illustre le fait que lors d'une palettisation ou d'un assemblage, nous sommes conduit à créer autant de couleurs que de combinaisons admissibles des variables en entrée de la transition. Ces couleurs traduisent l'association momentanée de pièces de types différents. Il est clair que cela conduit rapidement à une explosion combinatoire du nombre de couleurs à traiter. Les réseaux de Petri à Prédicats-Transitions permettent de palier ce défaut.

IV.1.2.3. Modélisation par réseau de Petri à Prédicat - Transition (BRA 83)

Dans ce modèle, le jeton est un n-uplet de constantes. Des prédicats relatifs à une ou plusieurs composantes du n-uplet sont associés à chacun des arcs. Les marques ne peuvent transiter que suivant les arcs dont elles vérifient les

prédicats. Les prédicats permettent une définition de l'ensemble des marques autorisées à emprunter un arc en compréhension et non plus en extension.

Pour notre exemple, nous obtenons :

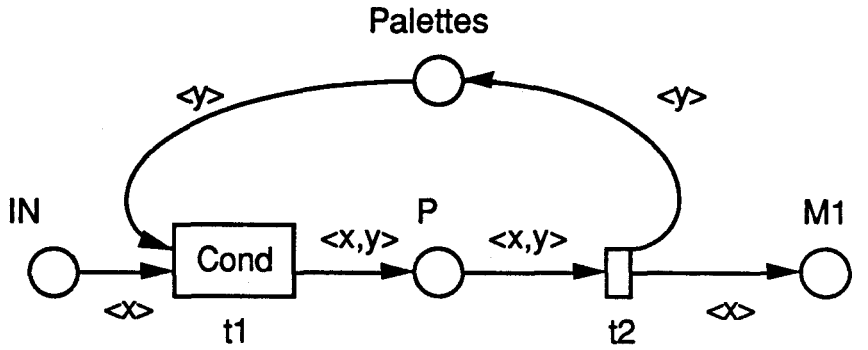


Figure 18

La condition de tir de t1 est :
cond = (x=m1m2).(y=pal1) ou (x=m1m1).(y=pal3)

Le jeton qui est généré lors du tir de la transition t1 est le couple <x,y> où x représente la pièce et y la palette. En conséquence, ces jetons gardent non seulement l'identité des individus qu'ils contiennent mais aussi les relations dynamiques entre ces individus dans une situation donnée.

La transition t2 génère le jeton représenté par la variable x du couple dans la place M1 et la palette y dans la place Palettes.

Dans les RdP à Prédicats-Transitions, les individus prennent des valeurs dans des ensembles de constantes (m1m2, pal1 ...).

IV.1.2.4. Modélisation par Réseau de Petri à Objets (SIB 85, SIB 88, SIB 91)

Dans le modèle RdP à Objets, les marques sont des objets instances de classes d'objets dont la valeur peut changer.

Dans un RdPO, les places peuvent contenir des marques qui peuvent être soit des constantes, soit des objets. Les marques sont donc identifiables, et elles ont un type donné qui détermine leur domaine de valeur. La valeur d'un objet peut être modifiée. Indiquons ici que la valeur d'un objet est en fait caractéristique de la valeur de l'ensemble de ses attributs.

Le franchissement d'une transition consiste à retirer des constantes et des objets des places d'entrée de la transition, en les ayant sélectionnés en fonction de leur valeur, puis à exécuter une action qui calcule de nouvelles constantes,

crée de nouveaux objets éventuellement, et modifie les valeurs d'attribut des objets pris en entrée. Ces constantes et objets sont déposés dans les places de sortie de la transition.

En fait une place contient des objets qui peuvent être liés dynamiquement les uns aux autres. Ainsi ce sont des n-uplets d'objets qui sont déposés et enlevés. On dit que chaque place a une certaine arité.

IV.1.2.4.1. Définition des Objets (SIB 90)

Une classe d'objets est identifiée par son nom, et elle décrit la structure générale des objets qui sont ses instances.

La structure générale des objets comprend
- une structure de données caractéristique de l'état courant de l'objet,
- un ensemble d'opérations caractéristique du savoir faire de l'objet.

Chaque objet est identifié par son nom, et appartient à une classe.

La structure de données d'une classe est caractérisée par un ensemble d'attributs (ou de champs) qui ont chacun un nom, et un type. Si ce type est simple, l'attribut est appelé propriété, s'il s'agit d'une classe d'objets, l'attribut est appelé une référence.

Considérons à titre d'exemple la classe Un_Individu définie par la date et le lieu de naissance, et les parents de l'individu.

Les attributs sont donc de deux types :

Classe Un_Individu :

Attribut	Type ou classe	
ma_date_naissance	date	;---propriété
mon_lieu_naissance	chaîne	;---propriété
mon_père	Un_Individu	;---référence
ma_mère	Un_Individu	;---référence

Un type simple est :
un type prédéfini standard (entier etc...),
ou un type défini par intervalle ou par énumération.

Un attribut dont le type est une référence est caractérisé par :
sa valeur, ou nom d'un objet,
les opérateurs admis pour les noms d'objets : création, affectation, test d'égalité, appel d'une opération.

Un attribut est dit public si tous les objets peuvent le consulter. Il est dit privé dans le cas contraire. La seule relation d'ordre entre les classes est la relation d'héritage. La relation de composition par inclusion n'existe pas.

On appelle méthodes les opérations qu'un objet peut réaliser. Une opération est caractérisée par :

- son nom,

- ses paramètres d'entrée (type simple ou classe d'objet),

- ses paramètres de sortie (idem),

- son corps qui peut utiliser les variables locales, les paramètres de l'opération et les attributs de l'objet.

Une opération peut faire appel, dans son code, aux services d'un objet qui est référencé dans ses paramètres d'entrée ou aux services de l'objet dans laquelle elle figure.

Un objet met donc à la disposition de ses clients (objets demandeurs) la consultation de ses attributs publics, et l'appel de ses opérations publiques.

Exemple de définition d'une classe d'objets :

Classe **Fenêtre** :

Public
titre, hauteur, largeur, ouvert, contient, élargir, position

Attributs

titre	: chaîne	;---propriété
hauteur, largeur, échelle	: entier	;---propriété
ouvert	: booléen	;---propriété
contient	: fenêtre	;---référence

Opérations

élargir (hauteur, largeur : entier) ; élargit la fenêtre
do ... end;
position : entier, entier ; retourne la position de la fenêtre dans l'écran
do ... end;

IV.1.2.4.2. Définition des réseaux de Petri à Objets

Nous allons donner une définition formelle de l'outil RdPO, puis nous illustrerons les notions présentées sur un exemple de palettisation.

Pour l'ensemble E, on notera E* l'ensemble des n-uplets d'éléments de E, n entier, et N(E) l'ensemble des parties de E.

Un RdPO est défini par la donné des éléments suivants :

1. Un ensemble **Typ** de types qui sont soit des types simples soit des classes d'objets

2. Un ensemble de Registres définis par un nom, un type de l'ensemble **Typ** et éventuellement une valeur initiale

3. Un ensemble **V** de variables typées qui ne pourront être substituées que par une valeur (constante ou objet) de même type qu'elles

4. Un ensemble **P** de places. Chaque place a une certaine arité et un type pour chacune de ses composantes. Une place d'arité 0 est dite de type jeton (marquage élémentaire au sens des RdP ordinaires).

5. Un ensemble **T** de transitions

6. Une fonction d'incidence avant
Pre : $\mathbf{P}x\mathbf{T} \rightarrow N(\mathbf{V}^*)$
et une fonction d'incidence arrière :
Post : $\mathbf{P}x\mathbf{T} \rightarrow N(\mathbf{V}^*)$
qui respectent l'arité et le type des places.

Si une variable v figure dans Pre(t,p), on dit que v est une variable d'entrée et p une place d'entrée de t;
si une variable v figure dans Post(t,p), on dit que v est une variable de sortie et p une place de sortie de t.

7. A chaque transition peut être associée une Précondition qui permet de tester la valeur des n-uplets se trouvant dans ses places d'entrée. Une précondition est une expression booléenne dans laquelle peuvent figurer des constantes, les registres du réseau et les variables d'entrée de la transition. Les variables dont le type est une classe d'objet peuvent être suffixés par un attribut (f.largeur) ou une fonction publique (f.ouvrir) de cette classe d'objets.

La classe fenêtre possède entre autre l'attribut largeur. La méthode ouvrir réalise l'affichage effectif en fonction des valeurs des attributs de l'objet.

8. A chaque transition peut être associé un ensemble d'instructions ou

Action. Il y a deux sortes d'instructions :

- l'affectation, qui permet de modifier la valeur des variables d'entrée et de donner une valeur aux variables de sortie, notée **variable:=expression**, où :

variable est une variable de sortie de la transition ou un registre du réseau

expression est une expression de même type ne faisant intervenir que des variables d'entrée ou des registres, éventuellement suffixés par un attribut ou une fonction publique de cette classe d'objets

- ou l'appel à une opération procédurale d'un objet, notée **variable.opération**, où :

variable est une variable de la transition ou un registre dont le type une classe d'objets

opération est l'une des opérations publiques procédurales de cette classe d'objets, éventuellement suivie de ses paramètres d'appel.

Marquage

L'état d'un réseau est déduit de son marquage.

Pour un type t, appelons Dom(t) son domaine. Dans le cas d'une classe d'objets, son domaine est l'ensemble des noms de ses instances. L'Univers des marquages d'un réseau est donc

U = union des Dom(t) avec t dans l'ensemble **Typ**

Le marquage d'un réseau est défini par la donnée de :

1. une fonction de répartition
   $\mathbf{m} : \mathbf{P} \rightarrow N(\mathbf{U}^*)$
qui indique l'ensemble des n-uplets que contient chaque place

2. la valeur des objets figurant dans les places du réseau

Franchissement

Soient M un marquage et t une transition d'un RdPO. t est franchissable depuis M, s'il existe des valeurs disponibles dans les places d'entrée de la transition qui satisfont la Précondition de t.

La substitution $S : V \rightarrow U$ associe à chaque variable une valeur de même type qu'elle et définit pour quelles valeurs des n-uplets de variables (constantes ou objets) la transition est franchissable.

Lorsqu'une transition est franchissable depuis un marquage M, son franchissement produit un nouveau marquage M'. Le calcul de M' se déroule selon les étapes suivantes :

1. retirer des places d'entrée de t les n-uplets de valeurs unifiées par les variables,

2. exécuter les instructions de l'action de t qui font appel à la fonction créer pour générer de nouveaux objets,

3. exécuter en parallèle toutes les instructions de l'action de t,

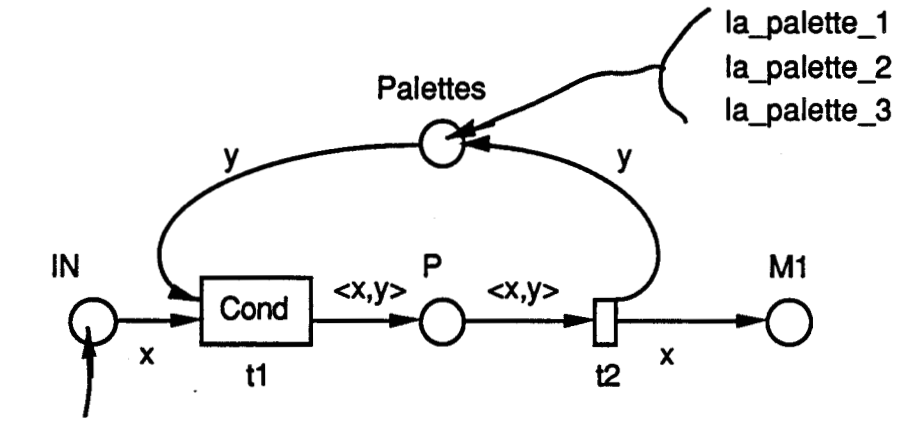
4. déposer dans les places de sortie de t les n-uplets de valeurs capturés par les variables de sortie.

A l'issue du franchissement d'une transition, le dépôt d'un objet dans une place de sortie est conditionné par la satisfaction de l'une des règles d'émission.

La disjonction des règles d'émission d'une transition doit être une tautologie afin que l'issue de son franchissement soit toujours définie. Par défaut, une transition a une seule règle d'émission constamment vraie.

IV.1.2.4.3. Exemple

Reprenons notre exemple de transfert par palette. La description par RdPO est la suivante :




```
t1 :
  condition x.nature appartient à l'ensemble y.natures_admissibles
  action ()
t2 :
  condition ()
  action begin
         y.nb_de_trajets := y.nb_de_trajets + 1;
         les_trajets := les_trajets + 1;
         end;
```

Figure 19

Le marquage du réseau est défini par :

- l'ensemble des n-uplets que contient chaque place, soit ici :

IN → la_pièce_1
Palettes → la_palette_1, la_palette_2, la_palette_3
P → ()
M1 → ()

- la valeur des objets figurant dans les places du réseau, soit ici :

la_pièce_1 : nature = m1m2

la_palette_1 : natures_admissibles = (m1m2)
nb_de_trajets = 15
la_palette_2 : natures_admissibles = (m2m1)
nb_de_trajets = 4
la_palette_3 : natures_admissibles = (m1m1)
nb_de_trajets = 150
les_trajets = 169

La condition associée à t1 permet de sélectionner en compréhension les

palettes et les pièces en consultant les attributs des variables x et y auxquels sont substitués les objets des places IN et Palettes.

L'action de la transition t2 permet de modifier les attributs des objets x et y, ainsi que la valeur du registre (les_trajets) qui est considéré comme une variable globale pour le réseau.

Il est possible d'ajouter à t2 une règle d'émission qui permet, par exemple, de faire une sélection sur la palette en fonction du nombre de trajets déjà effectués, et donc ici de renvoyer en maintenance les palettes usagées :

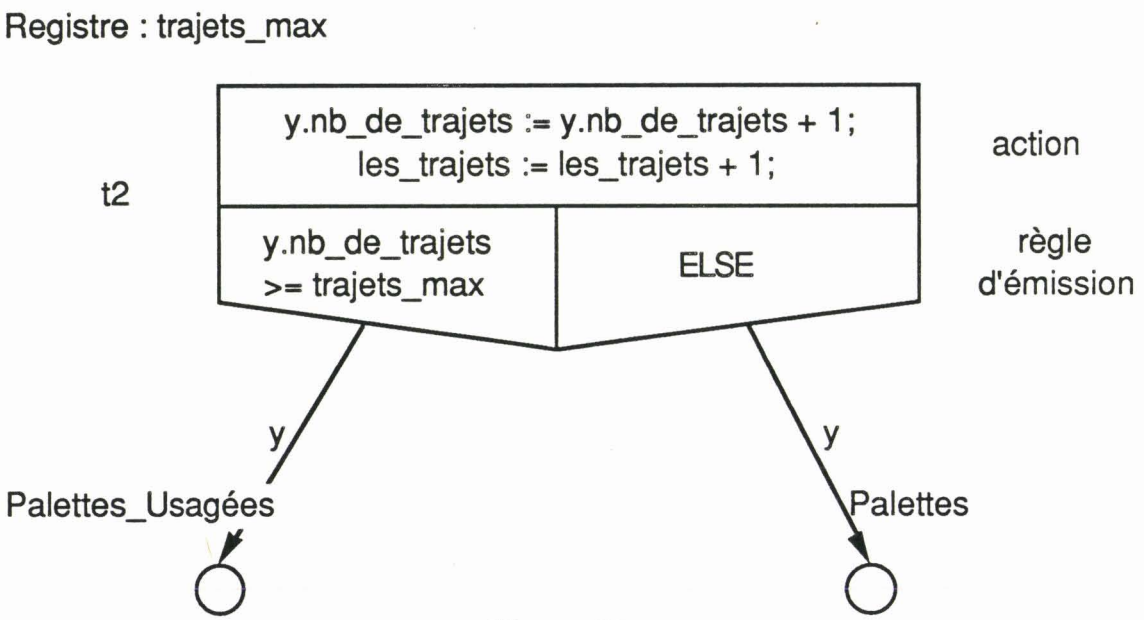


Figure 20

Notons enfin qu'un objet peut faire partie des n-uplets d'objets de plusieurs places mais qu'il ne peut exister qu'à un seul exemplaire dans chaque place, du fait que tous les objets sont discernables par leur identifiant, même s'ils appartiennent à une même classe.

Le marquage de M1 : m(M1) =(la_pièce_1, la_pièce_1) est donc illicite.

Nous avons vu, au travers d'un exemple simple la philosophie de la modélisation par RdPO. Le paragraphe suivant traite de l'interprétation qu'en fait Sibertin-Blanc. Nous verrons également la manière dont nous exploitons la puissance de cet outil dans le cadre de la productique, et en particulier dans l'élaboration des gammes logiques.

IV.1.2.4.4. Approche RdPO en vue de la modélisation du Réseau du Comportement (RC) d'un objet

Un objet, dans l'approche classique des langages orientés objet, est

caractérisé par sa structure de données (attributs) et par l'ensemble de ses opérations (méthodes). Dans de nombreux cas, les opérations d'un objet ne sont pas utilisables sans contraintes spécifiques, mais ne sont disponibles que lorsque l'objet se trouve dans un état donné.

Certaines opérations peuvent, par exemple, entrer en exclusion mutuelle. Ces contraintes définissent le mode d'emploi qui doit être respecté par les utilisateurs de l'objet afin de garantir que toute commande soit satisfaite.

Certains objets ont une activité déclenchée de manière spontanée ou reflexe qui n'est déclenchée par aucun message émanant d'un autre objet mais du fait de leur propre état interne (démons ou alerteurs).

Enfin les synchronisations entre objets qui sont induites par les messages font que les opérations d'un objet peuvent avoir une certaine durée.

Le comportement d'un objet, qui est déterminé par son mode d'emploi, son activité spontanée et les contraintes imposées par ses serveurs, est modélisé par un RdPO. Il est appelé Réseau du Comportement (RC) de l'objet (SIB 90).

L'objectif principal de ce réseau est donc de traduire explicitement sur un modèle les contraintes de réalisation mutuelles entre les méthodes d'un objet en fonction du contexte de cet objet et des objets environnants avec lesquels il communique pour réaliser ses opérations.

Cette approche est, en réalité, très proche de la fonction du module de l'INTERFACE que nous avons présenté au chapitre II. En effet la fonction de ce module est de satisfaire les commandes qui proviennent de la PC (services rendus par chaque Objet Physique Commandable cf chapitre III), tout en respectant les contraintes qui existent entre ces services. Nous pouvons assimiler un Objet Physique Commandable à un objet (au sens des langages orientés objet LOO) capable de rendre des services (commandes) compatible avec son état interne.

L'automate à états finis est bien sûr transposable de manière immédiate en un RdPO traduisant les conditions de réalisation des opérations de l'objet.

Prenons l'exemple d'un emplacement de stockage élémentaire. L'automate traduisant son état et les commandes qui le font évoluer est le suivant :

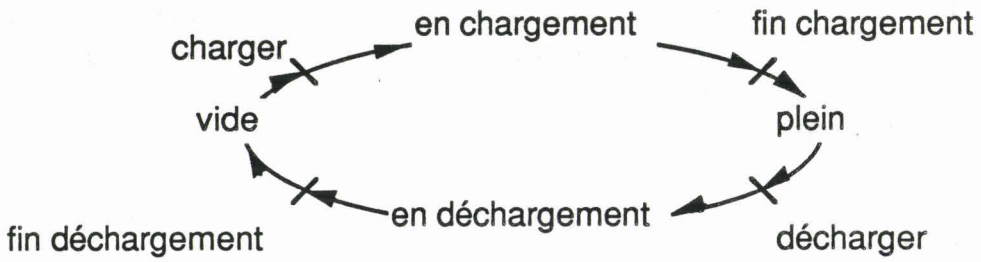


Figure 21

L'objet sous-jacent au sens LOO du terme est le suivant :

Classe EMPLACEMENT

Attribut :

mon-état : dans l'ensemble (en_chargement, plein, en_déchargement, vide)

mon_contenu : PIECE

Méthodes :

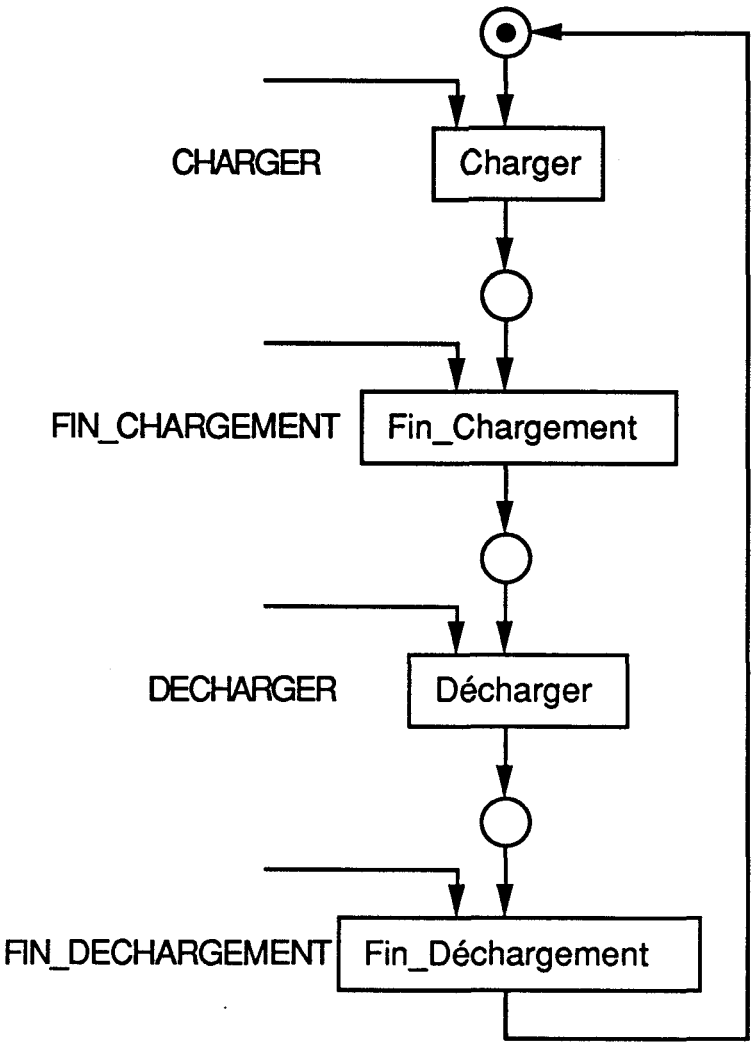
charger : mon_état := en_chargement

fin_chargement : mon_état := plein

décharger : mon_état := en_déchargement

fin_déchargement : mon_état := vide

Réseau du Comportement :



RC de l'emplacement de Stockage
Figure 22

Cependant, assez peu de contraintes peuvent être prises en compte explicitement par un RdPO. Aucune contrainte autre que le séquencement et l'exclusion mutuelle ne peut être gérée.

De plus, il est très délicat de traiter une contrainte qui met en jeu non seulement l'état du marquage du RC d'un objet mais aussi les méthodes (opérations) qui sont en cours d'évaluation chez des objets avec qui l'objet communique.

Le RdPO semble donc peu adapté, du point de vue de la productique, à la modélisation d'actionneurs réels mutuellement contraints tels qu'ils ont été présentés au chapitre II.

A l'heure actuelle, le modèle RdPO est surtout utilisé dans notre approche pour générer les gammes logiques et les gammes opératoires qui en découlent.

Les modèles RdP ordinaire et RdP coloré peuvent cependant suffire dans de nombreux cas.

Dans la section suivante (IV.1.3.), nous traitons des gammes logiques simples. Il s'agit des fonctions d'usinage, tournage, etc... et des fonctions d'assemblage. Dans la section IV.1.4. nous étudierons le traitement de gammes logiques emboîtées.

IV.1.3. Modélisation de gammes logiques simples

Nous nous intéressons tout d'abord aux gammes logiques faisant intervenir un seul composant, puis à celles faisant intervenir plusieurs composants. Cette section couvre l'intégralité des gammes logiques traitées dans l'ancienne version de la méthodologie CASPAIM (BOU 88a, KAP 88, CRA 89).

IV.1.3.1. Gammes logiques d'un composant isolé

D'après les hypothèses développées au chapitre II, nous supposons qu'un composant subit une seule opération à un instant donné. En conséquence il subit de façon disjonctive une fonction de transport qui ne modifie pas son état d'avancement, et une fonction de traitement qui modifie son état d'avancement. Les fonctions de transport ne sont abordées que dans la phase de génération des gammes opératoires (section IV.2.) puisqu'elles font intervenir explicitement les moyens opérationnels (manipulateurs).

La gamme logique d'un composant, par définition, ne décrit que la séquence et les alternatives éventuelles des fonctions de traitement qui doivent lui être appliquées. Enfin, une seule fonction est autorisée dans une situation donnée donc il est possible de modéliser l'état d'avancement du composant par un graphe d'état.

La modélisation de l'état d'un composant consiste à associer une place de RdP à chaque état atteint par le composant, et une transition par fonction.

On distingue deux types de fonctions unaires (KAP 88, BOU 88a):

1. les fonctions transformationnelles

Ces fonctions modifient l'état matériel du produit, c'est à dire ses caractéristiques intrinsèques, comme par exemple sa forme, ses propriétés physiques, sa constitution etc...

Ces fonctions sont par exemple :
- les usinages : tournage, fraisage, alésage,
- les conditionnements : traitement thermique, mise en peinture, lavage, graissage;

2. les fonctions informationnelles

Ces fonctions permettent de garantir la conformité des états matériels effectivement obtenus à partir des interventions extérieures.

Ces fonctions sont par exemple :
- des opérations de métrologie,
- des inspections qualitatives.

Prenons trois exemples caractéristiques afin de préciser le choix du modèle RdP à utiliser selon que l'on réalise une fonction transformationnelle et/ou une fonction informationnelle.

Ex 1. Considérons une pièce P qui doit subir successivement une opération de tournage, puis une opération de fraisage. Le RdP modélisant cette gamme logique est le suivant :

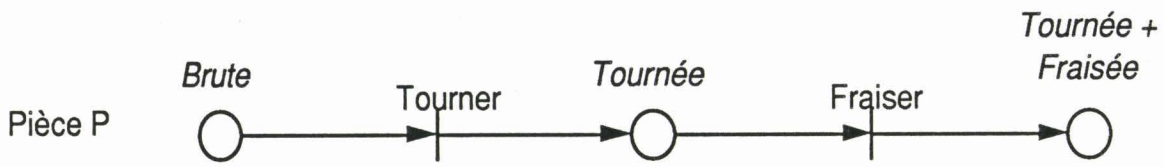


Figure 23

Les trois états successifs de la pièce sont : Brute, Tournée, Tournée et Fraisée.

Le RdP traduit :

- la précédence de la fonction de tournage sur la fonction de fraisage, pour un exemplaire donné,

- les trois états atteints par la pièce induits par le séquencement et la disjonction des deux opérations de traitement.

Ex 2. Considérons une pièce P qui doit subir une opération de tournage et une opération de fraisage, et ce dans n'importe quel ordre. Il existe alors deux gammes logiques distinctes :

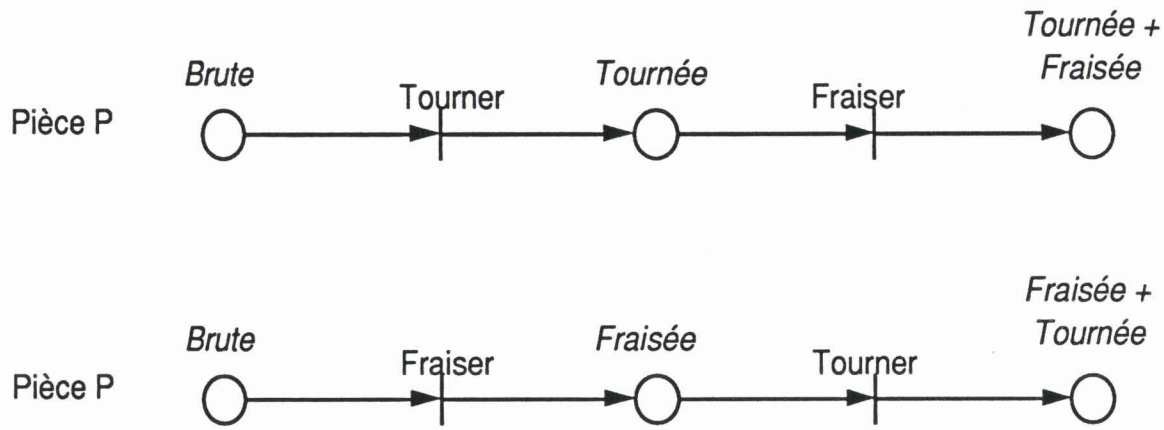


Figure 24

Dans la mesure où les deux états Tournée et Fraisée, et Fraisée et Tournée sont équivalents, le graphe d'état résultant des places correspondant à des états identiques est le suivant :

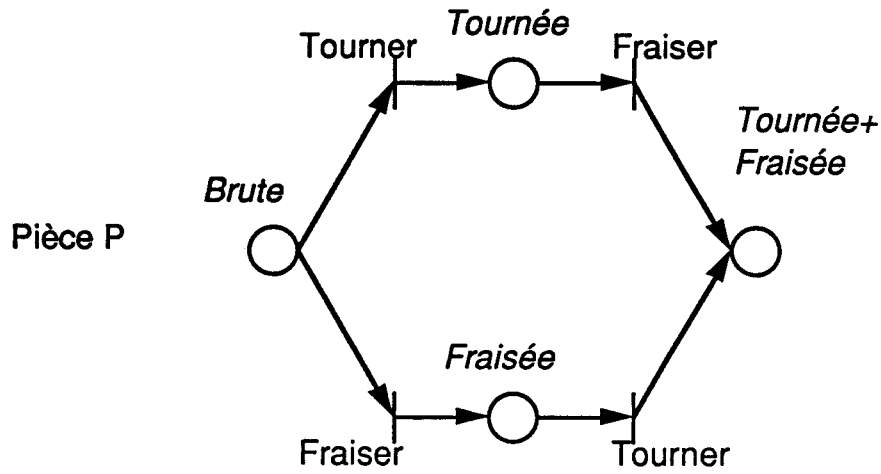


Figure 25

L'indéterminisme introduit par cette agrégation traduit la flexibilité de cette gamme logique. La résolution de cet indéterminisme doit bien sûr être effectué en phase d'exploitation. Ce problème sort des objectifs de notre mémoire.

Cependant la gamme logique d'une pièce doit traduire toute la combinatoire induite par la flexibilité, c'est à dire par les permutations d'opérations dont le séquencement est indifférent. Si n fonctions doivent être réalisées sur une pièce dans un ordre indifférent, n! gammes logiques peuvent être ainsi construites. Elles peuvent nécessairement être agrégées (voir Fig 24 et 25) en une seule puisqu'elles ont le même état initial et le même état final.

Prenons l'exemple d'une pièce devant subir trois opérations op_i dans un ordre indifférent. La gamme logique agrégée est la suivante :

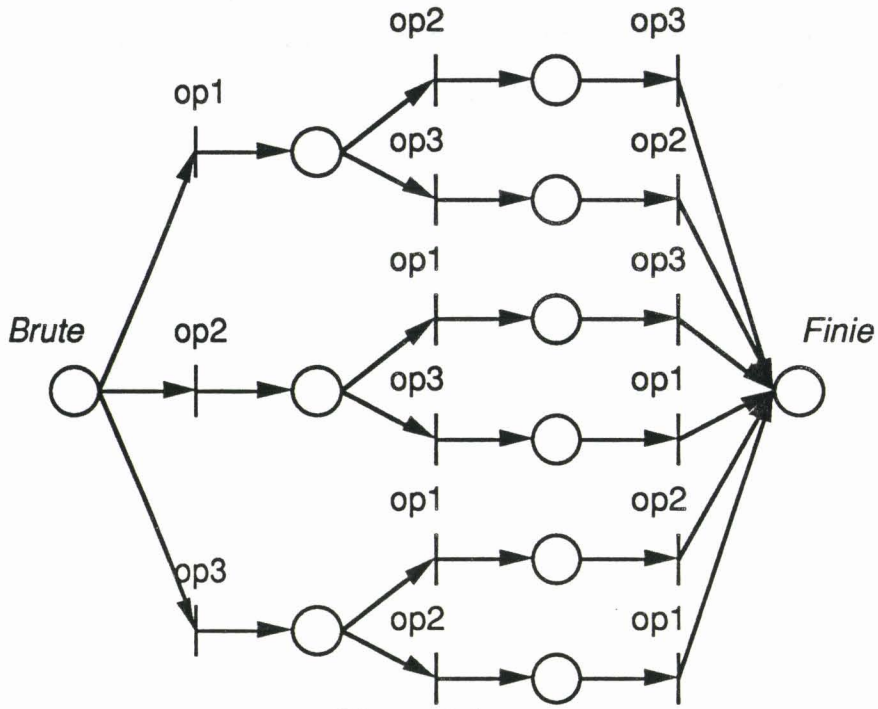


Figure 26

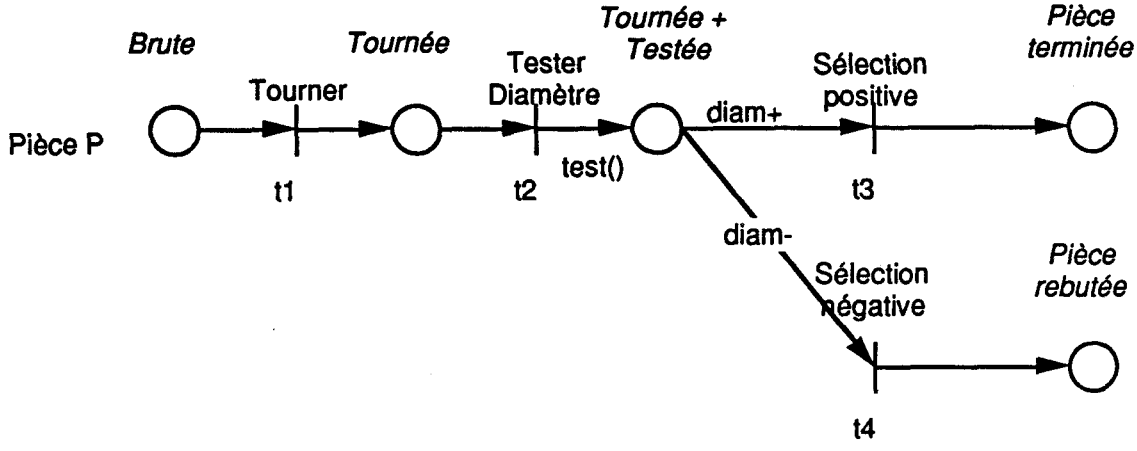
L'étude de nombreux cas nous a permis de constater que le nombre d'opérations permutables est rarement supérieur à 3. Cette explosion combinatoire inévitable d'états et de transitions n'apparait donc que très rarement dans le contexte des processus de production manufacturiers.

La modélisation par RdP ordinaire est donc suffisante pour une gamme logique qui ne réalise que des fonctions transformationnelles.

Ex 3. Considérons une pièce P qui subit une fonction transformationnelle puis une fonction informationnelle.

La fonction informationnelle Tester_Diamètre est supposée déterministe dans le sens où le test permet de classer la pièce dans deux intervalles distincts diam+ et diam-.

Comme sa nature l'indique, la fonction de test du diamètre restitue une information, unique mais qui peut prendre plusieurs valeurs. La modélisation par RdP ordinaire est donc insuffisante pour prendre en compte les valeurs de cet attribut. La modélisation par RdP coloré est l'extension minimale des RdP pour ce type de fonction. L'information qui résulte du test Tester_Diamètre est traduite en une couleur (par exemple diam+ et diam- selon le résultat du test) qui est ensuite testée pour tirer les transitions t3 ou t4 du graphe de la Figure 27.



avec test() = diam+ ou diam-

Figure 27

Les domaines de couleur associés aux arcs amont de t3 et t4 forment une partition donc il n'y a pas indéterminisme de choix entre t3 ou t4 pour le jeton coloré de la place Tournée+Testée.

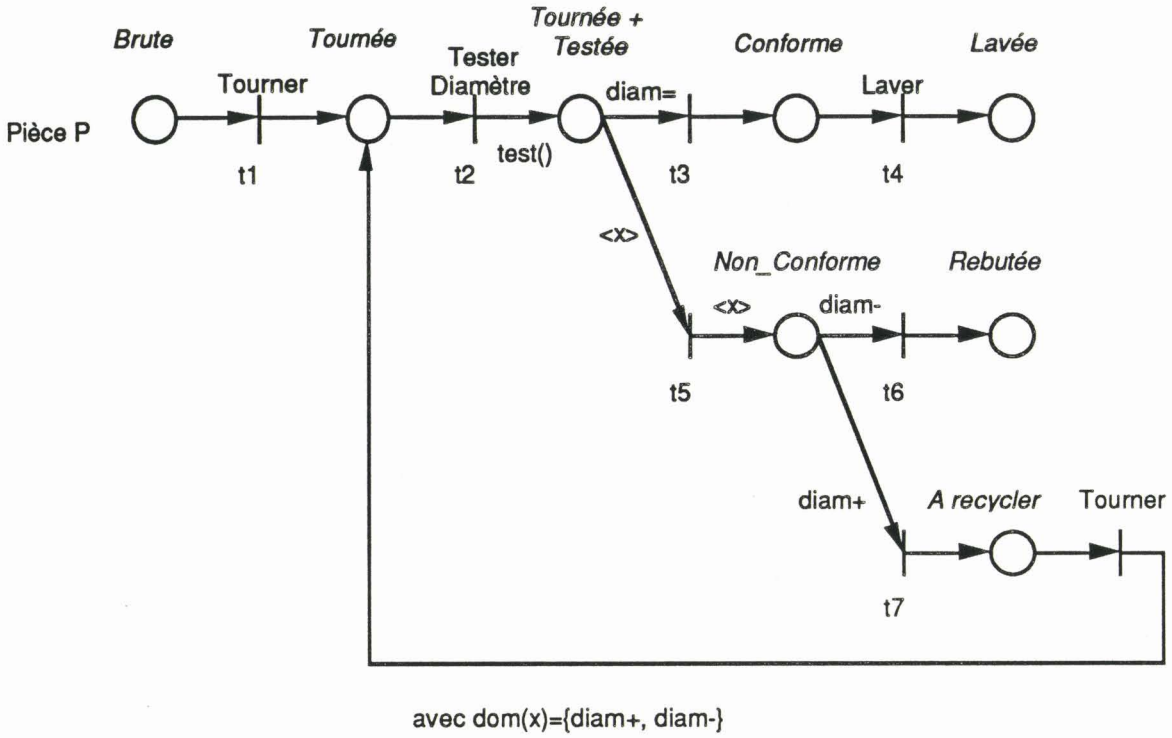
Complétons l'exemple précédent. Le résultat du test du diamètre peut prendre trois valeurs :

diam= : le diamètre correspond au diamètre voulu, avec une tolérance admissible,

diam+ : le diamètre est supérieur à la tolérance, la pièce peut être rectifiée,

diam- : le diamètre est trop faible, la pièce est rebutée.

La gamme logique est donc la suivante :



avec dom(x)={diam+, diam-}

Figure 28

Notons pour conclure que la gamme logique d'une pièce qui ne participe pas à un assemblage est nécessairement un **graphe d'état**. En effet, toute transition n'étiquette qu'un arc et un seul connectant deux places. Ce type de graphe dispose de propriétés particulières facilitant notamment les validations formelles.

Elle permet d'exprimer :

- la flexibilité introduite par les permutations de fonctions dont le séquencement est indifférent,

- les traitements sélectifs selon le résultat d'une fonction informationnelle,

- et enfin les rebouclages sur des fonctions transformationnelles mal ou incomplètement réalisées (traitement de défaut du point de vue des pièces produites).

Dans la section suivante nous traitons des interactions entre pièces lors d'assemblages ou de désassemblages.

IV.1.3.2. Gammes logiques de composants liés entre eux.

La majorité des cellules de production traitent des pièces qui doivent, au cours de leur fabrication, être assemblées avec d'autres pièces. Il est rare en

effet qu'une pièce ne subisse que des transformations unaires selon les hypothèses du paragraphe précédent.

Dans de tels cas, les gammes logiques des pièces qui entrent dans la composition d'un assemblage sont liées entre elles. En effet, la fonction d'assemblage se traduit par une nécessaire synchronisation prenant en compte les états des pièces concernées.

Par convention, nous avons choisi de représenter le RdP de la gamme logique d'une pièce horizontalement et de gauche à droite, et de séparer les gammes logiques des différentes pièces par un trait pointillé horizontal.

Enfin, dans la mesure où une opération d'assemblage concerne toutes les pièces dans leur ensemble, la transition qui lui est associée est placée sur l'un des traits de séparation des gammes logiques correspondant aux pièces constitutives de l'assemblage.

En effet, dans un assemblage, le réseau qui décrit l'évolution des pièces constitutives s'interrompt après exécution de la fonction d'assemblage. Une nouvelle pièce est alors considérée à ce niveau d'évolution du processus de fabrication.

Considérons l'exemple de l'assemblage d'une pièce P1 et d'une pièce P2 formant la pièce P1P2.

Ex 4.

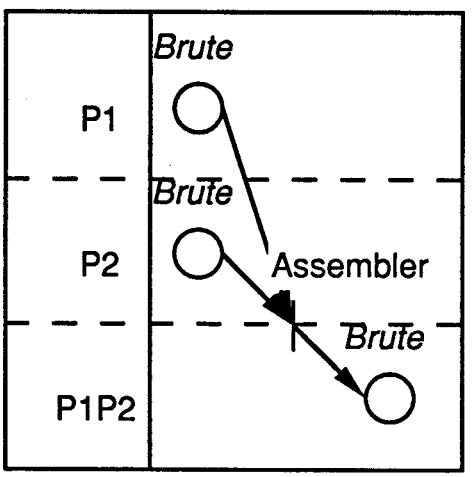


Figure 29

Ce réseau traduit bien la disparition des pièces P1 et P2 et la création de la pièce P1P2.

La pièce créée peut subir à son tour de multiples opérations transformationnelles, informationnelles ou d'assemblage.

Reprenons l'exemple du stylo à bille décrit au début de ce chapitre. Les gammes logiques des différentes pièces se synchronisent de la manière suivante :

Ex 5.

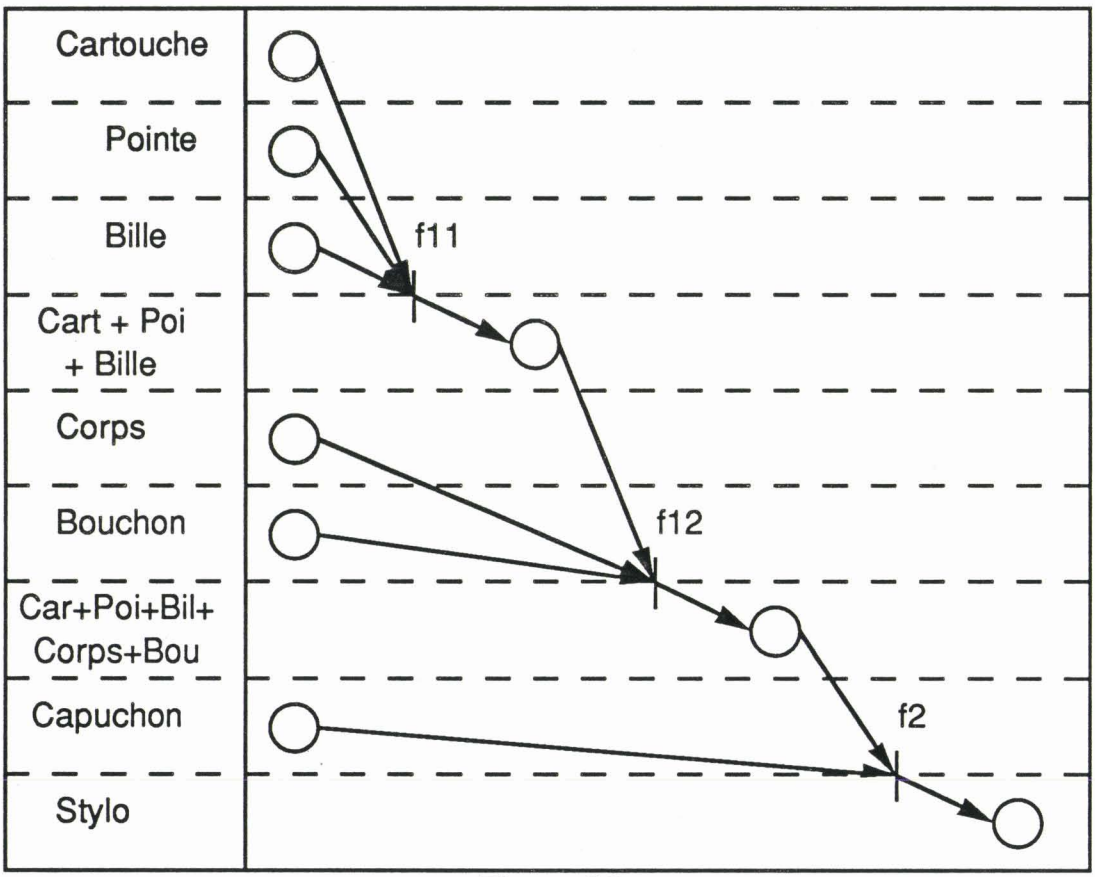


Figure 30

Le passage à des assemblages d'ordre quelconque ne présente aucune difficulté.

Une fonction de désassemblage se modélise de façon inverse:

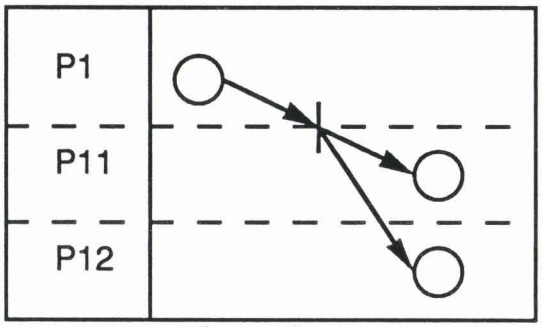


Figure 31

De la même manière que pour la gamme logique d'un composant isolé (cf

§ IV.1.3.1.), nous allons identifier la puissance de modélisation minimale de l'outil RdP en fonction du cas à traiter, sur la base de trois exemples concrets.

L'exemple du stylo de la Figure 30 est modélisé par un RdP ordinaire qui se trouve être un **graphe d'événements**. Nous verrons dans la section IV.3. que cette propriété est intéressante à exploiter pour la phase de validation.

Considérons la production de stylos à bille de deux couleurs : bleu et rouge. Les deux processus de fabrication sont bien sur identiques. Seuls les composants de couleur sont spécifiques à un produit final.

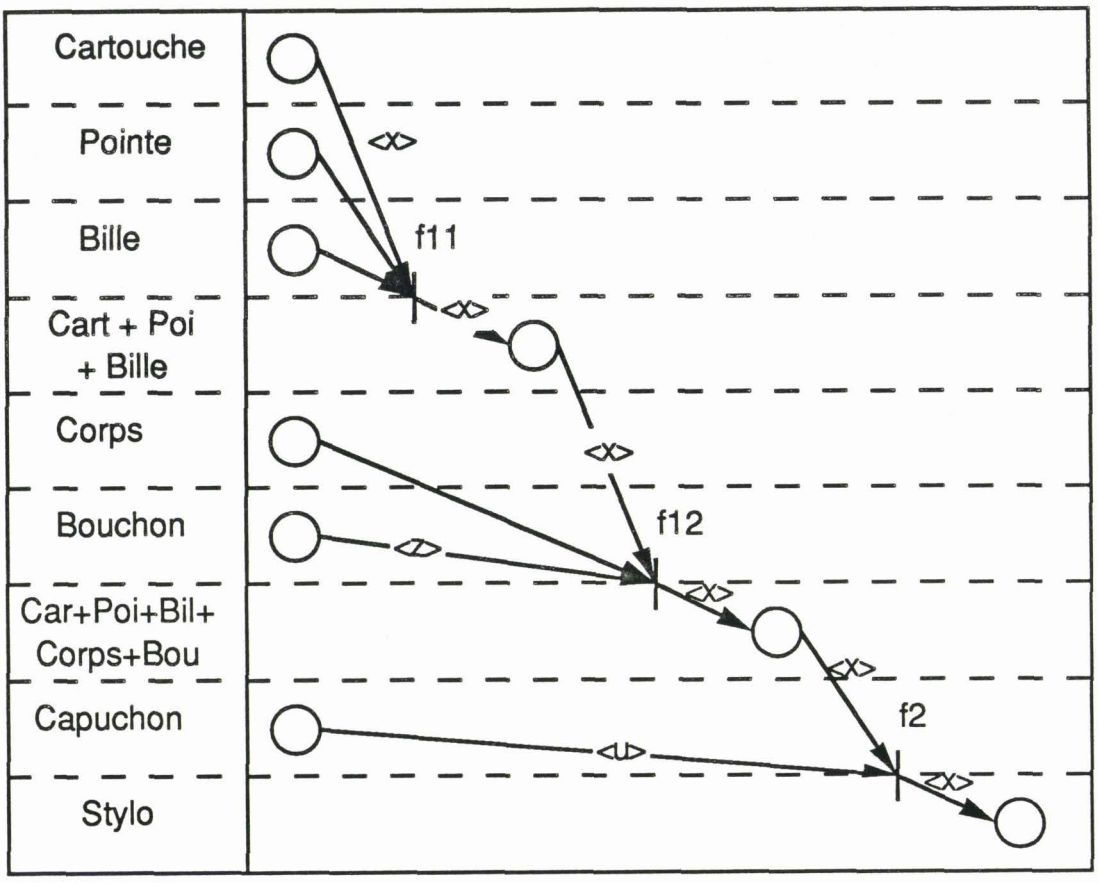
Ainsi

la cartouche d'encre, le bouchon, et le capuchon sont spécifiques,

la pointe, la bille, le corps sont communs aux deux processus de fabrication.

Il est donc intéressant d'agréger les deux RdP ordinaires en un RdP à Prédicats-Transitions sur la base du graphe d'événements précédent. Nous obtenons les gammes agrégées suivantes :

Ex 6.



avec les préconditions de tir :
f12 : <x=z>
f2 : <x=u>

et dom(x)=dom(z)=dom(u) = [Rouge, Bleu]

Figure 32

Les préconditions des transitions f12 et f2 traduisent le fait que les composants doivent être de la même couleur pour pouvoir être assemblés.

Notons que dans cet exemple, il s'agit de gérer des composants similaires par leurs traitements mais classés par types distincts : le type des composants bleus et le type des composants rouges.

Dans un type donné tous les composants sont considérés comme équivalents. La modélisation par RdP à Prédicats-Transitions dans laquelle on associe une information par type est suffisante.

Considérons maintenant un exemple où il y a nécessité d'identifier un objet spécifiquement parmi plusieurs.

Il s'agit de l'assemblage d'une ROM dédiée à une carte UC.

Ex 7.

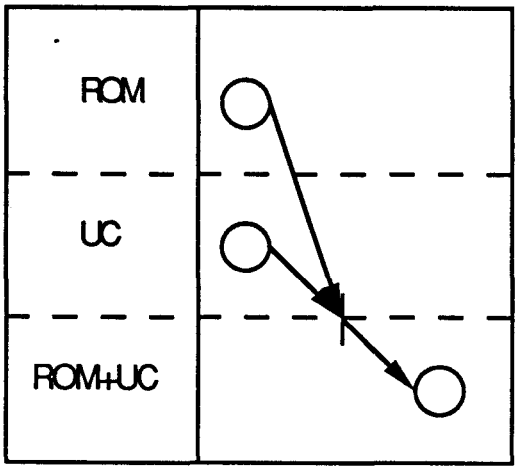


Figure 33

Ce réseau ne traduit pas convenablement la relation ROM<->UC avant l'assemblage. La ROM peut, en effet, être assemblée avec une carte UC quelconque. Pour cet exemple la modélisation par RdPO est indispensable.

On définit deux classes d'objets : Une_ROM et Une_Carte_UC :

Classe **Une_ROM** <r>	
Attributs	classe ou type
ma_carte_UC	Une_Carte_UC

Classe **Une_Carte_UC** <c>	
Attributs	classe ou type
mon_no_série	chaîne

Figure 34

Deux instances possibles pour le RdP à Objets suivant sont :

la_ROM_1 : instance de la classe Une_ROM
avec comme attribut ma_carte_UC = la_carte_UC_2

la_carte_UC_2 : instance de la classe Une_Carte_UC
avec comme attribut mon_no_série = 90XX007

Les deux objets peuvent être assemblés pour former un nouvel objet, de la classe Une_Carte_Complete :

Classe **Une_Carte_Complète** <c'>	
Attributs	Classe
ma_carte_UC	Une_Carte_UC
ma_ROM	Une_ROM

Figure 35

Le RdPO est donc le suivant :

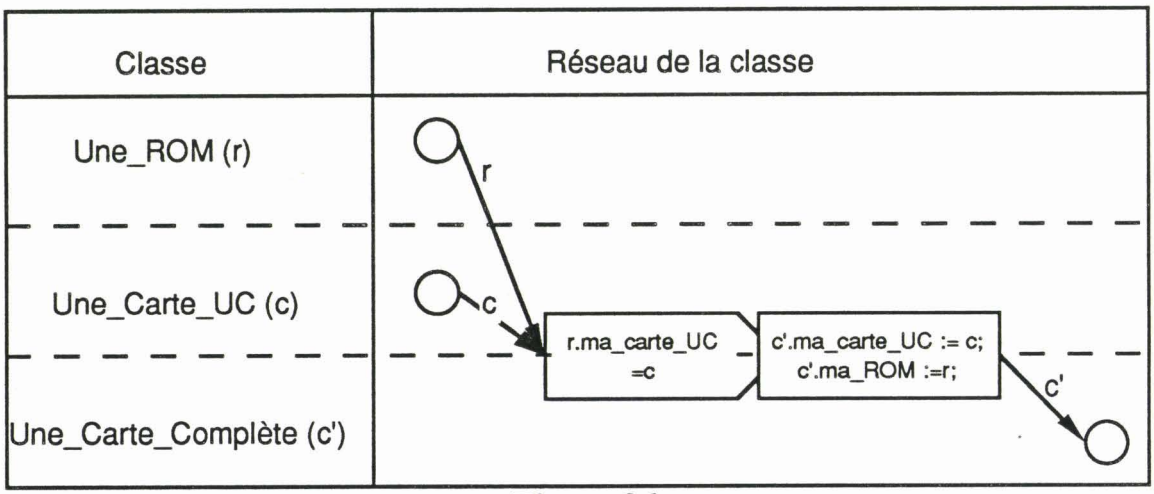


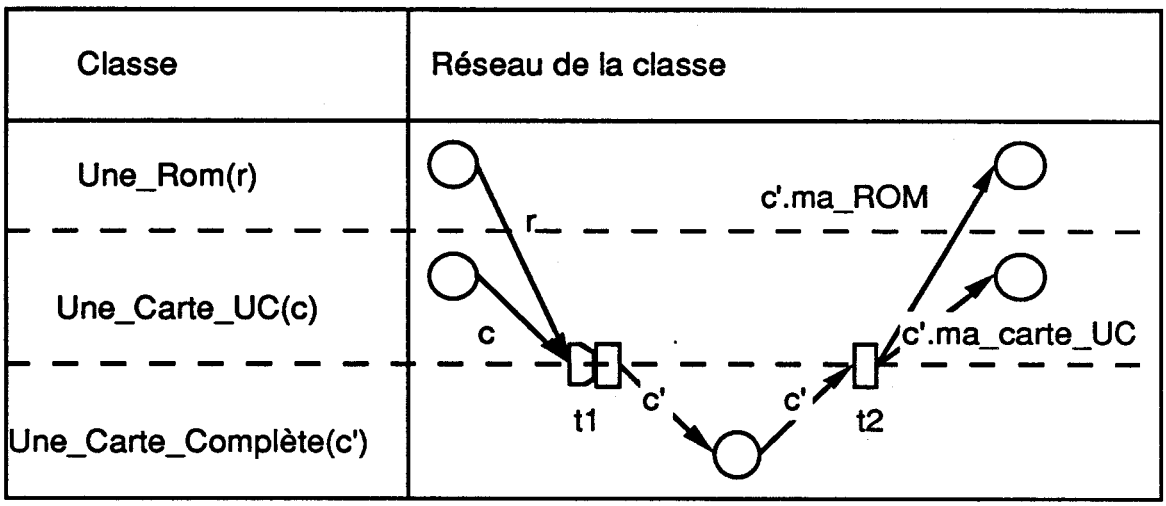
Figure 36

Conformément à l'esprit d'une fonction d'assemblage, c'est un nouvel objet qui est créé (de la classe Une_Carte_Complète). En effet les deux composants ont un rôle symétrique dans l'assemblage.

La précondition de la transition peut donc indifféremment être :
r.ma_carte_UC = c
ou c.ma_ROM = r, en ajoutant l'attribut ma_ROM à la classe Une_Carte_UC.

Notons enfin que les deux objets r et c ne sont pas détruits pendant l'assemblage. Les références ma_carte_UC et ma_ROM de l'instance c' de la classe Une_Carte_Complète en conservent une copie.

La carte complète peut donc être désassemblée en une carte UC et une ROM selon le réseau suivant :




```
t1 : précondition r.ma_carte_UC = c
     action begin c'.ma_carte_UC := c;
                  c'.ma_ROM := r;
          end;
```

```
t2 : précondition ()
     action ()
```

Figure 37

Les objets qui sont générés par la transition ne sont pas de nouvelles instances de leur classe. Il s'agit là d'une entorse à la définition formelle présentée au paragraphe IV.1.2., mais qui se justifie par le fait qu'en parcourant la partie action de la transition t1 les références ma_ROM et ma_carte_UC de l'objet c' existent effectivement au moment du tir de t2.

C'est pour cette raison que l'on utilise la référence c'.ma_ROM sur l'arc aval de t2 . En effet une solution du type de celle décrite sur la Figure 38 aurait pour effet de générer deux nouvelles instances de r et c sans tenir compte de leur rémanence effective.

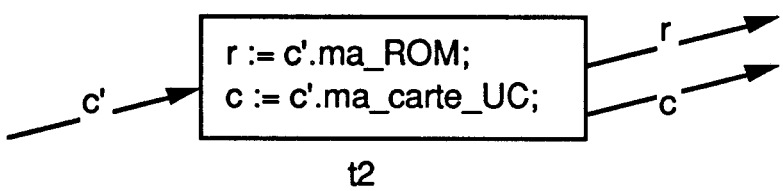


Figure 38

Dans les exemples précédents, les assemblages sont réalisés avec un seul exemplaire de chaque type de pièce. Considérons un autre exemple d'assemblage. Nous nous intéressons maintenant aux assemblages nécessitant plusieurs exemplaires d'une pièce donnée.

Ex 8. On considère une boite qui doit être fermée par un couvercle, et le couvercle doit être vissé par 2 ou 4 vis, selon sa constitution. Il s'agit d'un assemblage soit d'ordre 4 soit d'ordre 6. En effet les assemblages binaires

boite+couvercle, et couvercle+vis$_i$
peuvent être réalisés dans n'importe quel ordre.

La gamme logique de cet assemblage est schématisée sur la Figure 39.

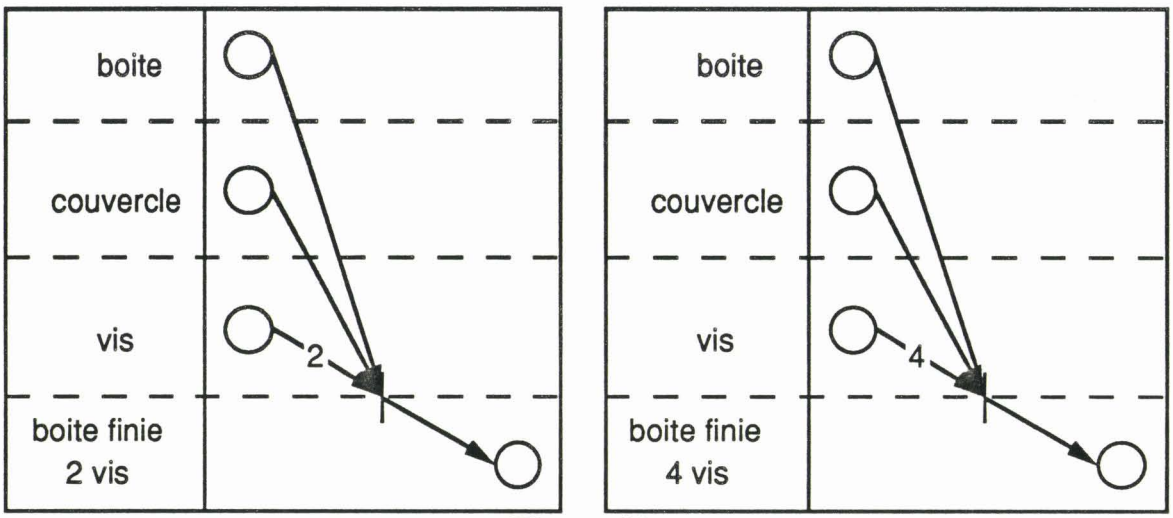
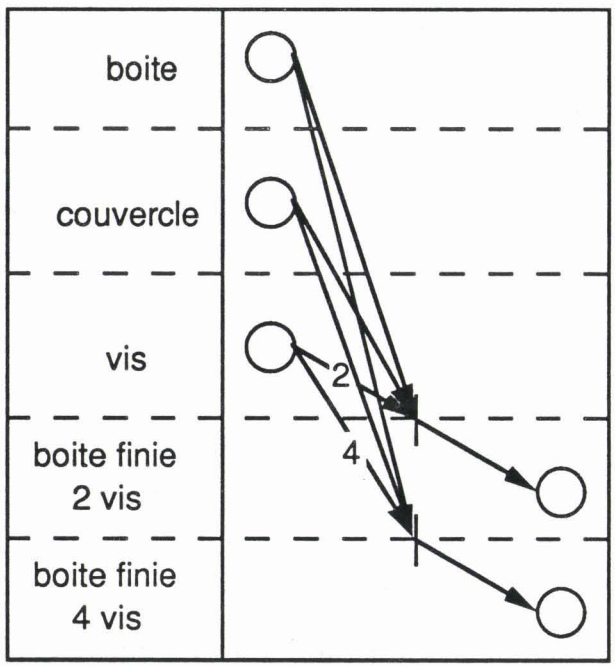


Figure 39

Ces réseaux ne sont plus des graphes d'événements puisque le poids des arcs est différent de 1. Chacun des réseaux peut être ramené à un graphe d'événement par multiplication des places. L'agrégation des deux graphes en un seul n'est pas possible car les poids des arcs diffèrent.

On obtient :



Agrégation des deux gammes logiques

Figure 40

Dans la modélisation par RdP et RdP coloré, la pondération des arcs est constante et indépendante du marquage. Pour les RdP à Objets, la pondération d'un arc n'a pas de sens puisque les objets sont individualisés et le marquage d'une place est un ensemble de n-uplets (d'arités différentes éventuellement) d'objets individualisés. La modélisation par RdPO de l'exemple 8 est donc :

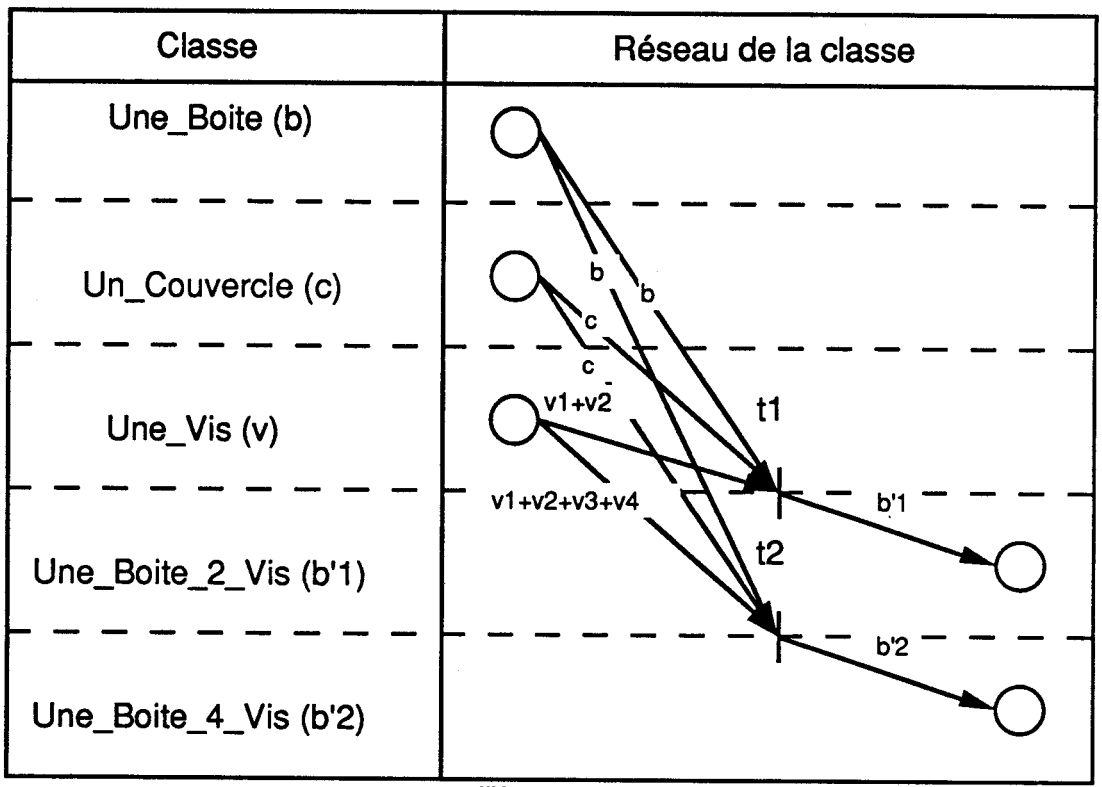


Figure 41

avec les classes

Une_Boite sans attribut,

Un_Couvercle sans attribut,

Une_Vis sans attribut,

Une_Boite_2_Vis

attribut	classe
ma_boite	Une_Boite
ma_vis_1	Une_Vis
ma_vis_2	Une_Vis
mon_couvercle	Un_Couvercle

Une_Boite_4_Vis

attribut	classe

ma_boite	Une_Boite
ma_vis_1	Une_Vis
ma_vis_2	Une_Vis
ma_vis_3	Une_Vis
ma_vis_4	Une_Vis
mon_couvercle	Un_Couvercle

L'action de t1 est :

```
b1'.ma_boite := b;
b1'.mon_couvercle := c;
b1'.ma_vis_1 := v1;
b1'.ma_vis_2 := v2;
```

L'action de t2 est :

```
b2'.ma_boite := b;
b2'.mon_couvercle := c;
b2'.ma_vis_1 := v1;
b2'.ma_vis_2 := v2;
b2'.ma_vis_3 := v3;
b2'.ma_vis_4 := v4;
```

Considérons la transition t1 seule :

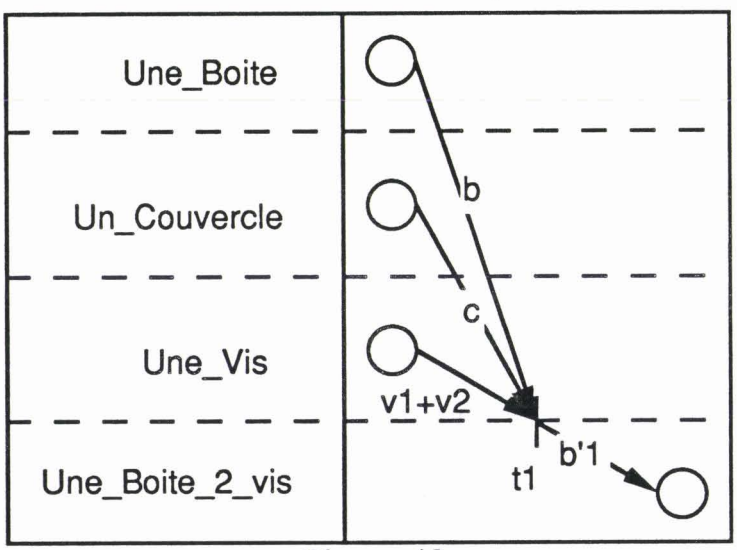


Figure 42

La condition de tir de t1 est la suivante :

il existe un objet boite qui peut se substituer à la variable b
et il existe un objet couvercle qui peut se substituer à la variable c
et il existe deux objets de la classe Une_Vis qui peuvent se substituer à v1 et v2.

Le franchissement de la transition t1 consiste à retirer les objets substitués à b et c, et deux objets de la classe Une_Vis.

Nous avons illustré dans quel esprit se fait le choix de l'outil de modélisation en fonction du cas à traiter sur la base de quelques exemples significatifs.

En schématisant, la puissance de l'outil choisi est directement fonction du degré de détail nécessaire à la réalisation des fonctions unaires ou n-aires. Il convient donc de disposer de classes de modèles adaptées à la complexité du problème traité.

Notons enfin que la puissance utilisée (RdPO en particulier) peut être parfois excessive pour un composant d'un assemblage (la vis) et nécessaire pour d'autres, lorsque par exemple la boite et le couvercle sont dépendants (couleur, cote...).

La section suivante traite de productions couramment utilisées telles que le traitement par lot (§IV.1.4.1.) et le traitement d'assemblages prédéfinis (§IV.1.4.2.) qui sont à la source de nouvelles difficultés de modélisation.

IV.1.4. Modélisation de gammes logiques emboîtées

Dans l'industrie manufacturière, il est rare que des pièces soient produites isolément et en grande quantité. Le plus souvent, il s'agit de productions à la demande de pièces regroupées sous la même commande. Une commande client décrit la nature et le nombre d'exemplaires de chaque type de pièces. Réaliser une commande client revient à produire puis réunir tous les exemplaires de la commande.

Il est donc habituel qu'un ensemble de pièces soit traitées en étant regroupées dans le temps (production de pièces avant tel événement) ou dans l'espace (n pièces appartiennent à un plateau et doivent se regrouper sur un autre plateau).

Dans ces deux cas nous remarquons que les pièces sont usinées isolement mais qu'elles appartiennent virtuellement à un ensemble de pièces ou lot. Ce mode de gestion est appelé production par lot.

Par définition nous dirons qu'un lot est un ensemble de pièces (de même nature ou non) qui possèdent un lien logique entre elles. Les pièces qui appartiennent à un même lot sont dites Virtuellement Liées (PVL).

IV.1.4.1. Traitement de lots

La modélisation d'un lot peut être menée à deux niveaux :

- en premier lieu en considérant le lot en tant qu'entité indivisible comme s'il s'agissait d'une pièce. Son graphe d'état est modélisé par un RdP de la même manière que pour une pièce;

- en second lieu en considérant les pièces de façon élémentaire.

La convention suivante a été choisie :

- si le lot subit une opération (étiquetage, pesage ...), alors les pièces qui le composent n'en subissent aucune;

- inversement si une des pièces d'un lot subit une opération, alors le lot n'en subit aucune.

Ces hypothèses permettent de garantir la cohérence de l'état du lot et des pièces avec leur modèle respectif.

Prenons un exemple simple de lot.

Ex 9. : un plateau contient 5 pièces identiques. Le plateau est d'abord pesé, puis les 5 pièces sont usinées dans un ordre quelconque, enfin le plateau est reconstitué et subit un traitement thermique.

La gamme logique du lot plateau comporte donc 5 états : brut, pesé, en cours d'usinage, usiné, traité.

Les pièces qui composent le lot ne subissent qu'une seule opération, entre le pesage et le traitement thermique.

Les gammes logiques sont donc les suivantes :

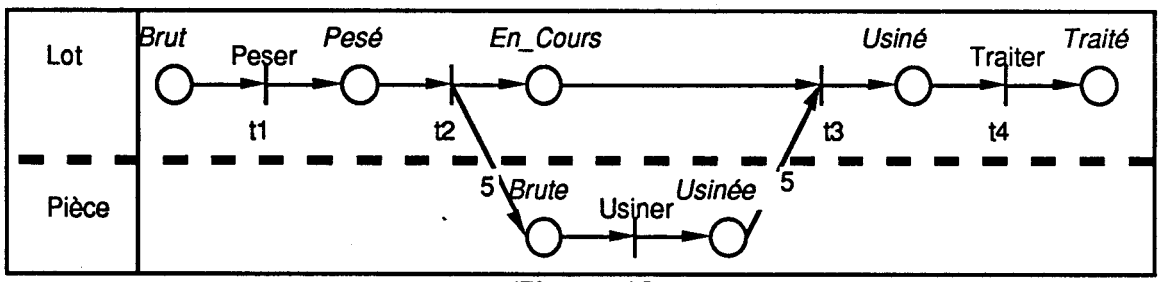


Figure 43

La synchronisation entre les gammes logiques du lot et des pièces est analogue à une fonction de désassemblage de par sa structure. Ici cependant aucune action ne sera associée à cette transition. Le rôle de t2 est de générer 5 jetons. Celui de t3 est de reconstituer le lot à partir des 5 pièces terminées.

Ce réseau illustre clairement le passage de la gestion de lot à la gestion de pièce et inversement (trait gras). La section IV.1.4.3. présentera une approche hiérarchique plus détaillée.

Nous allons maintenant préciser le choix du modèle RdP en fonction de la complexité du cas à traiter. La modélisation par modèle RdP ordinaire pour l'exemple 9 est valide à condition que toutes les pièces aient à subir la même opération d'usinage, et que les plateaux puissent être reconstitués avec 5 pièces quelconques. En particulier les pièces peuvent provenir de n'importe quel lot. Le lien lot $\leftrightarrow$ pièces n'est donc pas pris en compte par le RdP ordinaire.

Ex 10. : sur la base de l'exemple précédent, on souhaite traiter des lots de 2 pièces, en supposant que les pièces, une fois usinées, retournent sur leur plateau d'origine. Dans ce cas l'utilisation des RdPO est indispensable afin d'assurer le lien pièce $\leftrightarrow$ lot. La définition minimale des classes d'objets est donc la suivante :

Classe **Un_Lot**	
Attribut	Classe
ma_pièce_1	Une_Pièce
ma_pièce_2	Une_Pièce

Classe **Une_Pièce**	
Attribut	Classe
mon_lot	Un_Lot

Figure 44

Dans la configuration initiale du réseau, une instance de la classe Un_Lot doit faire référence à deux instances existantes de la classe Une_Pièce. Les références croisées sont donc établies dans la configuration initiale.

Le RdPO modélisant les gammes logiques est le suivant :

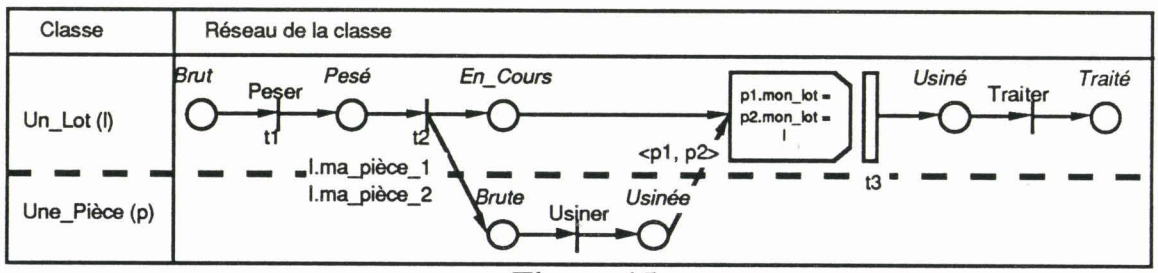


Figure 45

Ici encore, la transition t2 génère deux objets distincts existants. En effet, les pièces sont traitées isolément indépendamment les unes des autres lors de l'usinage, et non par couple. Nous recherchons le parallélisme d'exécution maximal.

La transition t3 est construite dans le même esprit : si il est possible de substituer deux instances de la classe Une_Pièce à p1 et p2, alors elles doivent vérifier la précondition de t3.

Cet exemple indique clairement que même si la gamme logique des pièces peut être modélisée par RdP ordinaire, il devient nécessaire de la modéliser par RdPO afin d'établir la relation hiérarchique avec la structure de données de l'objet père qui est ici Un_Lot. Là encore la puissance du modèle risque d'être excessive vis à vis du problème posé au niveau local, si l'on ne considère que le traitement des pièces.

Ex 11. : Dans cet exemple, on souhaite décomposer un lot L de 6 pièces en deux lots L1 et L2 de 2 et 4 pièces.

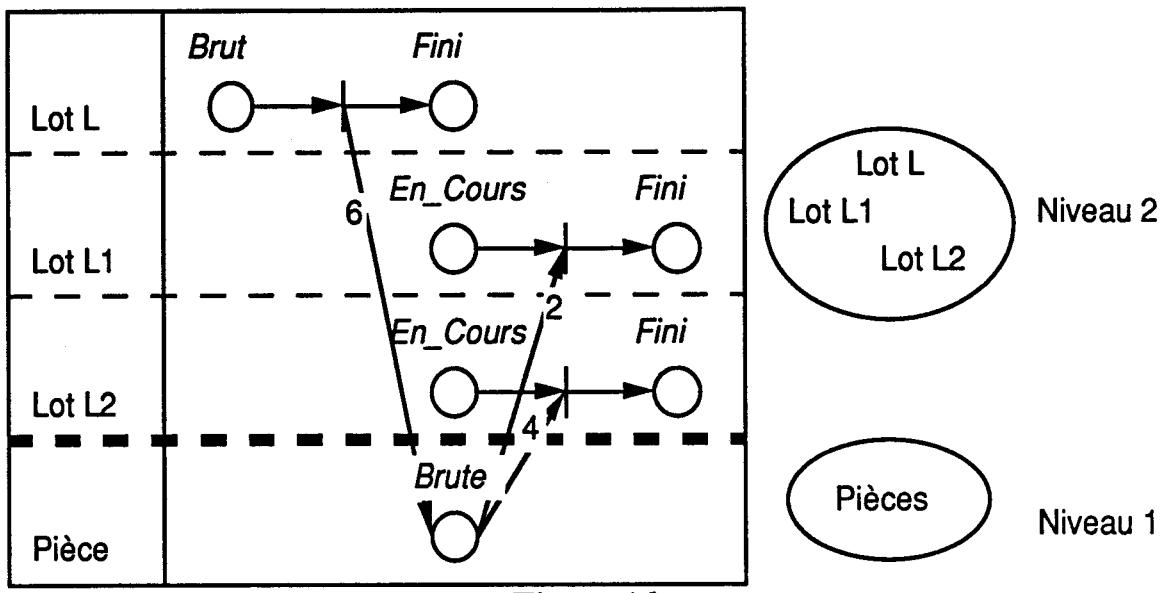


Figure 46

Ce réseau explicite le fait que les lots L1 et L2 restent dans l'état En_Cours tant que le nombre voulu de pièces n'est pas atteint. Ici encore la modélisation par RdP ordinaire ne permet pas de diriger une pièce vers un lot particulier.

Dans cette section nous avons présenté différentes manières de gérer un lot selon le degré de détail nécessaire.

Voyons maintenant comment sont gérés les assemblages prédéfinis.

IV.1.4.2. Traitement d'assemblages prédéfinis

Un assemblage prédéfini est un assemblage de pièces qui doivent être réellement assemblées au sens de la solidarisation physique, alors qu'elles étaient virtuellement liées entre elles au préalable (appartenance à un même support par exemple). C'est le cas par exemple de la ROM dédiée à une carte UC.

L'ensemble ROM+UC constitue ainsi un lot de pièces. On peut imaginer en effet que la commande client soit de fabriquer un ensemble ROM+UC avec un numéro de série particulier. Ce numéro de série doit figurer dans la ROM et sur la carte. Le lot est donc initialement un objet qui mémorise le numéro de série. Puis il est désassemblé fictivement en une ROM et une carte UC qui doivent être marquées. Une fois terminées (donc ici physiquement assemblées), le lot est constitué et terminé.

Il est clair que la modélisation par RdPO se justifie pleinement puisque l'information numéro de série peut prendre une infinité de valeurs.

Ex 12.

Nous supposons dans le RdPO suivant que le lot est formé initialement d'une ROM et d'une carte UC vierges. Le réseau traduit le processus de fabrication de la carte complète.

Les classes d'objets sont les suivantes :

Classe **Un_Lot**	
Attribut	Classe ou type
mon_no_série	chaîne
ma_ROM	Une_ROM
ma_carte_UC	Une_Carte_UC
ma_carte_ complète	Une_Carte_ Complète

Classe **Une_ROM**	
Attribut	Classe ou type
mon_lot	Un_Lot
mon_no_série	Chaîne

Classe **Une_Carte_UC**	
Attribut	Classe ou type
mon_lot	Un_Lot
mon_no_série	Chaîne

Classe **Une_Carte_complète**	
Attribut	Classe ou type
mon_lot	Un_Lot
mon_no_série	Chaîne

Figure 47

Le réseau de ces classes est le suivant :

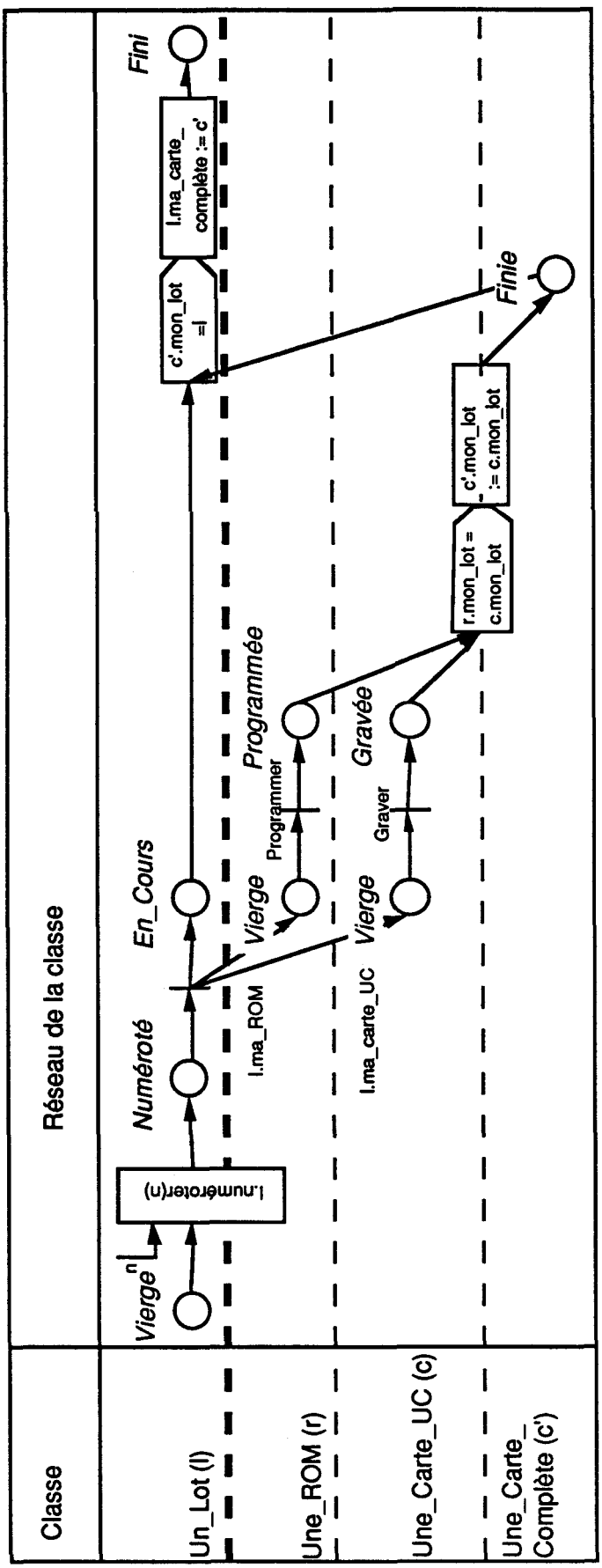

Classe
Réseau de la classe
Un_Lot (l)
Une_ROM (r)
Une_Carte_UC (c)
Une_Carte_ Complète (c')
Vierge n
l.numéroter(n)
Numéroté
l.ma_ROM
En_Cours
Vierge
Programmer
Programmée
l.ma_carte_UC
Vierge
Graver
Gravée
r.mon_lot = c.mon_lot
c'.mon_lot := c.mon_lot
Finie
c'.mon_lot =l
l.ma_carte_ complète := c'
Fini

Figure 48

Les instances initiales :

Le_lot_1 : Un_Lot

Attribut	Valeur
mon_no_série	()
ma_ROM	la_ROM_1
ma_carte_UC	la_carte_UC_1
ma_carte_complète	()

La_ROM_1 : Une_ROM

Attribut	Valeur
mon_lot	le_lot_1
mon_no_série	()

La_carte_UC_1 : Une_Carte_UC

Attribut	Valeur
mon_lot	le_lot_1
mon_no_série	()

Méthodes :

Classe	Méthode	Code
Un_Lot	l.numéroter(n)	begin l.mon_no_série := n; l.ma_ROM.numéroter(n); l.ma_carte_UC.numéroter(n); end
Une_ROM	r.numéroter(n)	r.mon_no_série := n;
Une_Carte_UC	c.numéroter(n)	c.mon_no_série := n;

Nous pouvons remarquer que lors de l'assemblage de pièces appartenant à un même lot, les relations nécessaires entre pièces pour pouvoir réaliser l'assemblage effectif se ramènent ici à comparer le nom des lots dont les pièces sont originaires. Le schéma des relations est alors très simple :

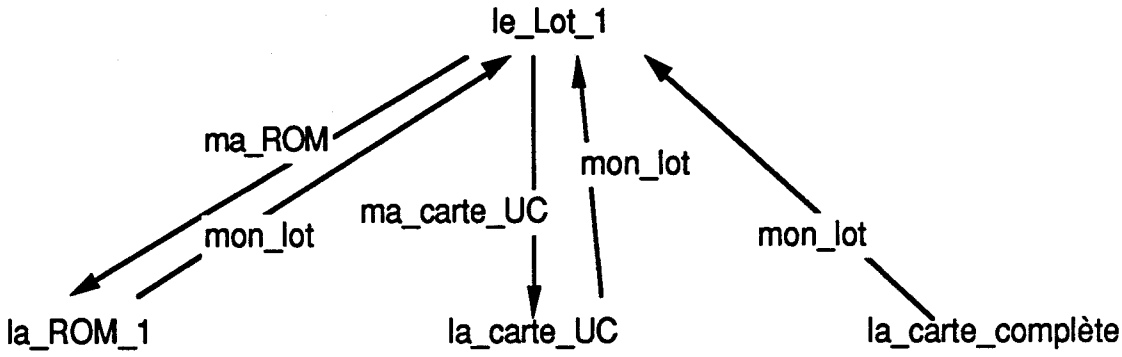


Figure 49

Nous verrons au §IV.1.4.3. que ces relations permettent d'étendre immédiatement la description d'un lot aux méga-lots (lot composé de lots).

Pour schématiser la gestion d'assemblages prédéfinis, il s'agit de :

- créer un lot qui regroupe l'ensemble des pièces à assembler,

- définir les références croisées entre le lot et toutes les pièces à assembler,

- lors d'un assemblage, la référence du lot doit être transmise à la pièce générée,

- le seul attribut indispensable à la dernière pièce de l'assemblage (le produit fini) est la référence du lot.

Il est clair que les attributs présentés précédemment sont les attributs strictement indispensables pour gérer le lot, et qu'il est possible d'en ajouter un nombre quelconque.

Dans les exemples précédents nous avons supposé que les lots étaient constitués au préalable. Voyons maintenant, sur l'exemple 12 comment il est possible de créer un lot.

Constituer un lot revient à rassembler un nombre connu de pièces de chaque classe, puis à réaliser les liens virtuels entre le lot et les pièces qui le constituent. Le RdPO qui traduit la constitution d'un lot dans le cas de l'exemple 12 est le suivant :

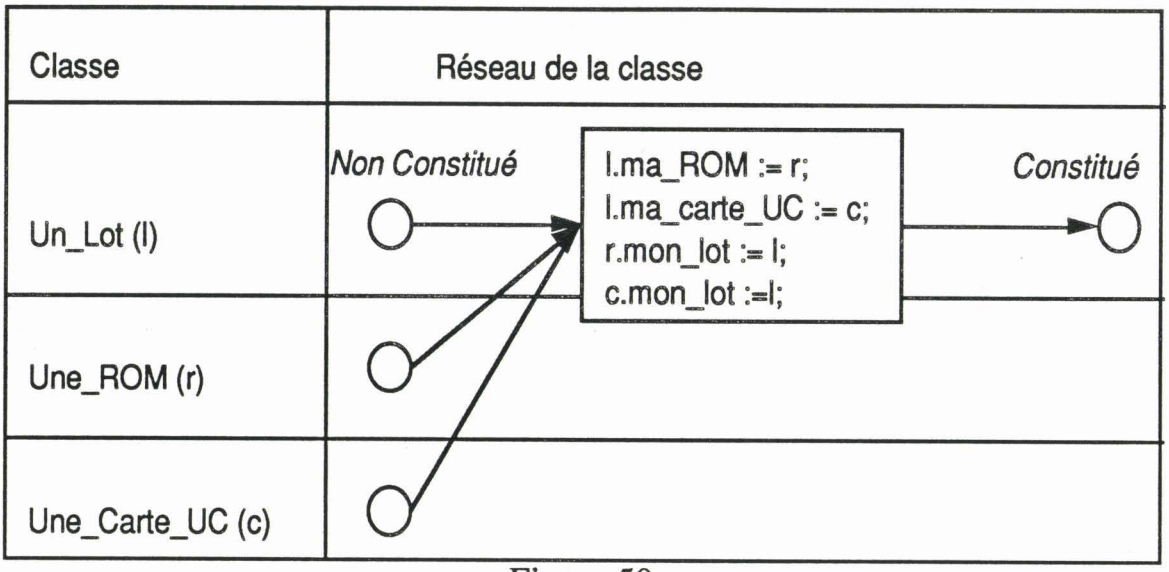


Figure 50

Marquage initial : l, r, c dont les attributs traduisant les liaisons virtuelles ne sont pas instanciés.

La condition de tir est suffisante pour constituer un lot :

il existe un exemplaire de la classe Une_ROM
et il existe un exemplaire de la classe Une_carte_UC.

Il est bien sur possible d'y adjoindre une précondition si nécessaire.

Voyons dans la section suivante comment les lots sont composés entre eux.

IV.1.4.3. Combinaison de gammes logiques emboîtées

Puisqu'un lot est considéré en tant qu'entité indivisible (à une exception près cf §IV.1.4.1.), il est légitime d'imaginer qu'il peut lui-même faire partie d'un lot. La définition du lot doit alors introduire la possibilité de liens avec son lot père et ses lots (ou pièces) fils (ou filles). Nous obtenons ainsi une vision pyramidale des gammes logiques en plusieurs niveaux.

Ex 13. Prenons le cas d'une commande client composée de plusieurs sous-commandes, elles même composées d'un ensemble de pièces de même type.

La gamme logique de la commande a donc trois niveaux :

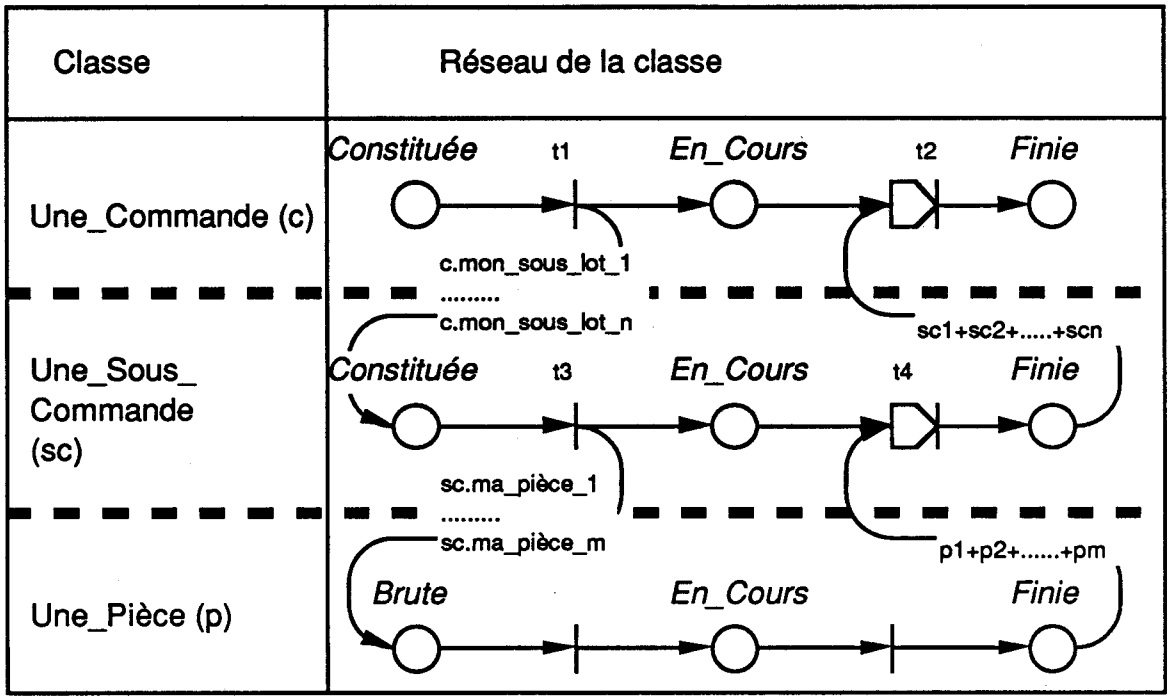


Figure 51

avec les préconditions :
t2: sc1.mon_lot = c
.....
scn.mon_lot = c

t4: p1.mon_lot = sc
.....
pm.mon_lot = sc

Le schéma des liaisons virtuelles en niveaux est le suivant :

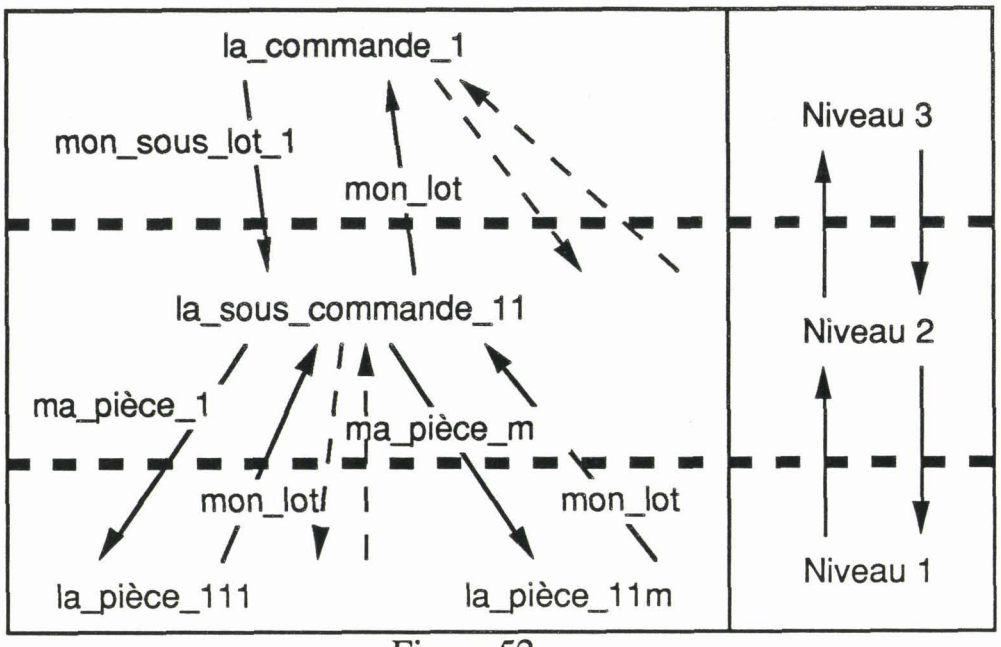


Figure 52

De manière générale, un lot

- doit faire référence à son lot père, si il existe,

- doit posséder un attribut par sous-lot qu'il possède, si ils existent,

- ne doit subir aucune transformation lorsqu'au moins un de ses sous-lots en subit.

Nous obtenons bien sur un arbre de lots dont les feuilles sont les pièces.

Considérons un exemple de traitement par lot en plusieurs niveaux.

Ex 14. Considérons une société de Vente Par Correspondance. Chaque client réalise une commande de vêtements qui doivent être expédiés ensemble dans un paquet. Les paquets sont regroupés par ville dans des cartons. Chaque carton est étiqueté (adresse, composition ...), puis expédié par camion dans un entrepôt de triage.

Nous avons donc un nombre important de niveaux :

vêtements $\leftrightarrow$ paquet $\leftrightarrow$ carton $\leftrightarrow$ camion

Nous supposons que les assignations carton $\leftrightarrow$ camion, paquet $\leftrightarrow$ carton, vêtements $\leftrightarrow$ paquet sont faites et en particulier que les stocks sont suffisants. Cette tâche est assurée par le module de planification à long terme.

Le RdPO des gammes logiques de ces différentes classes est le suivant :

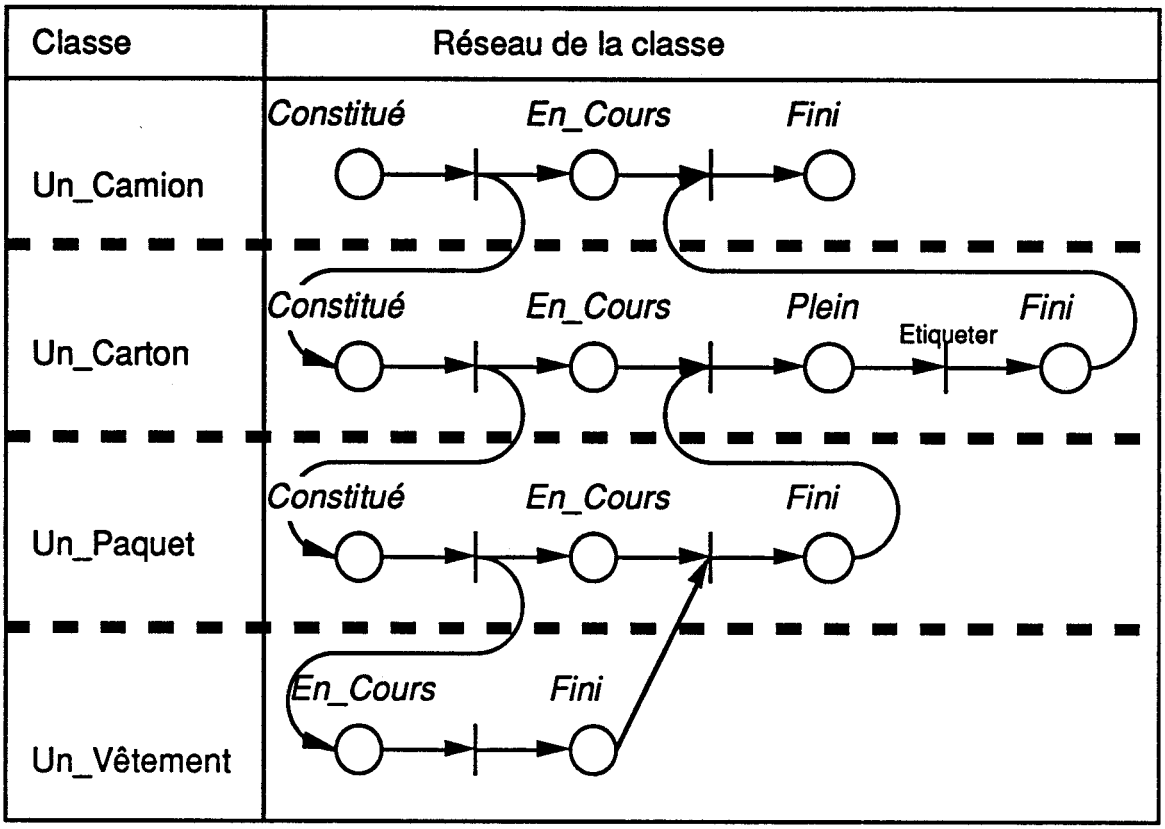


Figure 53

Les préconditions et les attributs des objets ont la même forme que dans l'exemple 13. Il est en fait rare qu'un carton soit alloué à un ensemble de paquets déterminé. En général, l'approvisionnement en emballages est fait indépendamment de la gestion de son contenu, et la production de vêtements est faite sans que l'existence d'emballages soit garantie.

Cependant, le lot camion est constitué de telle manière qu'il est possible d'assembler les vêtements dans leurs emballages et qu'ils puissent loger dans le volume du camion.

Le lot camion ne contient donc que les références des vêtements qui le composent mais pas celles des emballages. Cela correspond plus à la réalité industrielle, où les emballages sont gérés en stock illimité.

Le RdPO est donc le suivant :

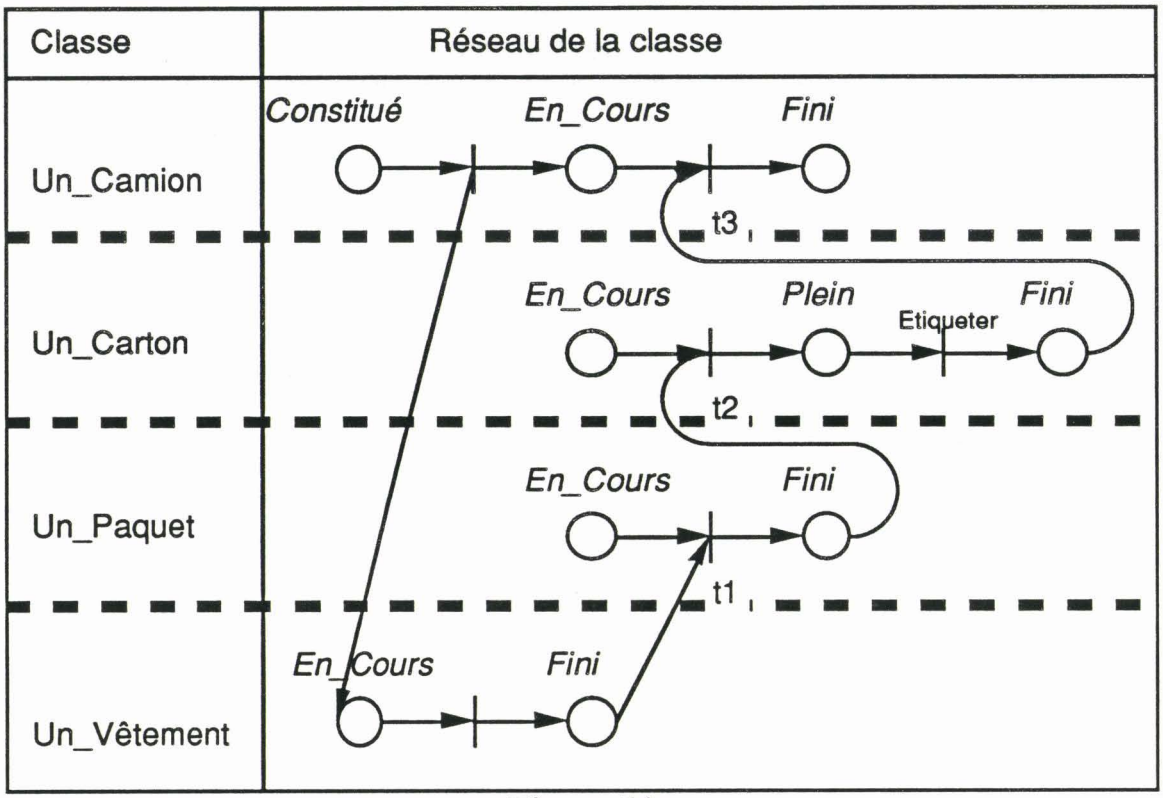


Figure 54

Le réseau traduisant les gammes logiques dépend donc fortement des hypothèses concernant la planification et l'ordonnancement. Il s'agit en effet de savoir si un sous-lot d'un lot est constitué lorsque les pièces terminales sont produites.

Pour la figure 54, les préconditions de tir sont les suivantes :

t1 : tous les vêtements appartiennent à la même cliente, à concurrence de la capacité d'un paquet, le dernier paquet d'une cliente est terminé lors du chargement du dernier article,

t2 : tous les paquets sont à destination de la même ville, à concurrence de la capacité d'un carton, le dernier carton est terminé lors du chargement du dernier paquet,

t3 : tous les cartons appartiennent au même camion, capacité non dépassée par hypothèse.

La référence du camion est transmise lors du tir de t1, t2 et t3.

Pour conclure cette étude de la modélisation des gammes logiques, nous avons synthétisé sur la Figure 55, l'ensemble des exemples traités, en indiquant la nature de la fonction traitée et la puissance la plus adéquate de l'outil de modélisation.

Exemple	Nature	Cohabitation de plusieurs gammes logiques	Modèle RdP	Libellé
1	Transformationnelle	Non	Ordinaire	Tournage + Fraisage
2	Transformationnelle	Non	Ordinaire	Tournage + Fraisage indifférent
3	Transformationnelle + Informationnelle	Non	Coloré	Tournage + test diamètre
4	Assemblage	Non	Ordinaire	Assemblage Simple
5	Assemblage	Non	Ordinaire	Stylo 1 couleur
6	Assemblage	Oui	Prédicats	Stylo 2 couleurs
7	Assemblage Spécifique	Non	Objets	Assemblage ROM + UC
8	Assemblage non Spécifique	Oui	Ordinaire	Couvercle à 2 ou 4 vis
9	Traitement par lot non Spécifique	Non	Ordinaire	Lot->5 pièces->Lot qcq
10	Traitement par lot Spécifique	Non	Objets	Lot->pièces->lot d'origine
11	Traitement par lot non Spécifique	Non	Ordinaire	L->L1 + L2
12	Assemblage binaire spécifique traité par lot	Non	Objets	ROM + UC spécifiques
13	Traitement de lots spécifiques	Non	Objets	Commande<->sous-commande <-> pièces
14	Traitement de lots spécifiques	Non	Objets	VPC

Figure 55

Ces exemples couvrent la globalité des problèmes rencontrés lors de la spécification des gammes logiques. Le modèle nécessaire dépend donc fortement de la nature de la production.

Il apparait de toute évidence que, dans un grand nombre de cas, la modélisation par RdPO soit nécessaire selon le bilan par complexité croissante ci-après :

modèle RdP ordinaire pour les cas (1 2 4 5 8 9 11) = E1,
modèle RdP colorés pour les cas 3 + E1 = E2,
modèle RdP à prédicats pour les cas 6 + E2 = E3,
modèle RdP à Objets pour les cas (7 10 12 13 14) + E3.

IV.1.5. Application à l'exemple

L'exemple choisi illustre l'intérêt de l'approche pour une synthèse des apports significatifs que nous cherchons à mettre en évidence.

Du point de vue fonctionnel, l'exemple comporte trois classes d'opérations :

- un traitement par lot,

- un assemblage binaire pur (sans contrainte d'antériorité d'une pièce sur l'autre),

- plusieurs fonctions unaires.

Nous verrons dans la section IV.2. qui traite des gammes opératoires, que nous avons imposé l'architecture matérielle de production, comportant :

- un convoyeur complexe, disposant d'un magasin de palettes vides,

- de palettes à plusieurs emplacements de stockage

- une machine d'assemblage,

- un robot à 2 pinces,

- une zone de stockage

- un tour à commande numérique,

- une machine à tarauder,

- un robot graisseur.

Revenons à l'aspect purement fonctionnel. La fonction de l'unité de production est de traiter l'ensemble des pièces brutes de fonderie suivantes :

- des écrous non taraudés, qui doivent subir un taraudage,

- des vis non filetées, qui doivent subir un filetage,

- puis un écrou et une vis sont assemblés (ici vissés ensemble), pour être graissés,

- le boulon est alors terminé.

Ces données nous permettent de construire les gammes logiques des pièces concernées. Ici, nous traitons simultanément des écrous, des vis et des boulons. Les gammes logiques sont les suivantes :

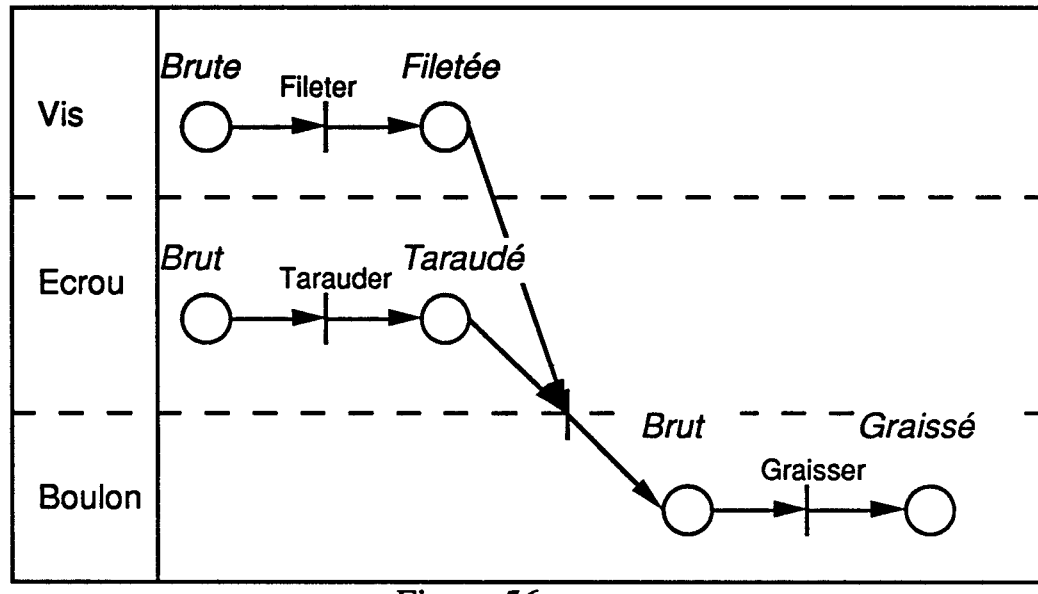


Figure 56

Il s'agit d'un RdP ordinaire.

Nous souhaitons de plus gérer les ensembles de pièces brutes de fonderie par lot. Un lot est composé d'un nombre identique de vis et d'écrous. Nous faisons l'hypothèse que les lots arrivent déjà constitués dans l'unité de fabrication. Deux lots peuvent avoir un nombre de pièces brutes différent. De plus, on souhaite panacher les lots, i.e. produire et gérer simultanément plusieurs lots de caractéristiques différentes. Nous pouvons ainsi associer une date de réalisation au plus tôt à chaque lot, et rendre prioritaires les pièces qui sont les plus urgentes dans l'unité de fabrication.

L'outil de modélisation est bien sûr le RdPO, puisqu'il faut assurer le lien entre le lot et les pièces, et associer des informations spécifiques à chaque pièce (son lot, sa date).

Les classes d'objets sont les suivantes :

Classe Un_Lot	
Attribut	Classe ou type
ma_date_due	date
mes_vis	instances de Une_Vis
mes_écrous	instances de Un_Ecrou
ma_date_réalisation	date

Classe Une_Vis	
Attribut	Classe ou type
ma_date_due	date
mon_lot	Un_Lot

Classe Un Ecrou	
Attribut	Classe ou type
ma_date_due	date
mon_lot	Un_Lot

Classe Un Boulon	
Attribut	Classe ou type
ma_date_due	date
mon_lot	Un_Lot

Figure 57

Les gammes logiques sont les suivantes :

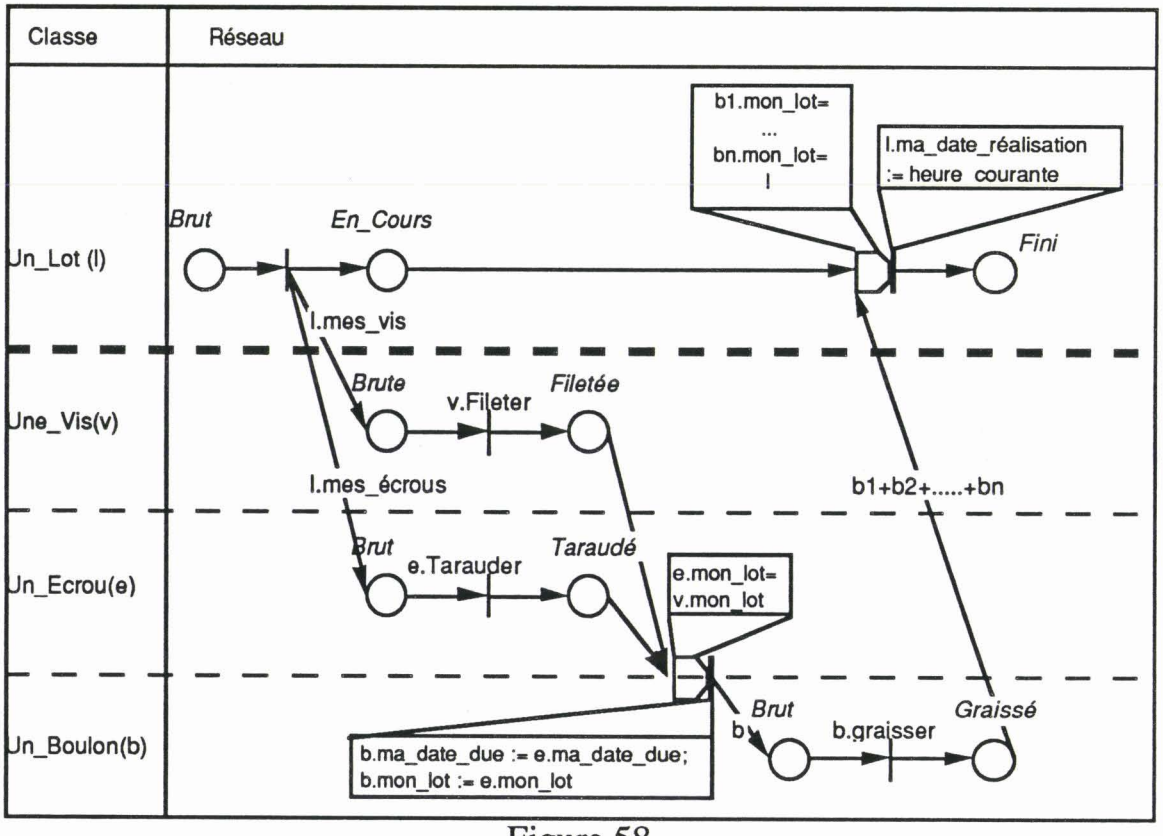


Figure 58

Ce réseau traduit :

- le désassemblage fictif du lot en plusieurs vis et écrous bruts,

- le traitement de chaque pièce brute,

- l'assemblage d'un objet boulon possédant les mêmes attributs que l'écrou d'origine (ou la vis),

- le traitement du boulon,

- le réassemblage fictif du lot lorsqu'il est possible de substituer toutes les variables b1 à bn, et que de plus tous les objets de ces variables appartiennent au même lot. Le lot est alors effectivement terminé (date de réalisation).

Cet exemple nous permet d'illustrer la nécessité de disposer d'un modèle puissant (le RdPO), dans le cas d'un traitement par lot, même si l'assemblage lui-même peut être modélisé par un RdP ordinaire.

IV.2. Génération des gammes opératoires

Sur la base de la description fonctionnelle des traitements appliqués à chaque objet produit, nous allons réaliser la prise en compte progressive des moyens de production et de manipulation (aspect opérationnel).

Nous proposons dans ce sens de mener à bien la synthèse entre une gamme logique décrivant l'élaboration d'un produit, et la description arborescente du procédé caractéristique des moyens de production utilisables.

Le résultat de cette phase est appelé gamme opératoire.

Définition :

Une gamme opératoire décrit la succession des états d'avancement d'une pièce, ainsi que la succession des lieux opératoires par lesquels elle transite. Elle associe ainsi les moyens physiques (ressources de production) aux fonctions de la gamme logique visant à l'élaboration effective des produits.

IV.2.1. Origine des données nécessaires

Comme nous l'avons présenté au chapitre II Fig 15, la spécification fonctionnelle (Phase 1) et la spécification opérationnelle (Phase 1') sont réalisées en parallèle et indépendamment.

La spécification fonctionnelle produit un modèle RdP que nous avons appelé gamme logique.

La spécification opérationnelle produit un modèle entité-relations arborescent des objets physiques commandables, ainsi qu'un modèle des relations d'accessibilité entre lieux physiques de stockage.

La phase de génération des gammes opératoires (Phase 2) exploite donc ces deux modèles et produit un modèle RdP dont la puissance est identique à celle de la gamme logique d'origine.

Ce modèle sera ensuite contraint par la prise en compte de la gestion des accès aux lieux physiques en phase 3 (voir chapitre V).

Il est clair que si la phase de génération des gammes opératoires échoue (dans le cas d'une inadéquation entre les fonctions à réaliser et les moyens disponibles), alors il faut soit reconsidérer les gammes logiques de départ, soit envisager d'étendre ou de modifier la spécification opérationnelle. Les retours

arrière vers la phase 1 et la phase 1' sont donc nécessaires et implicites dans la démarche CASPAIM-2.

IV.2.2. Les hypothèses et le modèle

Un objet peut subir, dans le contexte de la productique, une action qui modifie son état d'avancement, ou une action qui le fait changer de lieu, ou les deux à la fois.

Nous avons supposé, au début du chapitre, qu'il subit ces deux opérations de façon disjonctive, cette hypothèse nous permet de modéliser sa gamme logique par un graphe d'état. De la sorte l'état de l'objet est en bijection avec le marquage de son modèle.

La prise en compte des moyens opérationnels doit donc conduire également à un modèle graphe d'état. Il s'agit d'un RdP pour lequel une transition modélise de manière exclusive un transfert d'un lieu à un autre, ou une opération qui modifie son état d'avancement.

Une place marquée modélise donc le fait que l'objet (la marque de la place) se trouve dans un état d'avancement et un lieu précis. Une transition caractérise ici une seule modification de l'état d'un objet de position ou d'avancement dans sa gamme.

Nous avons choisi de nommer les places par l'état d'avancement de l'objet (au dessus de la place en italique), et son lieu de support (en dessous de la place).

Pour le réseau ordinaire suivant :

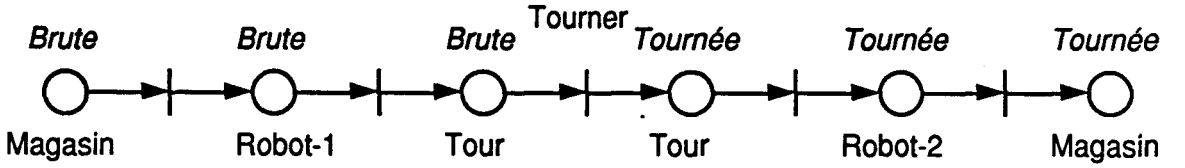


Figure 59

qui est bien un graphe d'état, on ne modifie qu'une seule des deux composantes du vecteur d'état de l'objet à la fois (état d'avancement, localisation de la pièce).

La phase de génération des gammes opératoires consiste à assister et contrôler le concepteur dans cette démarche.

IV.2.3. La démarche assistée de génération

La phase de génération des gammes opératoires présente les caractéristiques suivantes :

- une même fonction peut être accomplie indifféremment par plusieurs moyens,

- la flexibilité de routage peut être importante (indéterminisme de choix des chemins),

- la gestion des variantes de la spécification des moyens de production est délicate,

- une remise en cause de la spécification matérielle doit être de portée limitée.

Ces constatations nous ont conduit à générer les gammes opératoires par étapes successives. Nous avons choisi d'associer une étape de génération à chaque niveau de décomposition arborescente du procédé. La première étape prend en compte le premier niveau de l'arborescence.

IV.2.3.1. Les phases de chaque étape de génération

L'étape associée au niveau i se déroule schématiquement selon l'algorithme suivant :

1
spécification contrôlée des lieux de transit et des lieux opératoires faisant partie du niveau i par le concepteur,
sur la base de la connaissance du modèle de l'étape antérieure i-1 associée au niveau i-1, en détaillant quel sous lieu du niveau i-1 est choisi comme lieu support au lieu i

2
expansion assistée du RdP par ajout d'états autorisés afin d'exclure la simultanéité d'un transfert entre deux lieux du niveau i
et d'une opération de transformation

3
analyse du comportement, performances,
retour en **1** si il existe d'autres niveaux.

Nous allons présenter la démarche sur la base de notre exemple d'illustration afin d'en faciliter la compréhension. Le modèle arborescent du

procédé a été étudié au chapitre III. Il s'agit d'un arbre de profondeur 8, dont la racine est le support de toutes les machines, et les feuilles sont soit les emplacements de stockage élémentaires (préhenseurs), soit les effecteurs spécialisés (pistolet à graisse). Du point de vue des pièces et donc de la production, seules les feuilles sont effectivement gérées. En effet, un emplacement de stockage est systématiquement alloué à une pièce pour un transfert.

Pour schématiser, nous pouvons considérer que la PC, qui gère les pièces élémentaires, ne voit du procédé que les feuilles de l'arbre, alors que les autres éléments commandés sont gérés par le module Interface. Chaque niveau hiérarchique associé à la description des moyens opératoires permet ainsi d'abstraire la commande associée en une commande aveugle, laissant ainsi le soin de caractériser la commande fine relevant des niveaux inférieurs à une Interface de transformation des commandes aveugles en commandes fines.

Nous allons présenter en détail les trois premières étapes. Les autres étapes reviennent à une interprétation itérative de la démarche proposée.

IV.2.3.2. La première étape

Au cours de la première étape, nous assimilons le procédé au premier niveau de l'arbre. Les emplacements de stockage élémentaires sont alors artificiellement raccordés au deuxième niveau de l'arbre. Ainsi, le convoyeur est considéré comme la collection de tous les emplacements de stockage qui le composent :

Modèle de profondeur 5

Niveau 0	1	2	3	4
Cellule	Convoyeur	Rail-1	ES1	Ø
			.....	
			ES5	Ø
		Rail-2	ES6	Ø
			ES7	Ø
		Aiguillage1	ES8	Ø
		Aiguillage2	ES9	Ø

Modèle de profondeur 4

Niveau 0	1	2	3
Cellule	Convoyeur	ES1	Ø
		.....	
		ES5	Ø
		ES6	Ø
		ES7	Ø
		ES8	Ø
		ES9	Ø

Figure 60

L'abstraction réalisée du modèle sur le modèle 4 consiste à ne prendre en considération que le niveau 1 convoyeur. Les tronçons sont ainsi virtuellement omis. D'un autre point de vue, on peut considérer qu'il existe un convoyeur composé de 9 emplacements de stockage élémentaires. Nous disposons ainsi d'une vision macroscopique du convoyeur.

Le modèle de niveau 1 de la cellule consiste à ne retenir que les identificateurs des unités élémentaires du système retenu en leur attribuant les lieux opératoires qui les caractérisent.

Cellule	Stockage	ES1
		ESn
	Convoyeur	ES1
		ESm
	Robot-1	ES
	Robot-2	ES1
		ES2
	Robot Graissage	X
	Tour	ES
	Machine Ass	ES1
		ES2
	Taraudeuse	ES

Figure 61

Les relations d'accessibilité entre lieux de stockage sont déduites du schéma complet des relations :

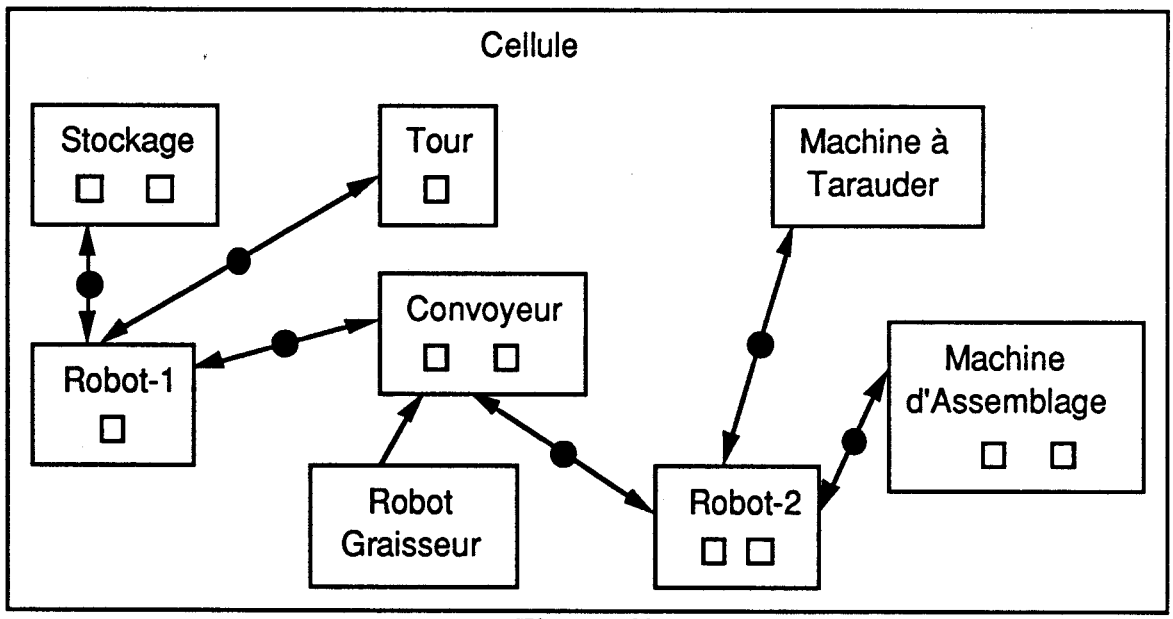


Figure 62

Ce schéma traduit les relations d'accessibilité directes. Par transitivité, il est bien sûr possible de définir d'autres relations d'accessibilité indirectes. Elles seraient redondantes et d'intérêt limité.

Sur la base de ces relations et des gammes logiques issues de la phase 1 de la démarche de conception, le concepteur spécifie, pour chaque état d'avancement, sur quel lieu doit se trouver l'objet au cours de son évolution dans la gamme.

Cette spécification est contrôlée à deux titres :

- pour vérifier qu'une fonction de traitement peut effectivement être menée à bien sur ce lieu (cohérence lieu/fonction),

- pour vérifier qu'entre deux lieux consécutifs, il existe bien au moins un chemin dans le schéma des relations d'accessibilité (cohérence des chemins).

La spécification du concepteur conduit, dans notre cas au schéma enrichi suivant. A chaque état de la gamme on associe un lieu opératoire.

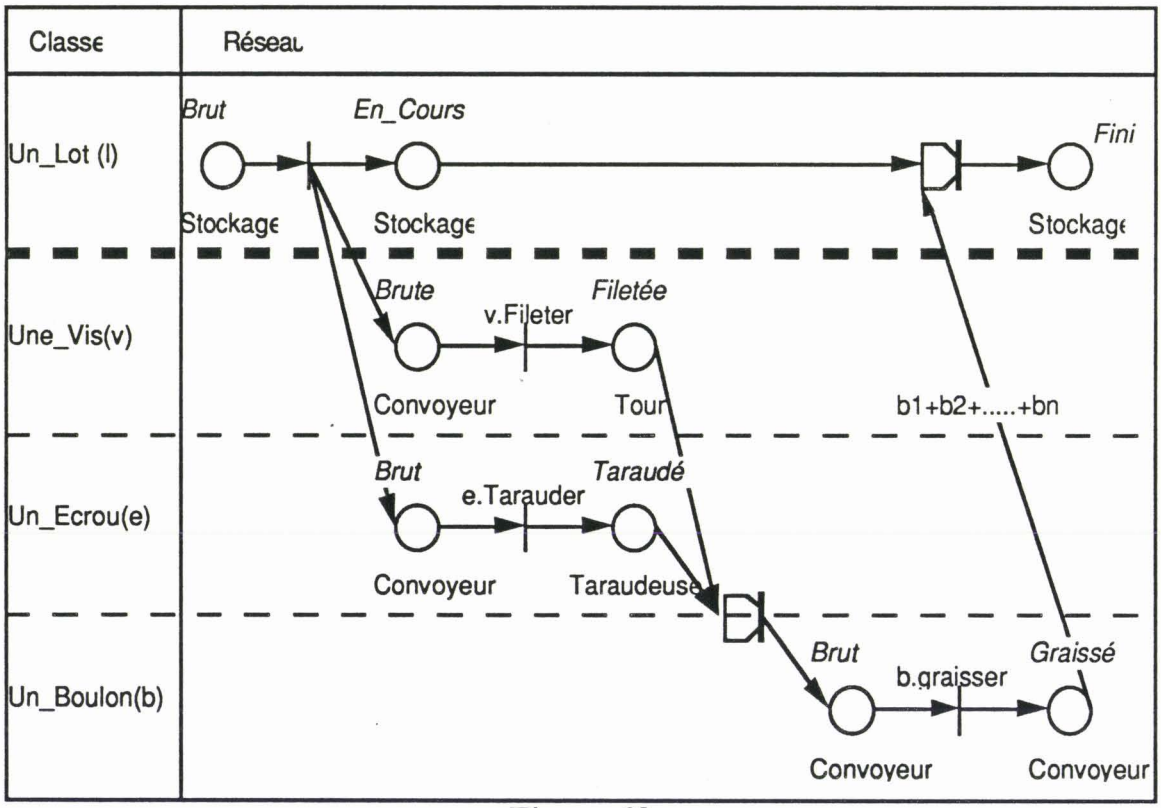


Figure 63

Cette spécification est valide. En effet, tous les lieux sont successivement accessibles, directement ou indirectement.

La phase de spécification contrôlée est donc terminée pour l'étape 1.

La deuxième phase de l'étape 1 consiste à contraindre le modèle afin de garantir la non simultanéité d'un transfert et d'une opération. Pour cela il suffit d'ajouter des places au réseau pour décomposer les opérations et/ou transferts simultanés éventuels. Tous les états ajoutés doivent être mutuellement accessibles. Il est clair que l'on voit ici apparaître toutes les alternatives de trajets des pièces entre les lieux de stockage. De fait, il est

possible de restreindre les possibilités de routage par rapport au potentiel du moyen de transport. Ce choix est bien sûr laissé à l'initiative du concepteur, ce dernier peut en particulier chercher à conserver tout le potentiel de flexibilité existant.

A titre d'illustration, nous allons détailler cette phase d'expansion au cas du désassemblage fictif du lot.

Lors du désassemblage fictif du lot en vis et écrous, aucune action n'est exécutée. Les pièces brutes sont donc initialement dans la zone de stockage, tout comme le lot lui-même. Le premier lieu de la vis est donc dans tous les cas le même que le lieu de l'assemblage dont elle est issue.

Nous obtenons donc :

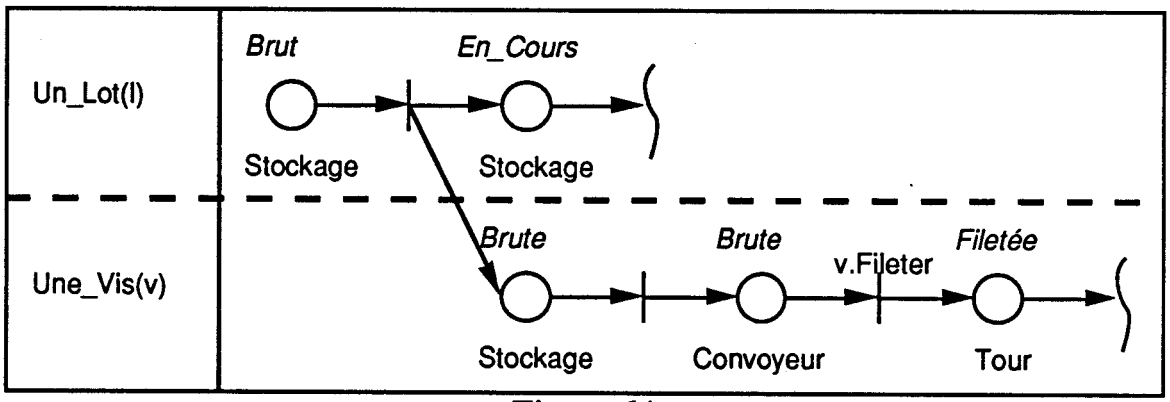


Figure 64

Par transitivité, le stockage est en relation d'accessibilité indirecte avec le convoyeur donc cette spécification est valide. Il faut cependant transiter par le Robot-1 pour atteindre le convoyeur (Fig 62).

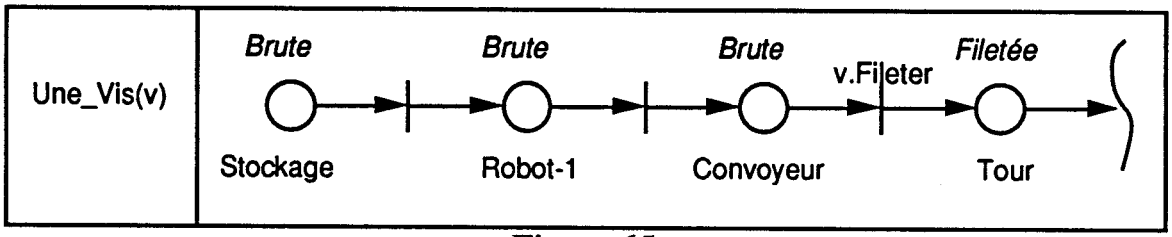


Figure 65

Puis à nouveau transiter par Robot-1 pour atteindre le Tour :

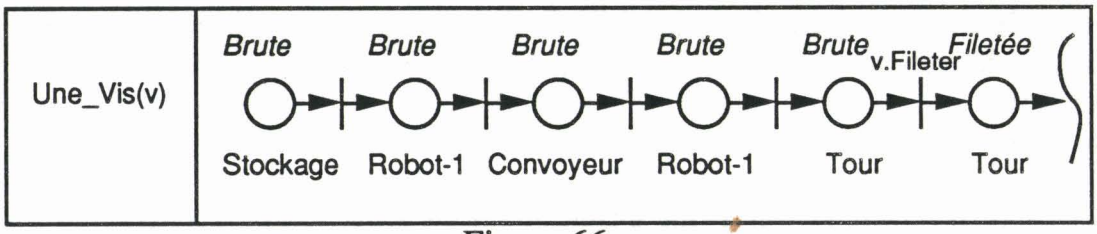


Figure 66

D'après le schéma des relations d'accessibilité, il est aussi possible d'atteindre le Tour directement. Nous obtenons donc une alternative dans la gamme partielle permettant de passer de l'état (Brute, Robot-1) à l'état (Brute, Tour) :

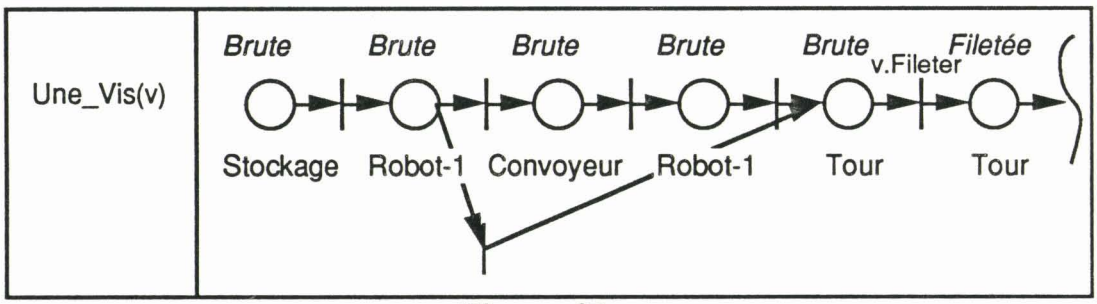


Figure 67

Nous procédons de la même manière pour la vis et pour l'écrou.

Considérons maintenant l'opération d'assemblage. On suppose qu'il s'agit d'un assemblage pur, c'est à dire que les deux pièces peuvent arriver indépendamment sur la zone d'assemblage. Dès qu'elles sont simultanément présentes, alors l'opération est déclenchée. Il s'agit d'une synchronisation par rendez-vous.

La portion de gamme opératoire est donc la suivante :

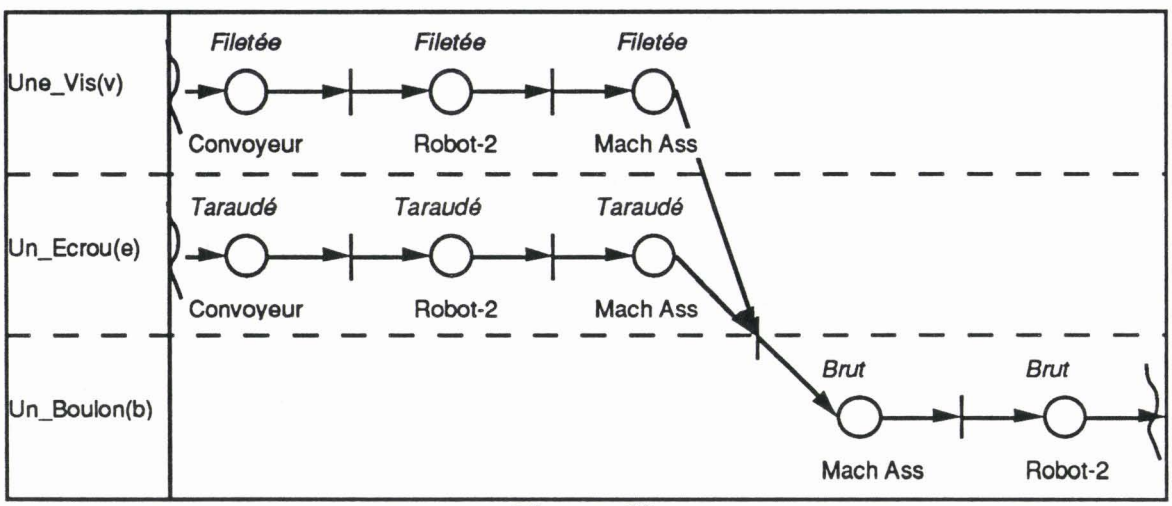


Figure 68

Le boulon est nécessairement sur le même lieu que les composants dont il est issu.

Prenons maintenant le cas d'un assemblage où un des composants (A) sert de base à l'autre (B) avant de réaliser l'assemblage effectif. Dans ce cas, le composant A doit se trouver sur la machine d'assemblage avant que le transfert de B ne commence. La gamme opératoire est donc la suivante :

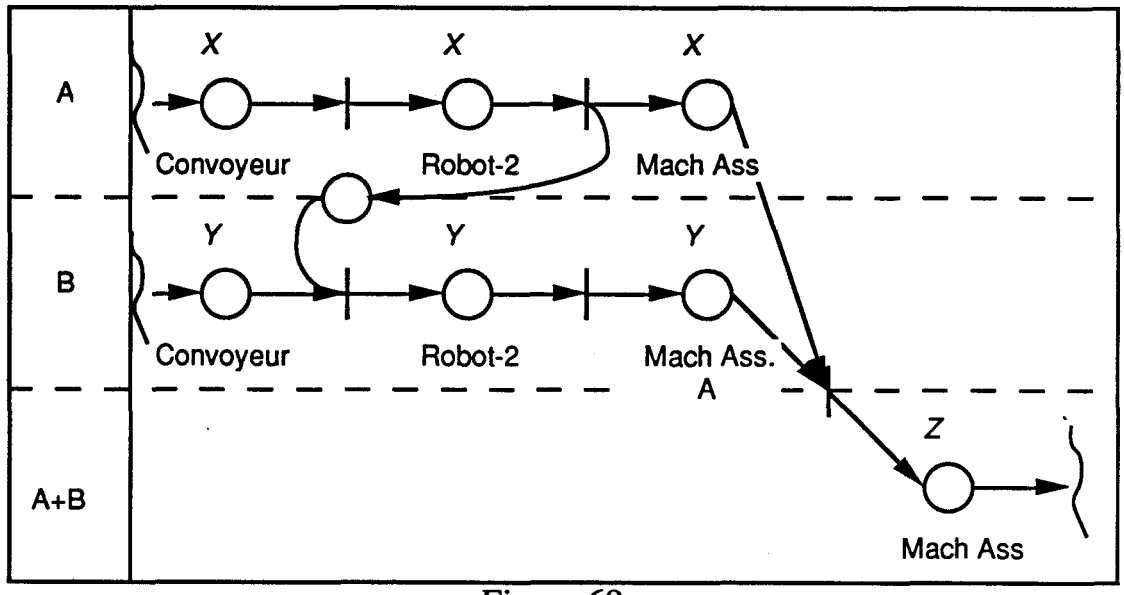


Figure 69

La pièce A est donc vue d'abord comme une pièce à traiter (gamme opératoire), et aussi comme un lieu de stockage (pour contenir B). La pièce A est le contenu d'un des emplacements de la Machine d'assemblage. Cependant A sert aussi de support pour la pièce B, et constitue ainsi un lieu physique opératoire de la machine d'assemblage. C'est la raison pour laquelle on notera ce lieu Mach Ass.A (notation pointée usuelle).

Nous devons, pour ce cas, introduire une place de synchronisation afin de garantir le positionnement préliminaire du support A.

Ces deux cas d'assemblage sont issus de la même gamme logique. La fonction d'assemblage est donc réalisée selon deux modes différents selon qu'il s'agit d'un assemblage pur et d'un assemblage avec contrainte d'antériorité.

De manière générale, dans le cas d'un assemblage :

- pour un assemblage pur, toutes les pièces ont le même support (le contenant de leurs emplacements de stockage), le produit également,

- pour un assemblage avec antériorité d'une des pièces, alors cette

pièce devient le support des autres, et elle ne se déplace pas. Le support du produit fini est alors le support de la pièce qui a servi à loger les autres (A est sur Machine_Assemblage, A+B aussi).

Ces deux règles permettent de valider la spécification des lieux de stockage et des lieux opératoires pour chaque pièce.

Les désassemblages obéissent à des règles similaires.

Le cas du lot est particulier puisqu'il n'a pas de support du fait que les deux pièces qui le composent évoluent séparément. Nous considérons qu'il reste sur la zone de stockage.

La gamme opératoire résultant de l'étape d'expansion de la gamme logique et des moyens de production associés est donc la suivante :

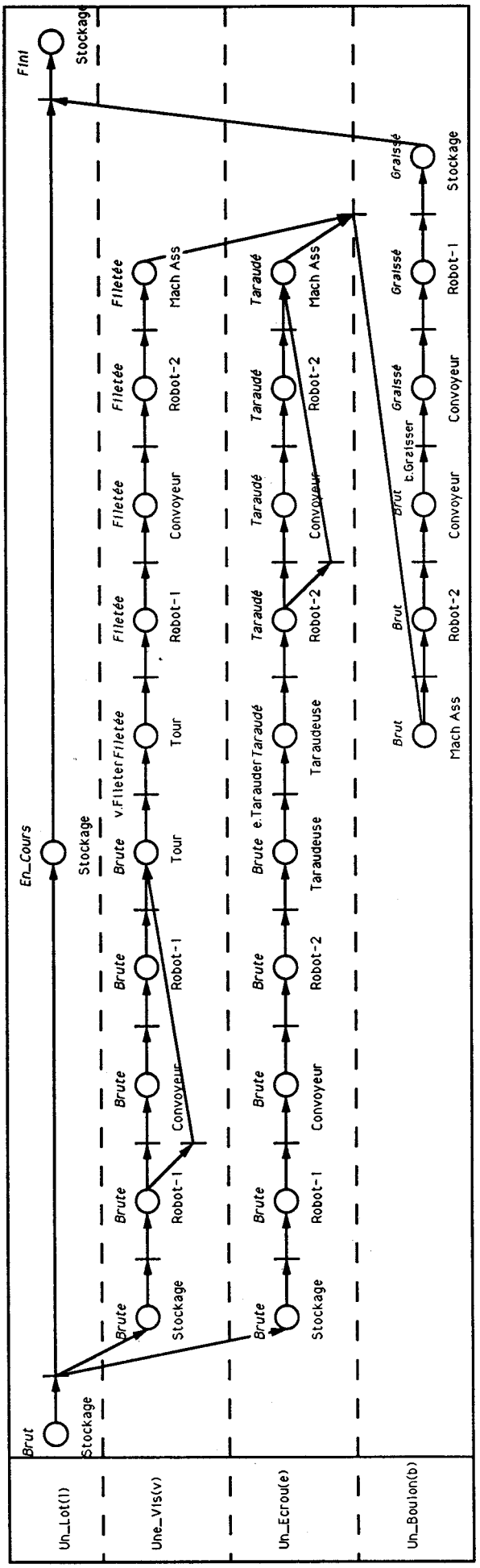

Un_Lot(l)
Une_Vis(v)
Un_Ecrou(e)
Un_Boulon(b)
Brut
Stockage
En_Cours
Fini
Brute
Robot-1
Robot-2
Convoyeur
Tour
v.Fileter
Filetée
e.Tarauder
Taraudeuse
Taraudé
Mach Ass
t.Graisser
Graissé

Figure 70

IV.2.3.3. La deuxième étape

La gamme opératoire de la première étape est enrichie et expansée en prenant en compte le deuxième niveau de la description du procédé. Le procédé est assimilé à :

Cellule	Stockage	ES1	
		ESn	
	Convoyeur	Rail-1	ES1
			ESm
		Rail-j	ES1
			ESm
		Aig-1	ES
		Aig-2	ES
	Robot-1	Tronc	ES
	Robot-2	Tronc	ES1
			ES2
	Robot Graissage	Bras	X
	Tour	Mandrin	ES
	Machine Ass	ES1	
		ES2	
	Taraudeuse	ES	

Figure 71

Le schéma des relations d'accessibilité est décrit Fig 72. Il donne des précisions supplémentaires concernant l'accessibilité des moyens intermédiaires associés à la gestion des lieux physiques.

On apprend ainsi sur ce schéma qu'un convoyeur est constitué de différents tronçons (rails, aiguillages), et que l'accessibilité initiale au convoyeur pour les clients potentiels est en fait limitée à certains tronçons bien spécifiques.

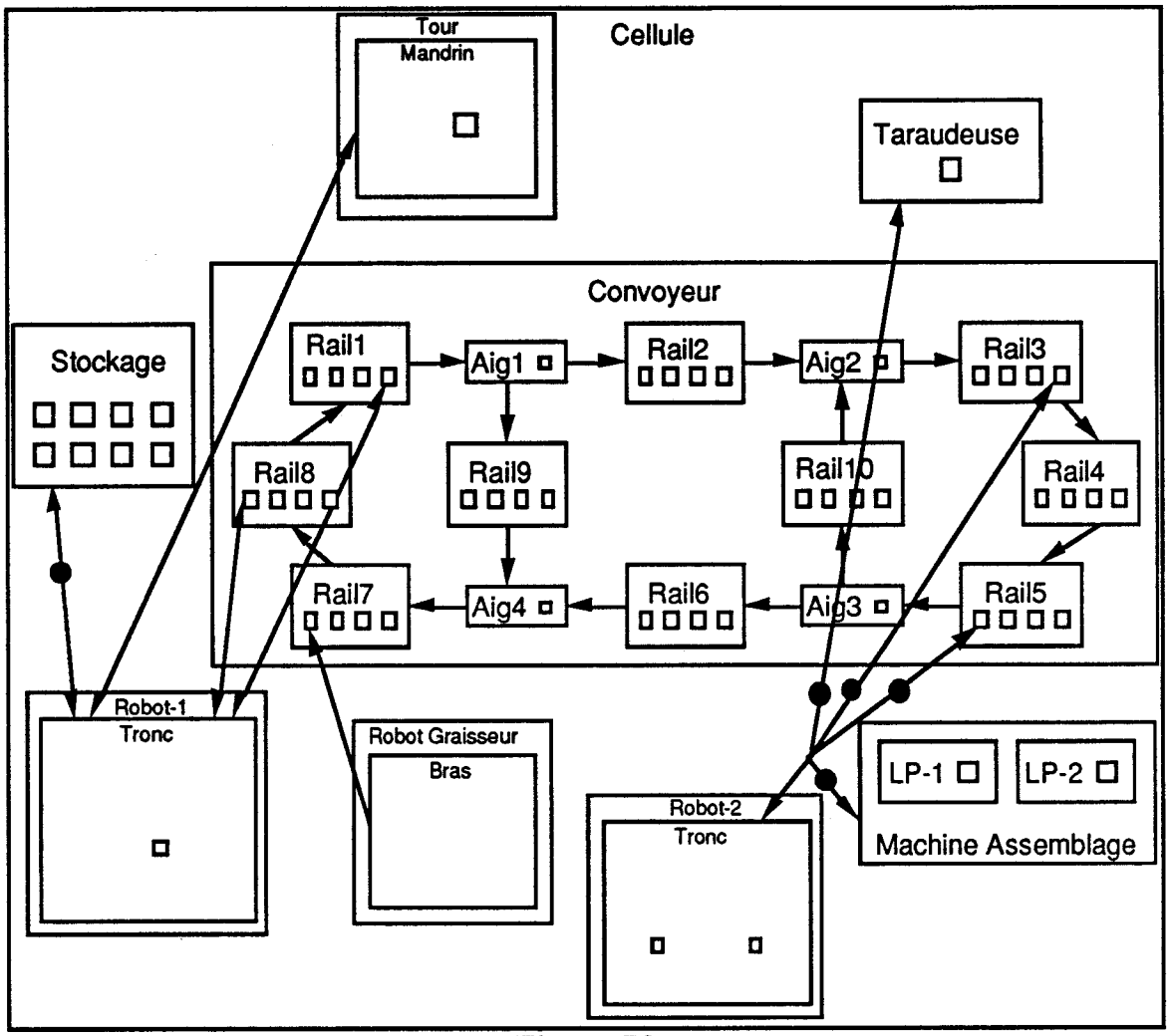


Figure 72

D'un point de vue fonctionnel, le rôle du convoyeur est de transférer les pièces des points d'entrée aux points de sortie. Sur cet exemple les rails 1, 3, 5 et 7 sont 4 points d'entrée/sortie de pièces. Mais du point de vue des pièces, peu importe le trajet effectif à l'intérieur du convoyeur. D'après la configuration des relations, et par transitivité, tout point d'entrée peut mener à tout point de sortie.

Donc du point de vue des pièces, le convoyeur est vu comme un moyen de transport complexe qui permet d'atteindre tout point de sortie depuis tout point d'entrée. Il est donc permis, du point de vue de la pièce, d'abstraire le convoyeur en ne prenant en considération que les points d'entrée (respectivement de sortie), et les chemins connectant les entrées aux sorties.

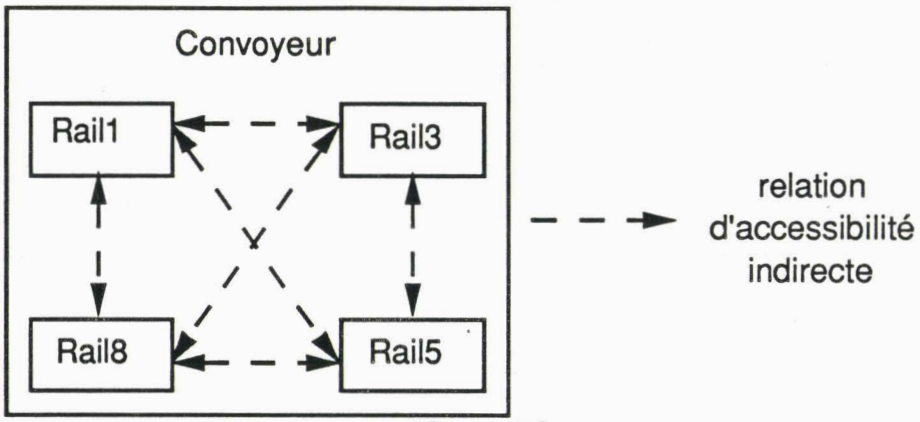


Figure 73

On ne voit ainsi que les points d'entrée/sortie, et les trajets potentiellement réalisables.

Comme l'étape 1, l'étape 2 commence par la spécification des lieux sous jacents par lesquels transitent les pièces.

Dans le cas de la Vis brute, elle quitte le Robot-1 pour atteindre le convoyeur plus précisément au niveau du lieu Rail-1. Ce sous lieu est noté Convoyeur.Rail-1. La spécification doit bien sûr être telle que deux lieux successifs d'une pièce soient atteignables par au moins un chemin dans le graphe issu des relations d'accessibilité.

Nous obtenons dans notre cas le réseau suivant :

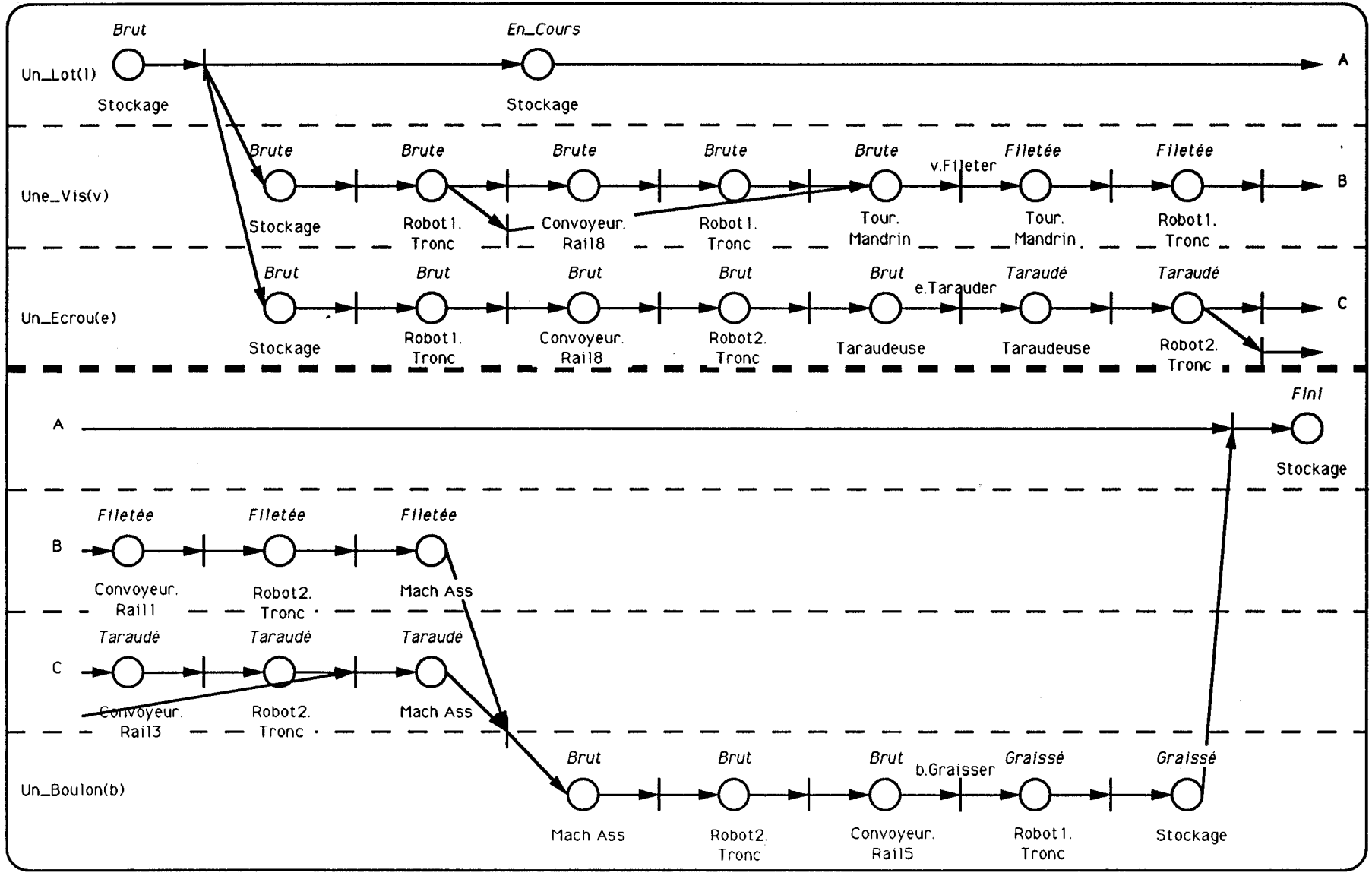

Brut
En_Cours
Un_Lot(l)
A
Stockage
Stockage
Brute
Brute
Brute
Brute
Brute
Filetée
Filetée
v.Fileter
Une_Vis(v)
B
Stockage
Robot1.
Tronc
Convoyeur.
Rail8
Robot1.
Tronc
Tour.
Mandrin
Tour.
Mandrin
Robot1.
Tronc
Brut
Brut
Brut
Brut
Brut
Taraudé
Taraudé
e.Tarauder
Un_Ecrou(e)
C
Stockage
Robot1.
Tronc
Convoyeur.
Rail8
Robot2.
Tronc
Taraudeuse
Taraudeuse
Robot2.
Tronc
Fini
A
Stockage
Filetée
Filetée
Filetée
B
Convoyeur.
Rail1
Robot2.
Tronc
Mach Ass
Taraudé
Taraudé
Taraudé
C
Convoyeur.
Rail3
Robot2.
Tronc
Mach Ass
Brut
Brut
Brut
Graissé
Graissé
b.Graisser
Un_Boulon(b)
Mach Ass
Robot2.
Tronc
Convoyeur.
Rail5
Robot1.
Tronc
Stockage

Figure 74

De nouveau, nous devons ajouter des états supplémentaires afin que deux lieux successivement atteints par une pièce soient directement accessibles.

La gamme opératoire de la deuxième étape est donc la suivante :

Brut
En_Cours
Ur_Lot(1)
Stockage
Stockage
A
Brute
Brute
Brute
Brute
Brute
Brute
v.Fileter
Filetée
Filetée
Ure_Vis(v)
Stockage
Robot1.
Trorc
Convoyeur.
Rail8
Convoyeur.
Rail1
Robot1.
Trorc
Tour.
Marcrir
Tour.
Marcrir
Robot1.
Trorc
B
Brut
Brut
Brut
Brut
Brut
Brut
e.Tarauder
Taraudé
Taraudé
Ur_Ecrou(e)
Stockage
Robot1.
Trorc
Convoyeur.
Rail8
Convoyeur.
Rail3
Robot2.
Trorc
Tarauceuse
Tarauceuse
Robot2.
Trorc
C
Fini
A
Stockage
Filetée
Filetée
Filetée
Filetée
B
Convoyeur.
Rail1
Convoyeur.
Rail5
Robot2.
Trorc
Mach Ass
Taraudé
Taraudé
Taraudé
Taraudé
C
Convoyeur.
Rail3
Convoyeur.
Rail5
Robot2.
Trorc
Mach Ass
Brut
Brut
Brut
Brut
b.Graisser
Graissé
Graissé
Graissé
Graissé
Ur_Bouler(b)
Mach Ass
Robot2.
Trorc
Convoyeur.
Rail5
Convoyeur.
Rail7
Convoyeur.
Rail7
Convoyeur.
Rail8
Robot1.
Trorc
Stockage

Figure 75

Ce principe est réitéré selon la même procédure pour recouvrir totalement la décomposition structurelle arborescente du procédé.

IV.2.3.4. La troisième étape

Considérons à nouveau le convoyeur. Les pièces peuvent se déplacer sur des palettes simples à un seul emplacement, ou sur des palettes pouvant accueillir plusieurs pièces indépendantes. Cela permet d'augmenter les flux de pièces tout en limitant le nombre de palettes en circulation.

Conformément à l'esprit de la démarche d'analyse du procédé que nous avons étudiée au chapitre III, la palette est considérée comme un support à plusieurs emplacements, qui se déplace sur l'ensemble de l'architecture du convoyeur. Du strict point de vue de la pièce transportée, le chargement sur la palette du convoyeur consiste dans un premier lieu à positionner la palette sur la zone de chargement (le Rail-1 par exemple), puis dans un second lieu à réaliser effectivement l'opération de chargement.

Cependant de multiples emplacements peuvent être chargés et déchargés simultanément lorsque la palette se trouve sur une zone de stationnement.

La palette joue donc un rôle identique à celui de la cage de l'ascenseur que nous avons présenté au chapitre II.

Ainsi, le trajet effectif de la palette dans l'ensemble des tronçons du convoyeur est transparent pour la Partie Commande, il est géré par le module INTERFACE. Pour la PC, la fonction de déplacement de la palette est un service qui est rendu par l'INTERFACE.

Le Procédé, dans la troisième étape, est assimilé à l'arborescence suivante :

Cellule	Stockage	ES1		
		ESn		
	Convoyeur	Rail-1	Palette-1	ES1
				ES4
			Palette-2	ES1
				ES2
			ES3	
			ES4	
		Rail-j	ES1	
			ESm	
		Aig-1	ES	
		Aig-2	ES	
	Robot-1	Tronc	Bras	ES
	Robot-2	Tronc	Pince-1	ES
			Pince-2	ES
	Robot Graissage	Bras	Pince	X
	Tour	Mandrin	Machoires	ES
	Machine Ass	ES1		
		ES2		
	Taraudeuse	ES		

Figure 76

Voyons, sur l'exemple du convoyeur, quels pourraient être les modèles manipulés par l'INTERFACE. La Partie Commande ne "voit" et ne gère du procédé que les emplacements de stockage pour les pièces.

Nous avons établi que la PC ne gère que l'allocation des feuilles de la description arborescente aux pièces produites. La PC n'a pas ici connaissance des caractéristiques de convoyage multiple de certaines palettes. Cette information n'est pas gérée par la PC mais concerne l'Interface. Les autres Objets Physiques Commandables sont gérés par l'INTERFACE. En effet, ils accomplissent un service (le positionnement sur leur support) vis à vis des clients qui sont les feuilles de l'arbre.

Ainsi, le schéma de contrôle du procédé peut se résumer ainsi :

- la PC gère l'intégralité des allocations des ES par lesquels transitent les pièces (feuilles de l'arbre),

- l'INTERFACE gère les autres Objets Physiques Commandables.

Il est clair que l'importance relative de ces deux systèmes de contrôle

dépend :

- de la complexité (profondeur) des moyens,

- du degré d'achèvement des contrôleurs locaux (Commandes Numériques).

Dans le cas du convoyeur, l'INTERFACE pourrait comporter les images suivantes :

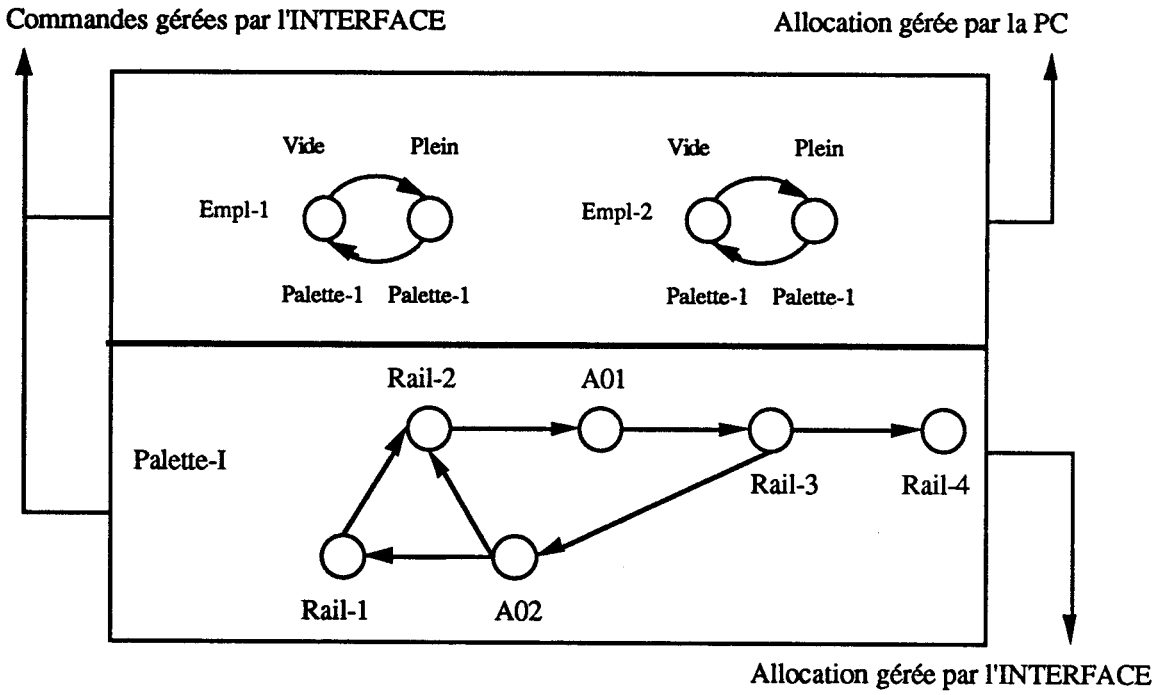


Figure 77

Cette figure représente schématiquement les états des ES de la Palette-1, ainsi que les états de l'objet mobile Palette-1. Toutes les commandes à destination des ES de la Palette-1 et de la palette elle-même sont gérées par l'Interface. La palette transite par les ES des rails et des aiguillages. L'allocation de ces ES aux palettes est gérée localement au niveau de l'Interface, indépendamment de l'allocation des ES des palettes.

La fonction de chargement d'une pièce sur une palette est réalisée en deux parties :

1°. positionnement de la palette à l'adresse de l'échange dans l'espace,

2°. chargement effectif de l'emplacement de la palette.

Les commandes transmises par la PC adressent chacun des Objets Physiques Commandables. Le chargement de l'emplacement ESi de la

Palette-1 sur le Rail-1 se décompose de la manière suivante :

1°. Contrôler la position de la palette :
(positionner Palette-1 sur Rail-1,
ESi non commandé)

2°. Charger le lieu physique :
(Palette-1 non commandé,
charger ESi)

Les commandes transmises adressent par la structure d'arborescence des moyens tous les Objets Physiques Commandables concernés par la feuille qui est allouée à la pièce manipulée. Il faut donc évaluer le contrôle des parents de la feuille dans la description arborescente des moyens physiques.

Les commandes ne précisent que l'état final à atteindre pour chaque Objet Physique. Cet état peut appartenir indifféremment :

- à un espace continu : position angulaire du tronc d'un robot sur sa base,

- à un espace discret (ou discrétisé) : état Vide ou Plein.

C'est à l'INTERFACE d'assurer l'asservissement en position pour atteindre l'état final dans le cas d'un état continu (position de la cage de l'ascenseur par exemple).

De plus, dans la mesure où l'INTERFACE commande des objets qui se déplacent sur leur support (palette sur son rail, d'un rail à l'autre), il est nécessaire de gérer l'allocation des ressources partagées au niveau local (capacité des rails à contenir un nombre fini de palettes). Cette allocation n'interfère pas avec l'allocation des feuilles de l'arborescence puisqu'elles sont gérées exclusivement par la PC.

De ce fait, il est possible d'envisager une optimisation à court terme de la trajectoire des palettes en fonction des requêtes qu'elles doivent potentiellement satisfaire (une palette peut recevoir l'ordre de se diriger simultanément vers plusieurs destinations puisqu'elle peut contenir plusieurs pièces) en fonction de critères. Ces fonctions de choix obéissent à la stratégie de fonctionnement du procédé qui est imposée par le Niveau Hiérarchique (voir Chapitre II).

Nous pouvons constater également que le schéma de contrôle du cheminement des palettes apparaît explicitement dans CASPAIM-1. En effet, le prégraphe décrit l'ensemble des lieux par lesquels transitent les palettes ainsi que les opérations qui doivent être appliquées aux pièces.

Nous considérons dans CASPAIM-2 que les palettes ont un fonctionnement propre, qu'elles sont gérées par un système de commande spécifique, qui doit en particulier contrôler l'accès aux différents tronçons physiques. L'ensemble des palettes est assujetti aux ordres qui émanent de la PC. C'est un moyen de transport complexe à plusieurs points d'entrée/sortie et dont le fonctionnement peut être paramétré en vue d'une optimisation.

CASPAIM-2 est donc une extension logique de la démarche CASPAIM-1.

Revenons aux gammes opératoires de la troisième étape. L'architecture du convoyeur a déjà été détaillée. Le Robot-2 possède deux préhenseurs. Le chargement d'une pièce peut donc être réalisé indifféremment par les deux pinces.

La gamme opératoire issue de la troisième étape est la suivante :

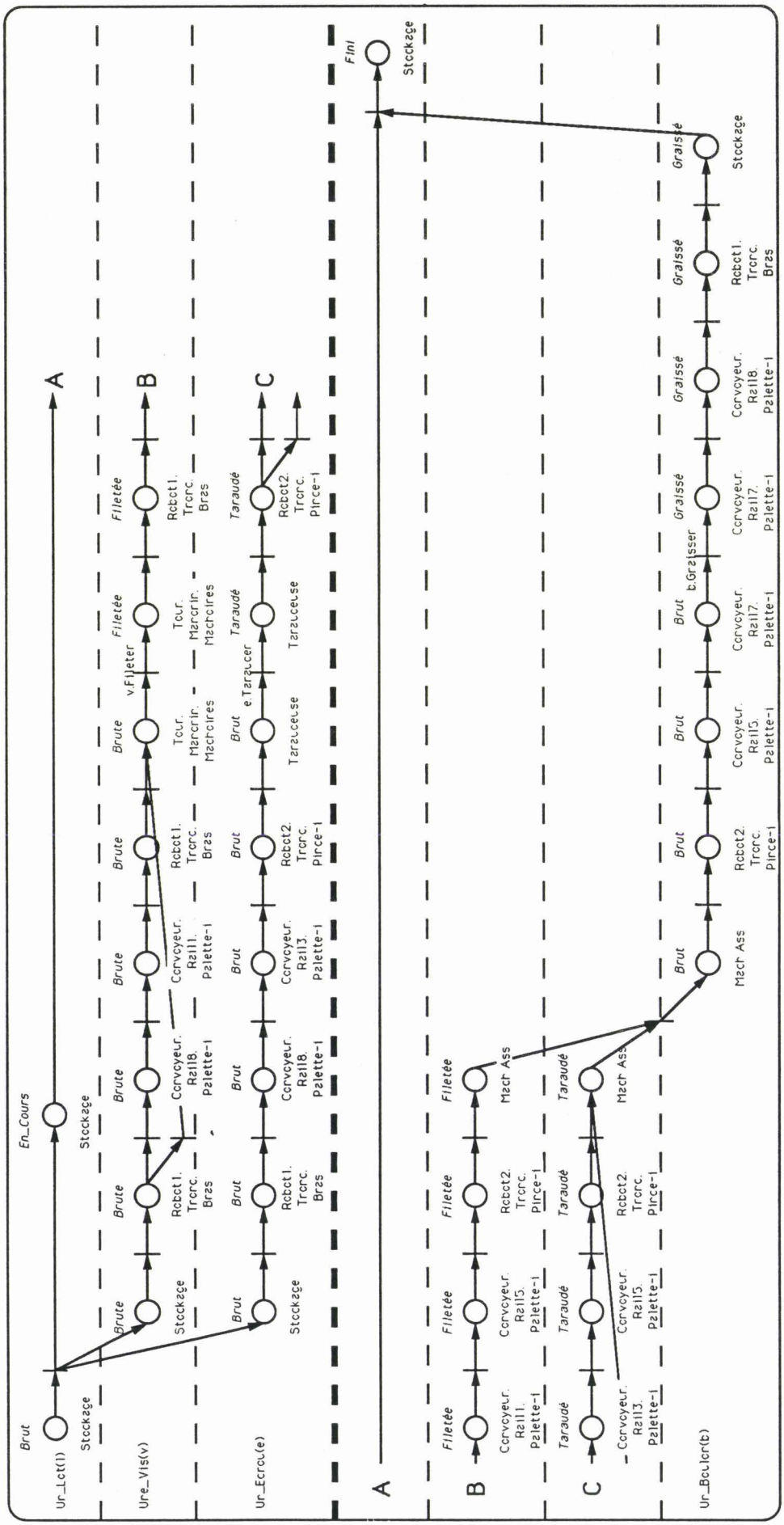

Ur_Lot(l)
Ure_Vis(v)
Ur_Ecrou(e)
Ur_Bouler(b)
Brut
Stockage
En_Cours
Fini
A
B
C
Brute
Filetée
Taraudé
Graissé
Corvoyeur. Rail1. Palette-1
Corvoyeur. Rail3. Palette-1
Corvoyeur. Rail5. Palette-1
Corvoyeur. Rail7. Palette-1
Corvoyeur. Rail8. Palette-1
Robot1. Trorc. Bras
Robot2. Trorc. Pince-1
Tour. Marcrin. Machoires
v.Fileter
e.Tarauder
Tarauceuse
b.Graisser
Mach Ass

Figure 78

IV.2.3.5. Passage d'une étape à ses sous-étapes

Toutes les étapes se déroulent selon le schéma du §IV.2.3.1. Lors du passage d'une étape à une sous-étape, on prend en considération un niveau d'affinement supplémentaire dans la description du procédé, lorsque plusieurs variantes sont envisageables pour une même étape de départ. La démarche proposée permet donc de spécifier plusieurs variantes d'architectures matérielles, et de les comparer sur le plan des propriétés et des performances.

Les variantes sont donc affinées à chaque niveau de spécification selon le schéma suivant :

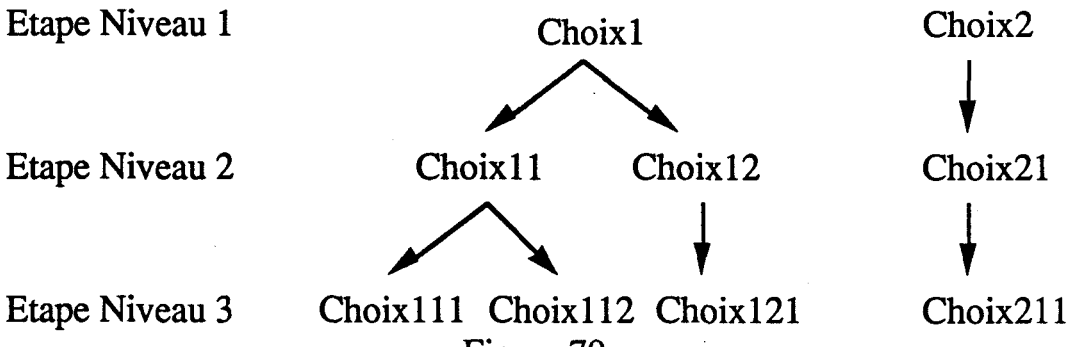


Figure 79

Pour ce Procédé de profondeur 3, les variantes terminales peuvent être comparées. Puisque les choix de niveau 2 ont été validés ils ne seront pas remis en question au niveau 3. Cette propriété constitue un énorme avantage par rapport aux démarches de conception antérieures relevant de CASPAIM-1.

La démarche de génération assistée des gammes opératoires par étapes successives est progressive, elle permet en outre une validation du résultat de chaque étape qui ne sera pas remis en cause lors de l'affinement.

IV.3. Validation : terminaison propre et quasi vivacité

IV.3.1. Introduction.

Dans cette section nous exploitons l'outil de modélisation RdP afin de vérifier les propriétés de terminaison propre et de quasi vivacité de la gamme logique dans un premier temps, puis de la gamme opératoire dans un second temps (§IV.3.6). Schématiquement, si ces propriétés sont vérifiées, alors l'état final du produit peut être atteint, quel que soit son processus de fabrication.

De ce fait, on est assuré que le produit sera effectivement fini (terminaison propre) d'une part, et qu'il peut l'être selon toutes les variantes que l'on souhaite (quasi vivacité) d'autre part.

Des méthodes de réduction de RdP peuvent être utilisées afin de vérifier les propriétés sur des réseaux de petite taille mais ayant conservé les propriétés des réseaux initiaux. Il nous semble cependant que les techniques présentées dans (BRA 83, BER 83, BER 85, EST 85, EST 87) s'appliquent à des réseaux ayant à priori une topologie quelconque. La méthodologie que nous avons développée exploite principalement des réseaux de type graphe d'état et graphe d'événements combinés entre eux. De ce fait, la vérification des propriétés est simplifiée de par la nature même des RdP générés.

Rappelons quelques définitions importantes (BRA 83).

Définition :

Un réseau normalisé R est un réseau possédant deux places distinguées Pd et Pf appelées respectivement place de début et place de fin, qui sont telles que : Pd ne possède aucune transition en entrée, et Pf aucune transition en sortie.

Un marquage initial Md de ce réseau est tel que seule la place Pd est marquée et contient une marque.

Un marquage dit terminal Mf est tel que seule la place Mf est marquée et contient une marque.

La majorité des gammes logiques que nous avons étudiées sont des réseaux normalisés, pour lesquels la place Pd et la place Pf correspondent respectivement à l'état initial et l'état final.

Le marquage terminal Mf traduit le fait que la pièce (ou le lot) est réalisée.

De manière informelle la terminaison propre d'un réseau normalisé se définit comme la possibilité d'atteindre le marquage terminal, depuis tout marquage accessible. En terme de gamme logique, cette propriété traduit la possibilité d'atteindre l'état final à partir de tout état atteignable depuis l'état initial.

La propriété de quasi vivacité (cf définition en annexe) de réseau n'est cependant pas assurée. Ce qui se traduit par le fait que l'on ne peut pas assurer que toutes les variantes de fabrication sont effectivement réalisables.

Définition :

Un réseau normalisé <R; Md> est dit à terminaison propre si pour tout marquage accessible M depuis Md, le marquage terminal Mf est atteignable.

Soit R=<P U {Pf,Pd}, T; Pré, Post > un réseau normalisé.

Le réseau bouclé associé à R, noté Rb, est obtenu en ajoutant la transition tb qui relie Pf et Pd.

Des deux propriétés suivantes (BRA 83) :

Si le réseau <Rb; Md> est borné
et si la transition tb est vivante
alors
<R; Md> est à terminaison propre.

et

Si de plus <Rb; Md> est vivant
alors
<R; Md> est quasi-vivant.

nous déduisons la propriété (1) :

Si <Rb; Md> est borné et vivant
alors
<R; Md> est à terminaison propre et quasi vivant

Dans le cas des gammes logiques, cela se traduit par la propriété suivante :

- toutes les variantes du processus de fabrication sont accessibles depuis l'état initial,

- depuis tout état atteint, il est possible d'atteindre l'état final.

Nous nous sommes ramené à la vérification de la finitude et de la vivacité du réseau bouclé issu de la gamme logique qui est supposée normalisée.

La propriété de Vivacité

Les définitions des termes utilisés ci-après sont rappelées en annexe.

Nous souhaitons utiliser le théorème de Commoner (BRA 83) dans deux contextes différents.

Soit G un graphe d'événements, et R son réseau associé. Pour le marquage M :

Le RdP <R; M> est vivant SSI tout circuit élémentaire contient au moins une marque.

Soit G une machine à états, et R son réseau associé. Pour le marquage M :

Le RdP <R; M> est vivant SSI G est fortement connexe et contient au moins une marque.

IV.3.2. Graphe d'état

Considérons le cas du graphe d'état GL traduisant la gamme logique d'une pièce. Il s'agit donc par hypothèse d'un réseau normalisé.

Appelons GLb le réseau bouclé obtenu à partir de GL. GLb a la forme suivante :

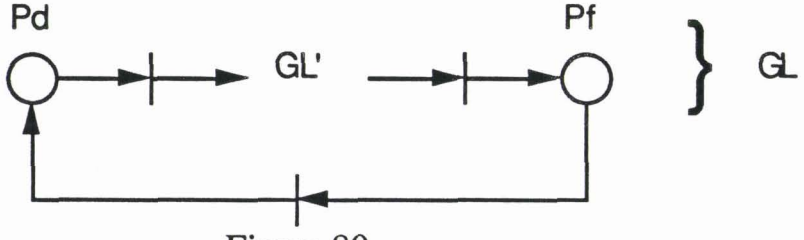


Figure 80

D'après la propriété (1),

Si <GLb; Md> est borné et vivant
alors
<GL; Md> est à terminaison propre et quasi-vivant

Puisque tout RdP qui est un graphe d'état est borné, il suffit de démontrer

que le réseau <GLb; Md> est vivant. D'après Commoner, GLb doit être fortement connexe et contenir au moins une marque.

Puisque le réseau est monomarqué, nous obtenons la proposition suivante (2) :

Si GLb est fortement connexe
alors
<GL; Md> est à terminaison propre et quasi vivant.

Lors de la spécification de la gamme logique, nous devons donc nous assurer de la forte connexité du réseau bouclé généré à partir de la gamme logique.

Dans la pratique, le graphe d'état construit et rebouclé est souvent fortement connexe.

Un contre-exemple concerne le traitement de pièces rebutées. En effet, l'état rebutée est un état puits autre que l'état final normal. La propriété de forte connexité n'est bien sûr pas vérifiée et le graphe d'état n'est pas vivant. L'interprétation en est qu'une fois atteint l'état rebutée, la pièce ne peut plus évoluer (état puits), et en particulier atteindre l'état final.

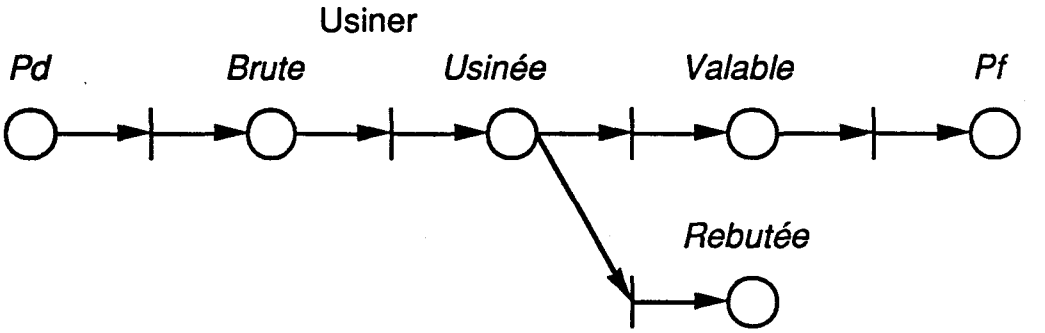


Figure 81

La propriété de forte connexité doit être validée lorsque la spécification de la gamme logique est laissée à l'initiative du concepteur.

Les gammes logiques suivantes conduisent à des réseaux bouclés fortement connexes :

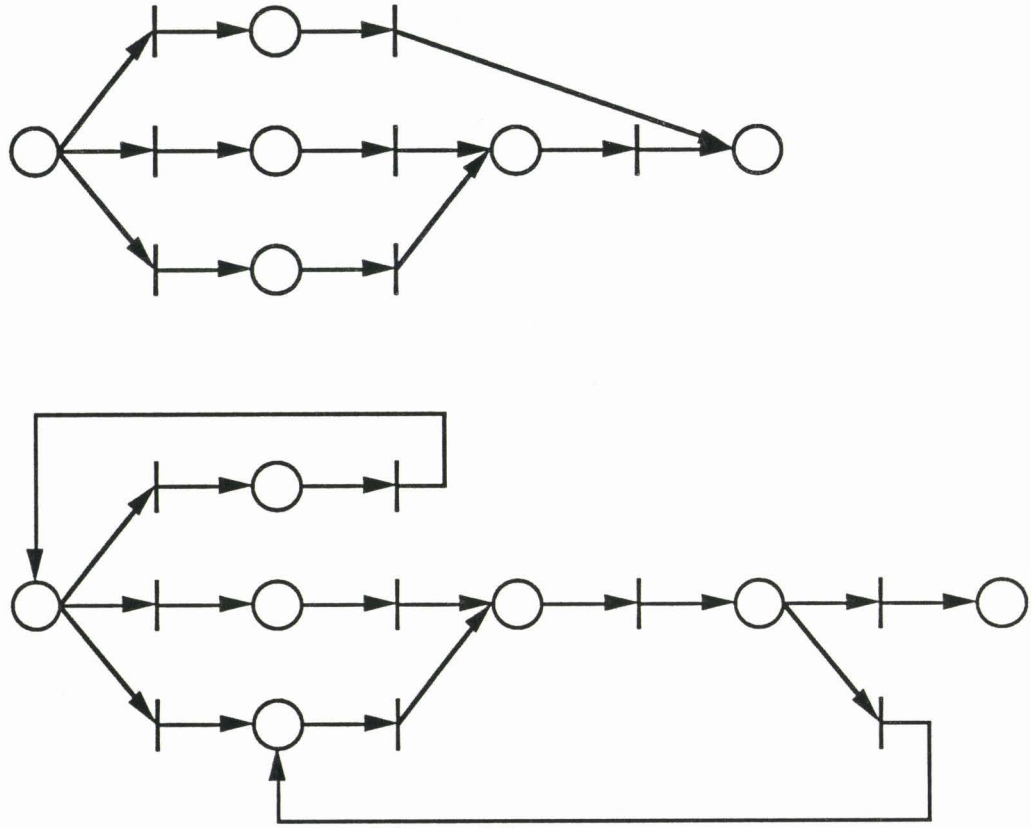


Figure 82

Dans le cas de gammes logiques comportant un grand nombre de variantes et de traitements de défaut (rebouclages), la propriété doit être vérifiée automatiquement au moyen d'algorithmes dédiés tel que l'algorithme de (TAR 72) présenté dans (GON 79). Cet algorithme réalise la recherche des points d'articulation d'un graphe orienté.

Cependant il s'est avéré que la propriété peut être vérifiée de façon simple sur les exemples que nous avons traités jusqu'à maintenant.

IV.3.3. Graphe d'événement

Considérons un graphe d'événement GL traduisant la gamme logique d'un lot de pièces ou d'un assemblage. Il est clair que la propriété de terminaison propre ne peut s'appliquer qu'aux traitements par lots (à plusieurs niveaux éventuellement), et aux désassemblages de n pièces suivis d'un assemblage de ces mêmes n pièces.

Le théorème de Commoner et la propriété (1), dans le cas d'un graphe d'événement nous donnent la proposition suivante (3) :

Si dans le GLb tout circuit élémentaire contient au moins une marque
et si <GLb; Md> est borné

alors
<GL; Md> est à terminaison propre et quasi vivant

Dans la pratique, nous sommes conduit à vérifier les prémisses de cette proposition sur chaque cas concret.

IV.3.4. Méthode de construction des gammes logiques garantissant la terminaison propre et la quasi vivacité

Nous avons cherché à définir un schéma de génération des gammes logiques qui permette, par construction, de garantir les propriétés de quasi vivacité et de terminaison propre.

Nous avons, pour cela, défini une grammaire de génération du RdP associé à une gamme logique.

La première partie de cette grammaire concerne les graphes d'état. La gamme logique d'une pièce isolée doit suivre la grammaire suivante :

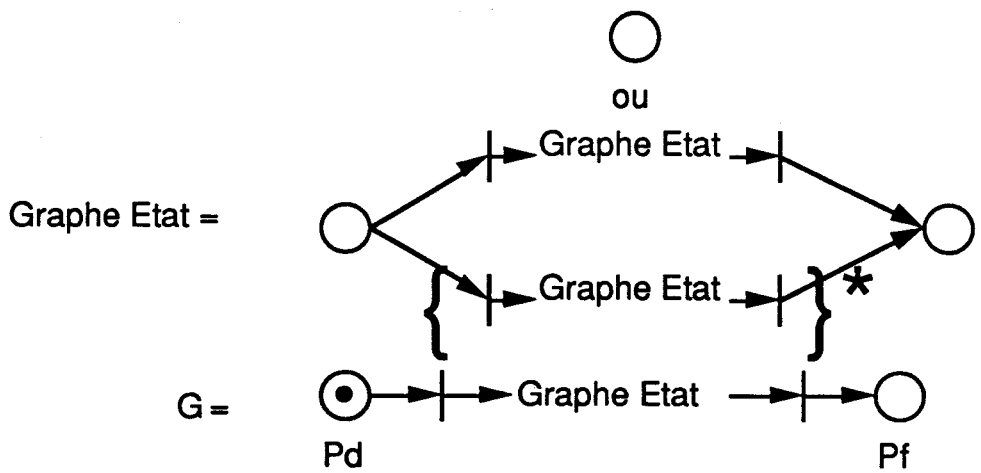


Figure 83

Il est aisé de vérifier que le RdP généré est effectivement un graphe d'état : toute transition a une place en amont et une place en aval.

Le réseau G est un réseau normalisé, dont le réseau bouclé est borné (graphe d'état), et fortement connexe par construction. Il est également monomarqué, donc tout graphe d'état construit sur cette grammaire est quasi vivant et à terminaison propre.

La deuxième partie de la grammaire concerne les graphes d'événements. Ils sont construits selon la grammaire suivante :

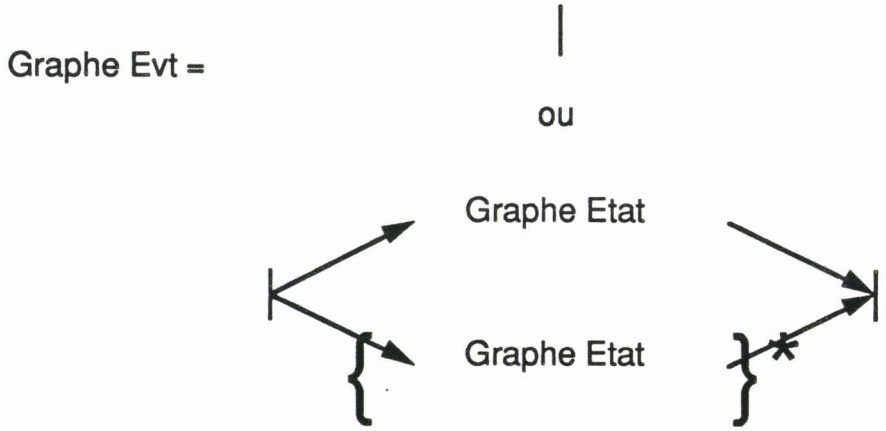


Figure 84

Chaque graphe d'état représente le traitement d'une pièce, et le graphe d'événement représente le traitement d'un lot unique, ou le désassemblage de n pièces suivi du réassemblage de ces mêmes pièces.

Le réseau bouclé issu de Graphe Evt est le suivant :

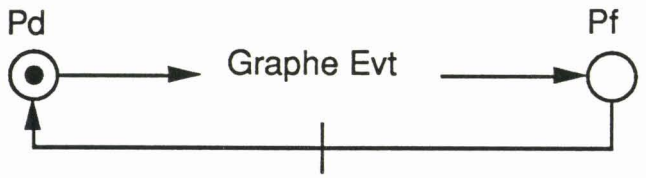


Figure 85

Du fait de la construction des graphes d'état, tous les circuits du réseau bouclé passent par la place Pd. En effet, les rebouclages dans le graphe d'état ne peuvent être générés par la grammaire. Donc tous les circuits élémentaires du réseau bouclé contiennent une marque. Il est donc vivant. Enfin, il est borné, puisque tous les graphes d'état le sont. Le Graphe Evt muni des places Pd et Pf est donc bien quasi vivant et à terminaison propre.

Cette grammaire permet donc la génération de gammes logiques ayant ces deux propriétés.

Il serait bien sûr intéressant de pouvoir exploiter la définition récursive des lots à l'aide d'une grammaire analogue.

Nous définissons une deuxième grammaire :

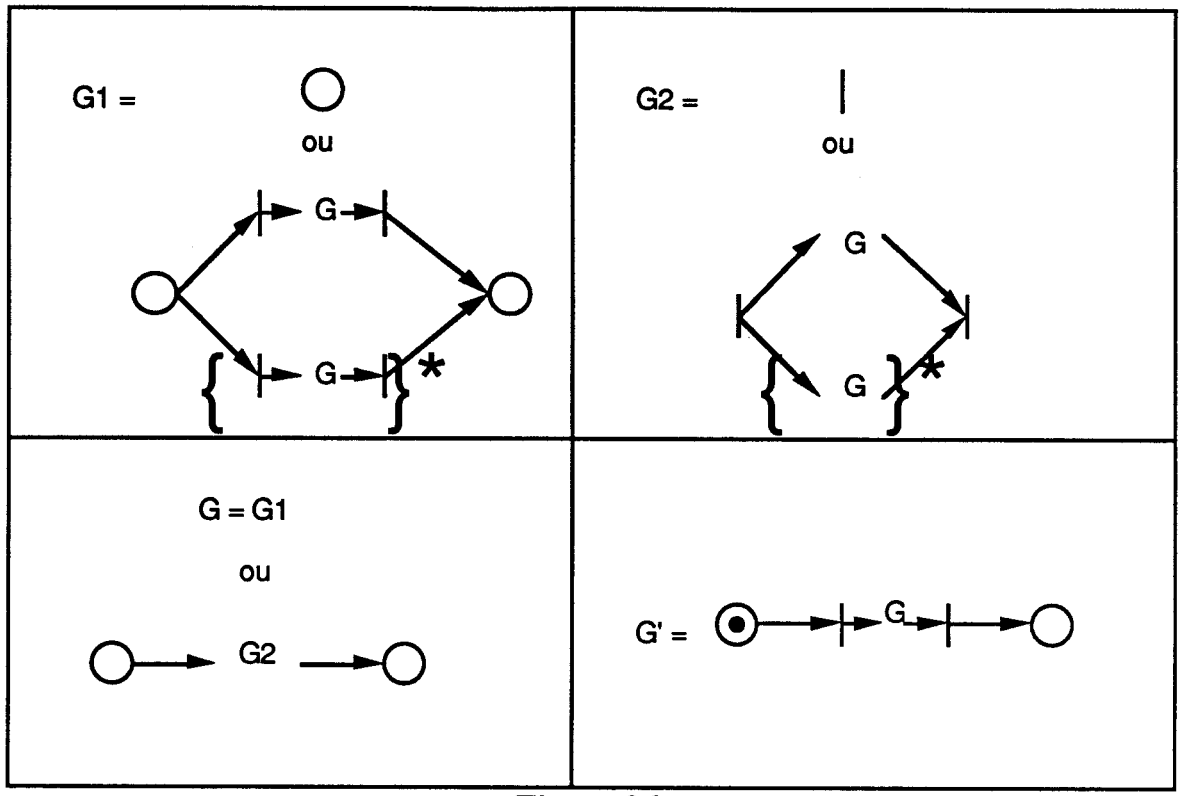


Figure 86

Nous pouvons démontrer que, d'une part, G' bouclé est fortement connexe, monomarqué et borné. D'autre part tous les circuits de G' bouclé passent par la place marquée. Donc G' est également quasi vivant et à terminaison propre.

Cependant aucune de ces deux grammaires ne permet de réaliser l'assemblage de pièces séparées en une pièce finie. Il faut, sur ce type de cas, vérifier les propriétés 2 et 3 sur le réseau.

Nous proposons à cette fin un ensemble de primitives de composition de graphes d'état et de graphes d'événement dont la validation est plus délicate que pour les deux grammaires précédentes, mais permettent plus de souplesse de traitement.

Le premier ensemble de primitives concerne les graphes d'événements. Ils traduisent les liaisons de synchronisation d'état entre des entités dont les niveaux d'abstraction sont différents (liaison entre un lot et ses constituants, ou entre un lot de lots et les lots qui le constituent). Ils définissent donc le squelette des liaisons inter-niveaux de la gamme logique.

Les deux primitives sont les suivantes :

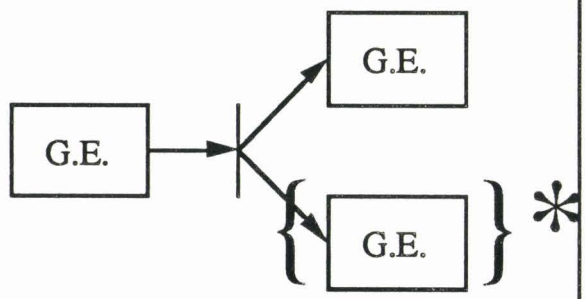
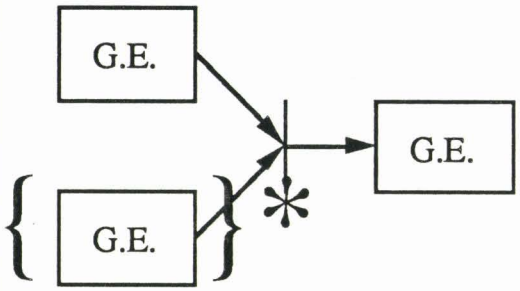
Début du traitement parallèle des composants d'un lot ou désassemblage	
Fin du traitement parallèle des composants d'un lot ou assemblage	

Figure 87

Le second ensemble concerne les graphes d'état :

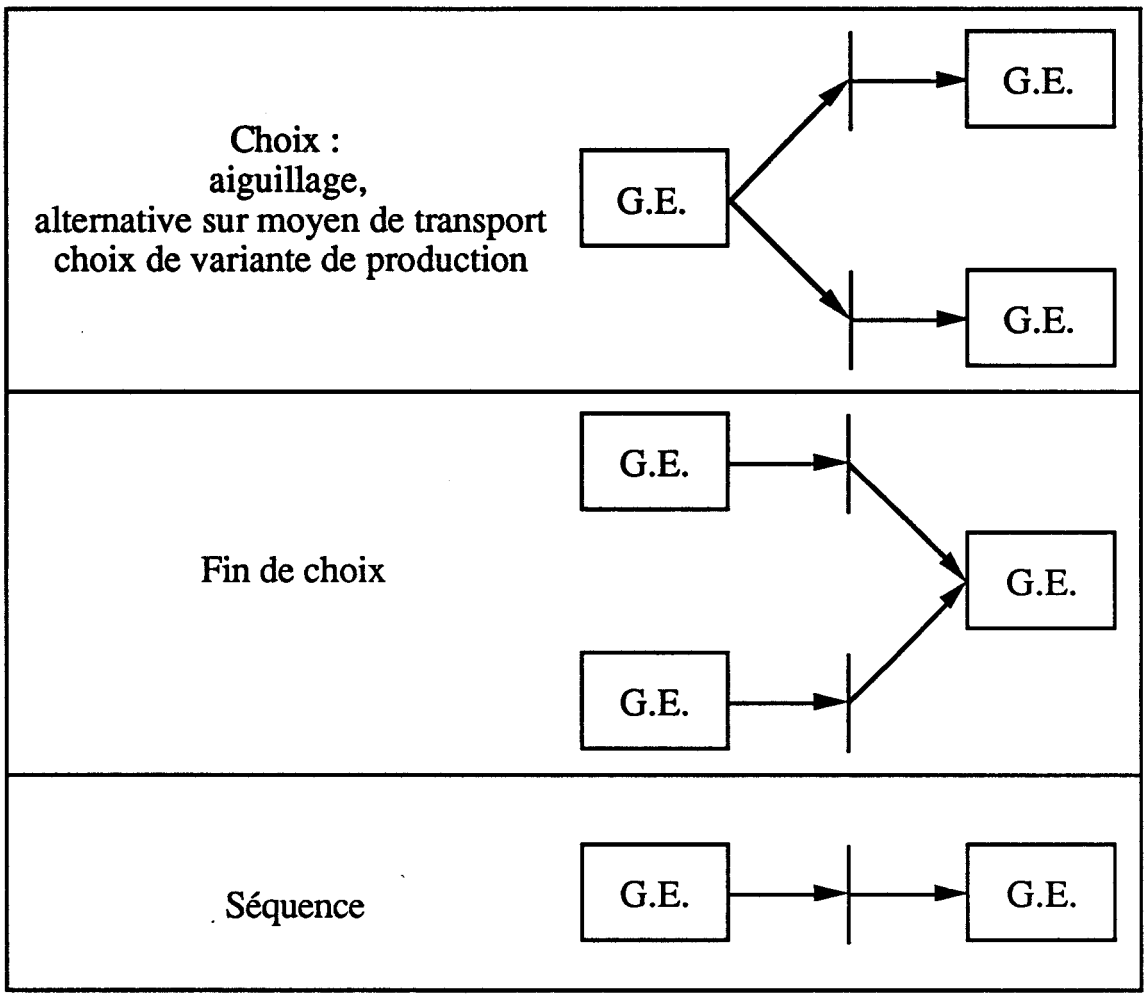


Figure 88

Les rebouclages sont possibles.

Il faut donc s'assurer que les deux propriétés 2 et 3 sont effectivement vérifiées sur le réseau construit à l'aide de ces 5 primitives.

IV.3.5. Cas des gammes logiques dont les arcs ont une pondération supérieure à 1

Les propriétés 2 et 3 ne concernent que les réseaux ordinaires dont les arcs ont un poids de 1. Nous avons vu dans les sections IV.1.3. et IV.1.4. que, en ce qui concerne le graphe d'état d'une pièce seule, la coloration, nécessaire pour les fonctions de test, ne sert qu'à rendre déterministe une alternative conditionnée par le test. La propriété 2 doit donc être vérifiée indépendamment de la coloration.

Dans le cas de réseaux dont le poids de certains arcs est supérieur à 1, les propriétés doivent être vérifiées sur le réseau qui est construit à partir du réseau initial en multipliant les places et les arcs afin qu'ils soient pondérés à 1.

Ainsi, le réseau suivant :

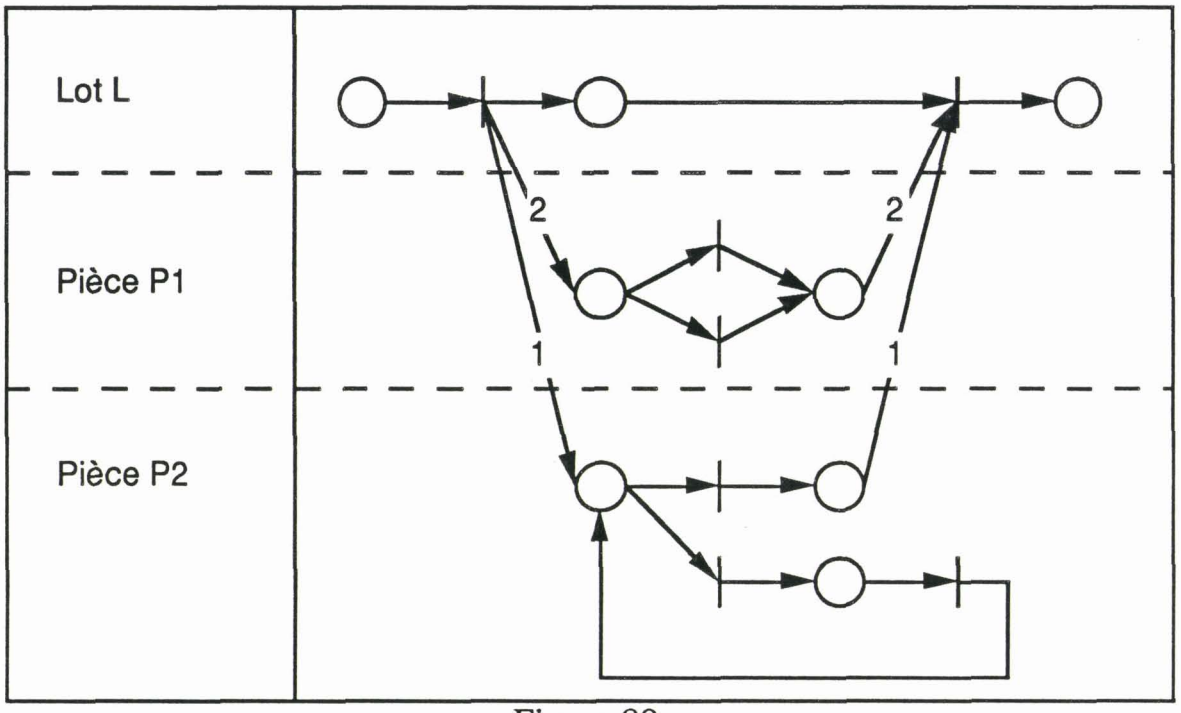


Figure 89

doit être transformé pour obtenir le réseau équivalent de la Figure 90 :

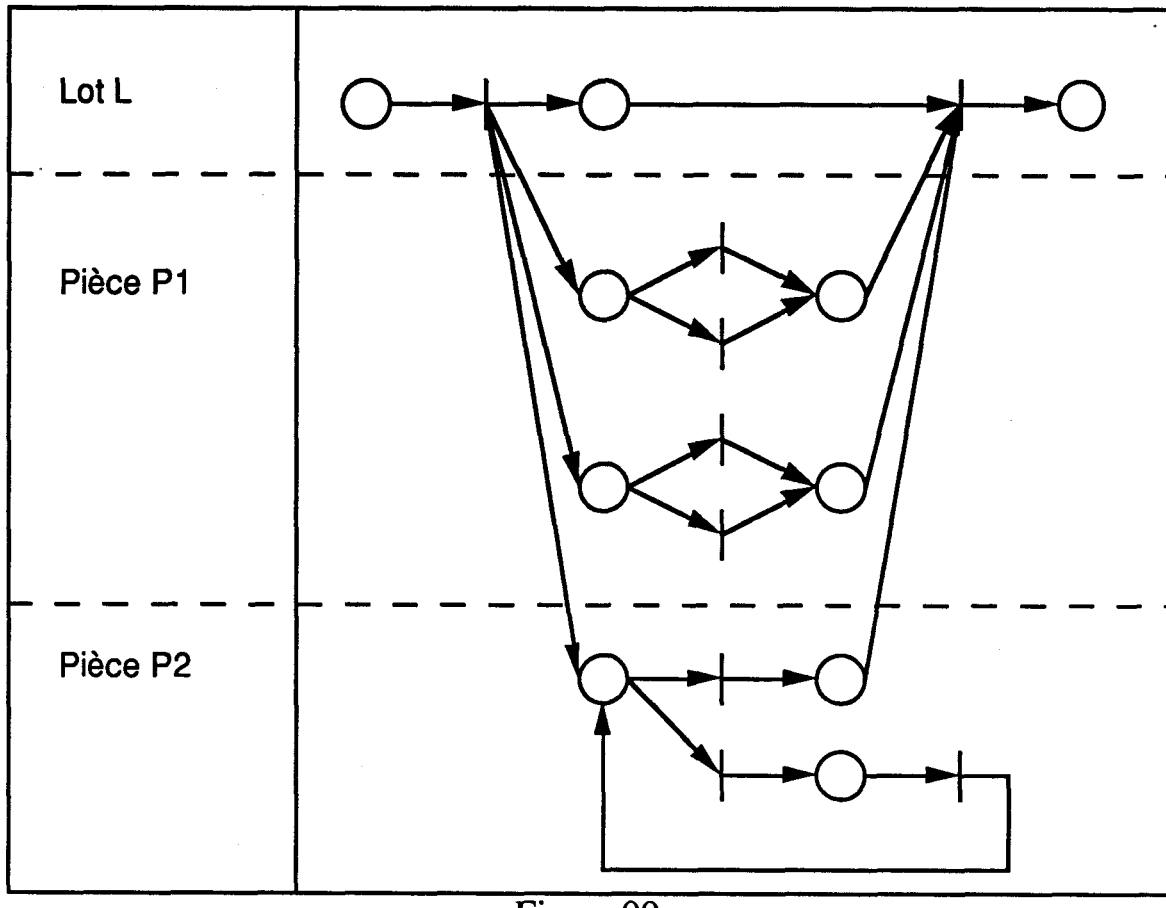


Figure 90

Les propriétés sont vérifiées sur ce réseau démultiplié. Il est équivalent au premier. Il est clair que si, pour les pièces séparées P1 et P2, le graphe d'état vérifie la propriété (2), et que les poids des arcs des 2 transitions du lot sont les mêmes, alors la gamme logique vérifiera la propriété (3).

Moyennant quelques précautions (poids des arcs identiques en amont et en aval des graphes d'état), il est possible de composer les gammes logiques tout en conservant les propriétés de terminaison propre et de quasi vivacité.

La transformation des réseaux à Prédicats-Transitions et à Objets est faite dans le même esprit. Ici, les arcs génèrent des n-uplets donc il faut ajouter autant de places que d'individus ou d'objets du n-uplet (arité du n-uplet).

Les propriétés sont vérifiées sur le réseau transformé. Dans tous les cas rencontrés, en effet, si l'on suppose que le lot ou la pièce est seul dans le RdP de sa gamme logique, alors il est possible de raisonner comme s'il s'agissait d'un réseau ordinaire. Les RdPO ne servent qu'à garantir des liens virtuels entre lot et pièces, ou entre pièce et pièce. Mais si l'on suppose que le lot traité est seul dans sa gamme logique, alors les liens avec ses composants n'ont pas lieu d'être vérifiés. Ils le sont nécessairement. On peut donc se ramener à un RdP ordinaire.

De manière générale, on se ramène toujours au RdP ordinaire pondéré à 1, quel que soit la nature du réseau d'origine. Les opérations de transformation (par expansion) sont regroupées en deux catégories selon la nature du réseau :

- réseaux ordinaires et colorés dont les poids des arcs sont supérieurs à 1,

- réseaux à Prédicats Transitions et à Objets dont les n-uplets ne sont pas des singletons.

Les propriétés 2 et 3 sont vérifiées sur ce réseau expansé.

IV.3.6. Validation des gammes opératoires : conservation de la propriété de terminaison propre de la gamme logique dont est issue la gamme opératoire

La génération des gammes opératoires consiste, du strict point de vue du modèle RdP, à ajouter :

- des alternatives dans les graphes d'état,

- des places et des transitions supplémentaires par rapport à la gamme logique,

- des synchronisations lors d'assemblages avec une contrainte d'antériorité.

D'après la démarche que nous avons présentée au § IV.2.3., il est aisé de vérifier les propriétés (2) et (3) qui traitent respectivement des graphes d'état et des graphes d'événement. Il s'avère que dans de très nombreux cas, les propriétés de quasi-vivacité et de terminaison propre de la gamme logique d'origine sont conservées pour la gamme opératoire. Les modifications apportées au graphe initial sont élémentaires et dans de nombreux cas n'altèrent pas les propriétés de quasi-vivacité et terminaison propre du modèle initial.

Conclusion

Ce chapitre nous a conduit à définir une nouvelle méthode de génération des gammes opératoires. L'approche est originale, elle permet notamment d'agréger de manière cohérente et contrôlée les informations initiales

contenues de manière totalement indépendantes au niveau de la gamme logique d'une part et de la description des moyens de production d'autre part.

La synthèse des gammes opératoires est progressive, ce qui constitue à notre avis l'un des principaux apports de la démarche proposée :

- chaque niveau de spécification opérationnelle est pris en compte selon une approche descendante progressive et rigoureuse. La description obtenue au niveau i ne sera pas remise en cause par le niveau inférieur i+1,

- la part du contrôle dévolue à l'Interface est parfaitement identifiée. Cette répartition du contrôle introduit de façon rationnelle le concept d'Interface en permettant au concepteur d'en spécifier les rôles de façon précise,

- en effet, il résulte de la structuration progressive des gammes logiques puis des gammes opératoires de garantir les propriétés impératives de quasi-vivacité et de terminaison propre des gammes.

Enfin, il nous apparait que la méthodologie proposée puisse conduire à une automatisation, permettant d'assister l'Ingénieur dans la démarche de conception d'un automatisme complexe.

CHAPITRE V

PRISE EN COMPTE DES PROTOCOLES D'ACCES AUX RESSOURCES ET ANALYSE DES BLOCAGES

INTRODUCTION

Dans le chapitre IV, nous avons présenté la manière dont nous prenons en compte progressivement, au cours de la conception du système de production, l'aspect opérationnel du système de production. Cette démarche est menée par étapes, et validée au fur et à mesure de l'affinement de la description.

Cependant les notions de contraintes d'accès n'ont pas été explicitées. En effet, une même zone opératoire (une machine par exemple) peut apparaître à plusieurs reprises dans une même gamme opératoire ou dans plusieurs gammes indépendantes.

Nous allons, dans le chapitre V, présenter le mécanisme de gestion de ces contraintes (cf §V.1.), ainsi qu'une proposition visant à identifier un grand nombre de situations bloquantes (cf §V.2.) que nous appliquerons à notre exemple.

V.1. Spécification des protocoles d'accès aux lieux physiques

V.1.1. Le principe de la spécification

Définition :

Le protocole d'accès à une ressource est un mécanisme traduisant l'allocation et la libération de la ressource.

Ce protocole d'accès, dans la méthodologie CASPAIM-2, est exprimé explicitement sur les gammes opératoires. Il parait en effet naturel de considérer que les ressources nécessaires aux opérations sur une pièce soient allouées directement à la pièce. Ainsi, une pièce qui doit subir une opération de tournage requiert l'allocation de la ressource emplacement de stockage (ES) du Tour, l'utilise, puis la restitue en fin d'opération.

La mise en œuvre de cette démarche consiste à enrichir la gamme opératoire :

- en ajoutant autant de place que de ressources partagées,

- en définissant le mécanisme d'allocation et de restitution au moyen

d'arcs RdP connectant la place image de l'état de la ressource à la gamme opératoire.

Ce mécanisme traduit donc le fait qu'une ressource ne peut être allouée qu'à une seule pièce à un instant donné. Certaines machines nécessitent une gestion cohérente de l'ensemble de leurs emplacements de stockage, ce cas doit être considéré de manière non restrictive.

Prenons à titre d'exemple, le cas des pièces P et P' dont les gammes opératoires sont décrites sur la Figure 1.

Ex 1:

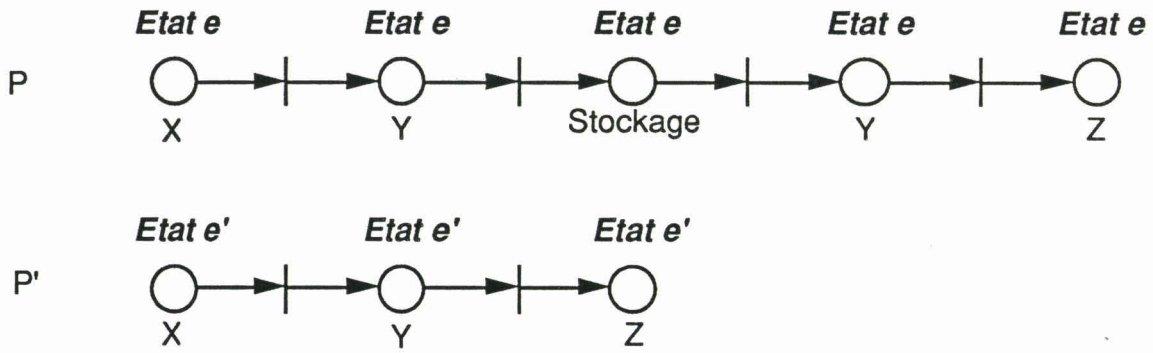
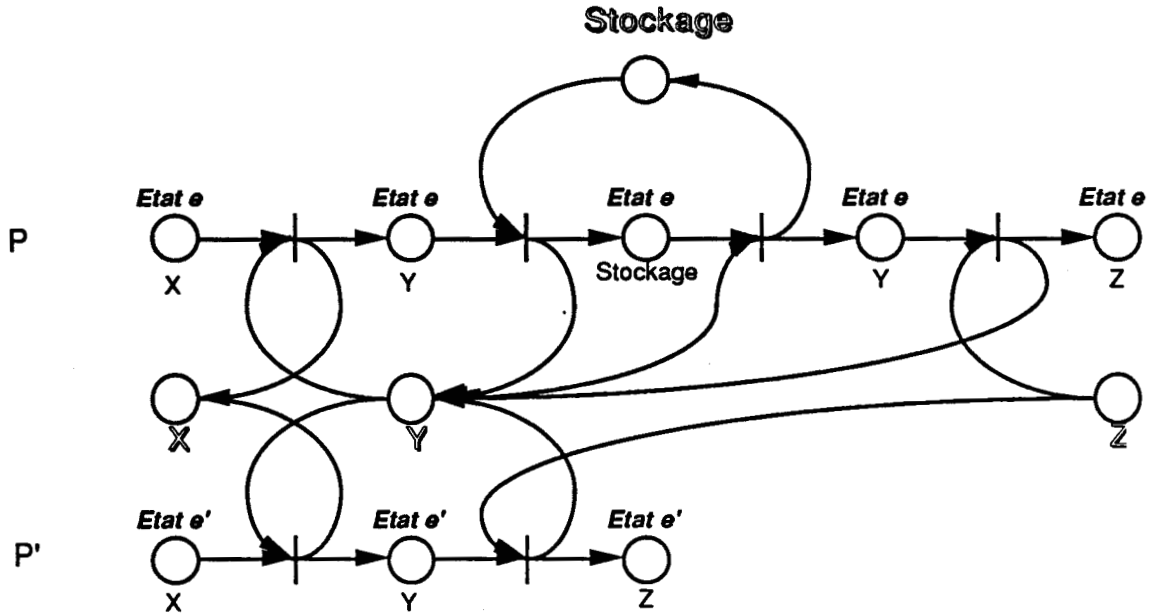


Figure 1

Définir les protocoles d'accès aux lieux X, Y, Z et Stockage consiste à spécifier les mécanismes d'allocation des emplacements de stockage élémentaires associés aux lieux.

Nous supposons que toutes les ES des lieux sont équivalents vis à vis de l'allocation. Nous créons ainsi une place RdP par lieu. Chaque place contiendra initialement autant de jetons que d'emplacements du lieu. Le fait qu'un jeton soit dans cette place signifie que l'ES qui lui est associé est libre (non alloué).

A titre d'illustration, supposons que le résultat de la spécification des protocoles d'accès aux lieux soit, par exemple, le suivant :



X : place contenant les jetons non alloués du lieu X

Figure 2

Il est ici possible de définir tout type de protocole d'accès à Y. En effet, s'il parait souhaitable de continuer à réserver l'emplacement Y lors du chargement sur la zone de stockage, on définira le schéma suivant :

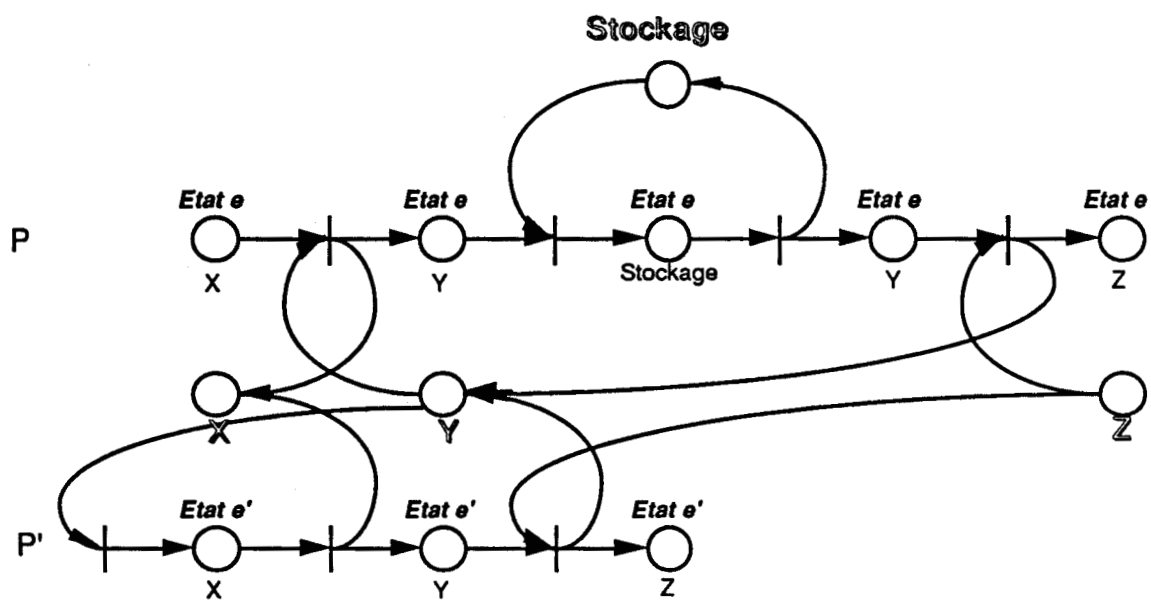


Figure 3

L'ES de Y reste effectivement alloué à la pièce P lorsque celle-ci se situe sur la zone de Stockage. Pour la pièce P', l'ES de Y est alloué avant que la pièce ne soit sur X. L'allocation est, dans ce cas, de plus longue durée que pour

la Figure 2. Nous verrons au §V.2. qui traite des blocages, que cette solution est plus sûre mais également moins performante que la solution de la Figure 2.

Cet exemple simple illustre la liberté dont dispose le concepteur pour spécifier ses protocoles. Cependant un certain nombre de vérifications doivent être réalisées pour assurer le bon comportement du système de production malgré l'introduction des contraintes supplémentaires introduites par les protocoles ainsi créés.

V.1.2. Validation de la spécification

La spécification des protocoles d'accès aux lieux physiques par le concepteur doit vérifier les trois conditions suivantes :

1. Tous les lieux de stockage d'objets (préhenseur, plateau etc...) doivent être gérés, ils devront donc être systématiquement étudiés dans ce sens,

2. L'ES d'un lieu doit être alloué AVANT que l'objet y accède effectivement,

3. Chaque objet restitue toutes les ressources qu'il utilise.

Les seules exceptions à la condition 3 sont les lieux d'entrée et de sortie, dont les protocoles sont partagés avec les unités de production amont et aval de l'unité considérée.

Malgré ces trois contraintes, le concepteur dispose d'un grand nombre de degrés de liberté pour spécifier les accès.

La condition 1 est la conséquence des hypothèses exposées au chapitre II concernant le procédé commandé. On suppose en effet que la PC maîtrise l'intégralité des commandes du procédé. Cela implique naturellement que l'ensemble des objets commandés par la PC soit géré explicitement, et en particulier le mécanisme d'allocation des ressources.

La condition 2 relève du bon sens. En effet, sa non satisfaction entrainerait inévitablement des collisions et des défaillances dues à une gestion inadaptée.

La condition 3 permet de s'assurer que, pour une pièce donnée, il n'y a pas de perte de jeton, et par là même blocage par famine du fait d'une mauvaise gestion du point de vue de la pièce. Il est clair que la condition 3 n'est pas suffisante pour garantir le non blocage du réseau complet, mais qu'elle est cependant nécessaire.

L'exemple précédent vérifie bien ces trois conditions. Nous pouvons remarquer que, dans le cas d'une gamme opératoire qui ne comporte pas d'alternative, la condition 3 est vérifiée si la somme des poids des arcs entrants dans une place de gestion de protocole est identique à la somme des arcs sortants.

Voyons maintenant le cas des gammes opératoires plus complexes.

Considérons la gamme opératoire suivante :

Ex 2

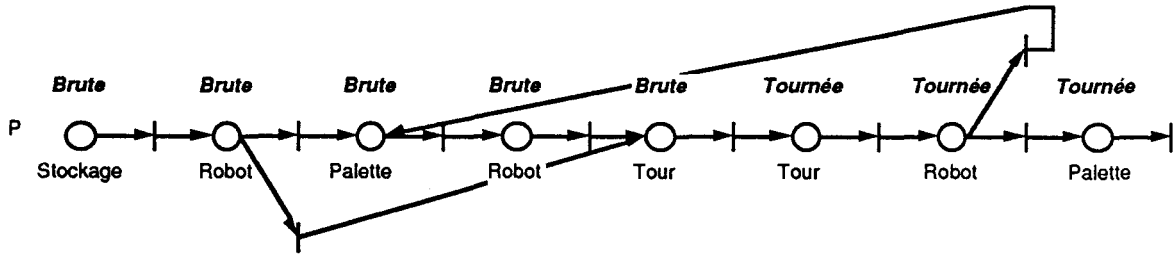


Figure 4

Elle traduit :

- le transfert de la pièce P du stockage vers le Tour via une palette ou directement vers le Tour,

- le tournage de la pièce,

- le rebouclage sur la palette dans le cas d'un usinage incomplet,

- l'évacuation vers le stockage via la palette lorsque la pièce est terminée.

La forte connexité du réseau bouclé issu de ce réseau normalisé permet de garantir les propriétés de quasi-vivacité et de terminaison propre de la gamme opératoire (voir § IV.3.).

Nous devons ensuite définir les conditions d'accès aux lieux Stockage, Robot, Palette et Tour. Nous ne faisons aucune hypothèse sur la capacité de chacun de ces lieux (nombre d'ES élémentaires). Les différents ES d'un lieu sont cependant équivalents vis à vis des fonctions d'allocation.

Considérons les protocoles suivants décrits sur le Figure 5 pour les trois lieux étudiés :

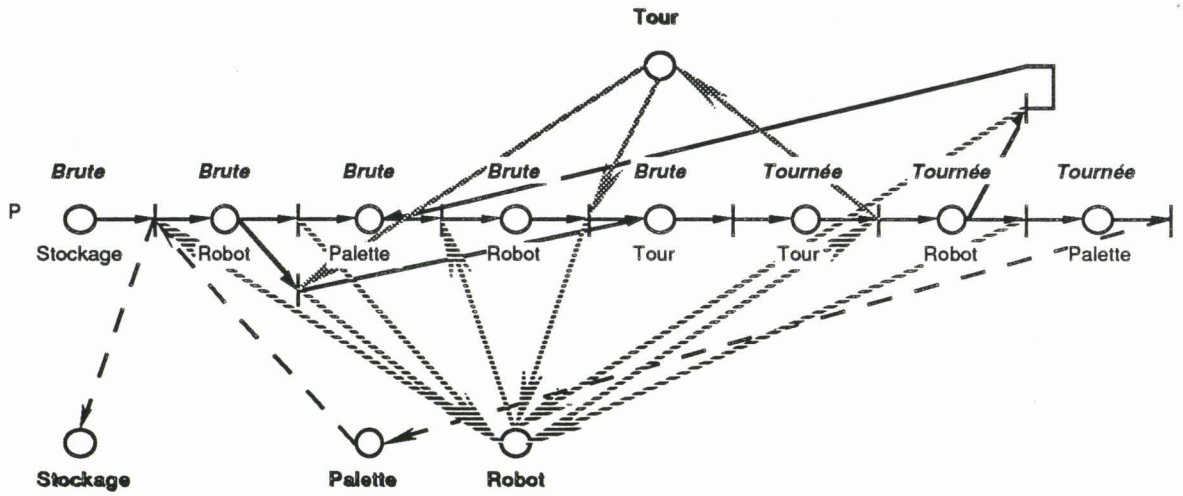


Figure 5

Les trois conditions sont vérifiées.

Nous proposons une notation simplifiée des protocoles d'accès. Elle consiste à regrouper sur un arc amont et un arc aval de la transition les noms des places de gestion d'accès.

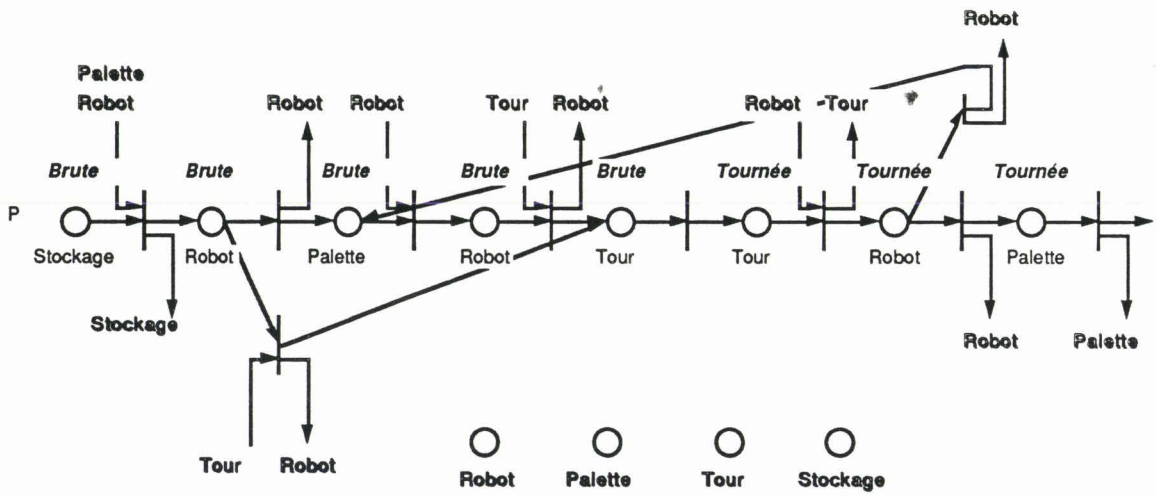


Figure 6

Dans cette configuration, l'ES de la palette reste alloué à la pièce durant le Tournage. L'ES de la palette est donc maintenu en allocation un long moment bien qu'elle soit vide.

De plus, l'ES du Tour est alloué tardivement. Il peut en résulter un blocage dans le cas où, en fin d'opération sur le Tour, le Robot aurait été chargé pour alimenter le Tour, et ne serait de fait plus disponible pour assurer son déchargement. Il conviendrait alors d'allouer l'ES du Tour avant le déchargement de la palette.

Nous obtenons le réseau suivant :

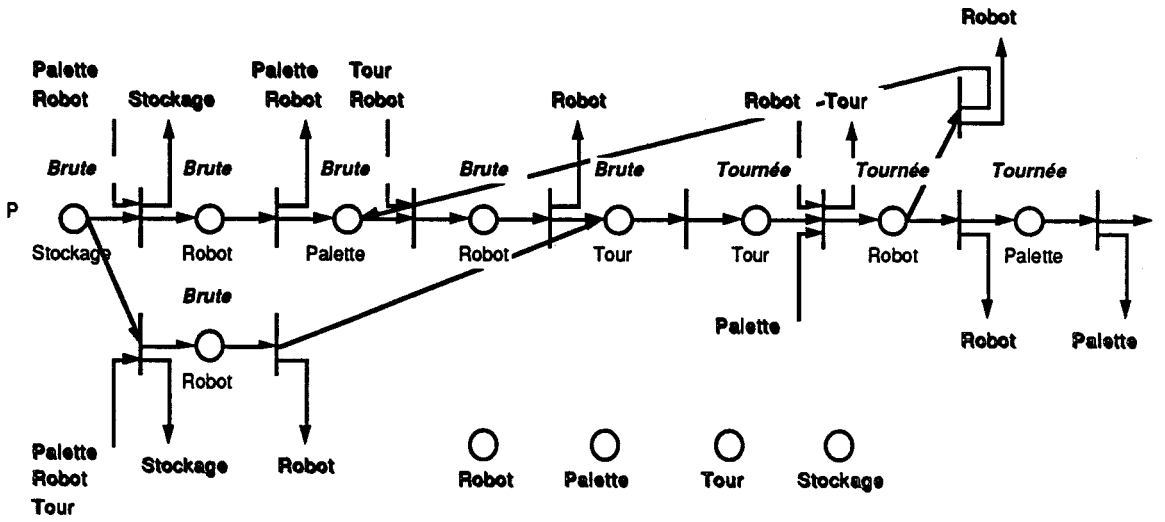


Figure 7

Nous pouvons constater que nous avons été conduit à dupliquer la place (Brute, Robot) qui suit la zone de stockage afin de satisfaire la condition 3 (allocation de l'ES du Tour différente selon l'origine de la pièce).

De manière générale, lorsqu'il existe des alternatives dans le graphe d'état d'une pièce, il peut être nécessaire de dupliquer des états lorsque les conditions d'accès ne sont pas les mêmes pour chaque branche. Il est à noter que la forte connexité du réseau bouclé ne s'en trouve pas modifiée.

Considérons le cas d'un assemblage ternaire (Bille + Pointe + Cartouche) (BOU 90). Il est composé de deux assemblages binaires indépendants Bille+Pointe et Pointe+Cartouche. Ici la pointe doit être fixée avant les transferts de la Bille et de la Cartouche. Il s'agit donc d'un assemblage avec contrainte d'antériorité dont la gamme opératoire est la suivante :

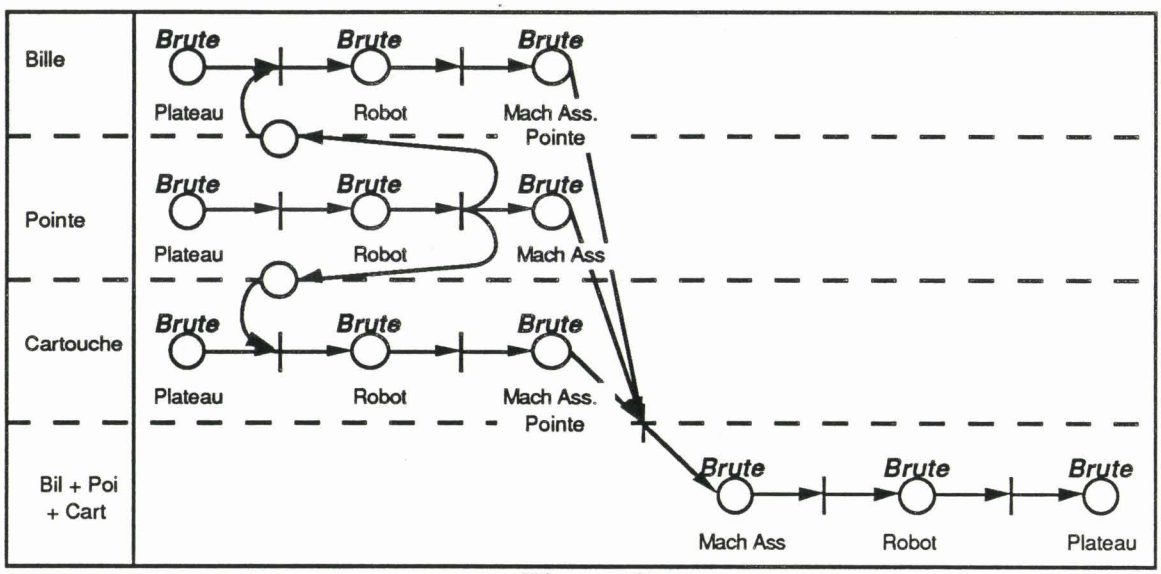


Figure 8

Les trois pièces sont originaires du même lieu (le plateau). Nous souhaitons conserver un ES alloué parmi les 3 afin d'y loger la pièce résultante. On peut ainsi garantir qu'il existera un ES en fin d'assemblage. Nous pouvons choisir de ne pas libérer l'ES associé à la Cartouche par exemple.

La machine d'assemblage ne supportera que la pointe. L'ES qui lui est associé est libéré lorsque l'ensemble quitte la machine d'assemblage.

Le protocole d'accès au lieu Mach Ass.Pointe est déjà explicité du fait de la contrainte d'antériorité. Le résultat de cette spécification est donc, par exemple, le suivant :

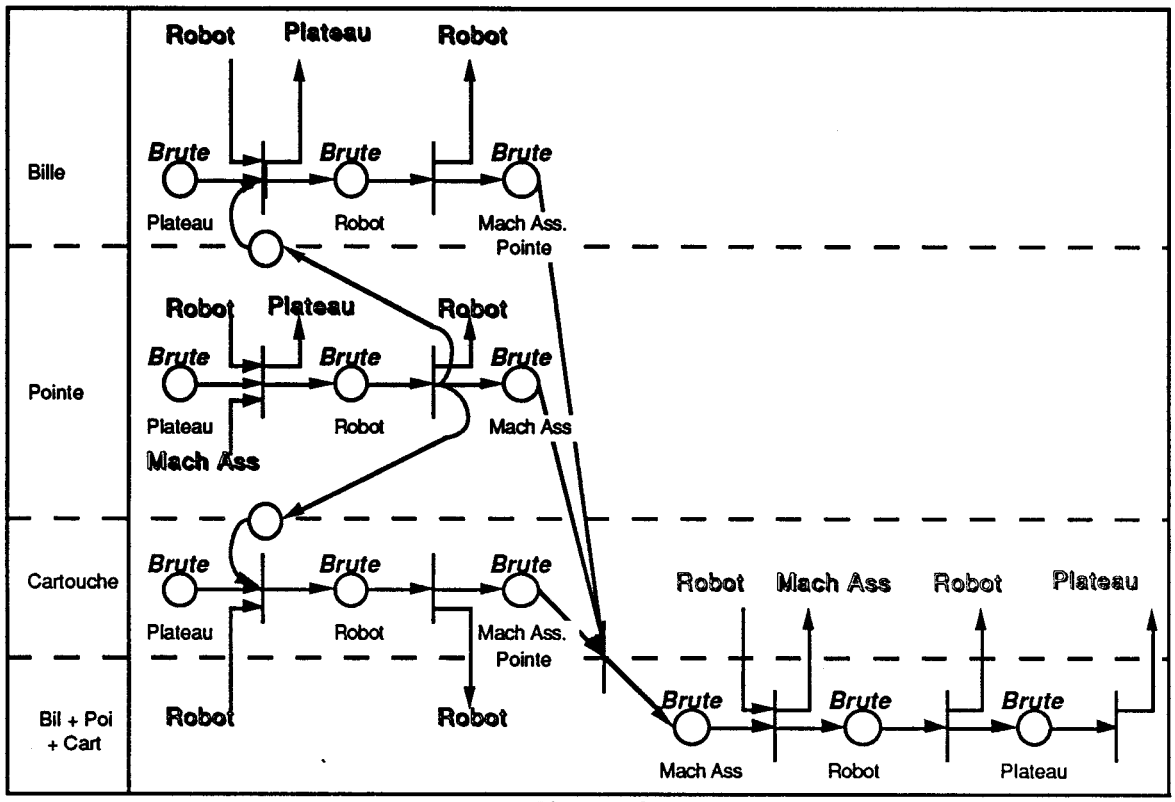


Figure 9

Considérons maintenant le cas d'un assemblage pur c'est à dire un assemblage pour lequel il n'existe pas de contrainte d'antériorité d'une pièce par rapport à l'autre. Il s'agit de notre exemple d'application Ecrou + Vis. La portion de gamme opératoire est la suivante :

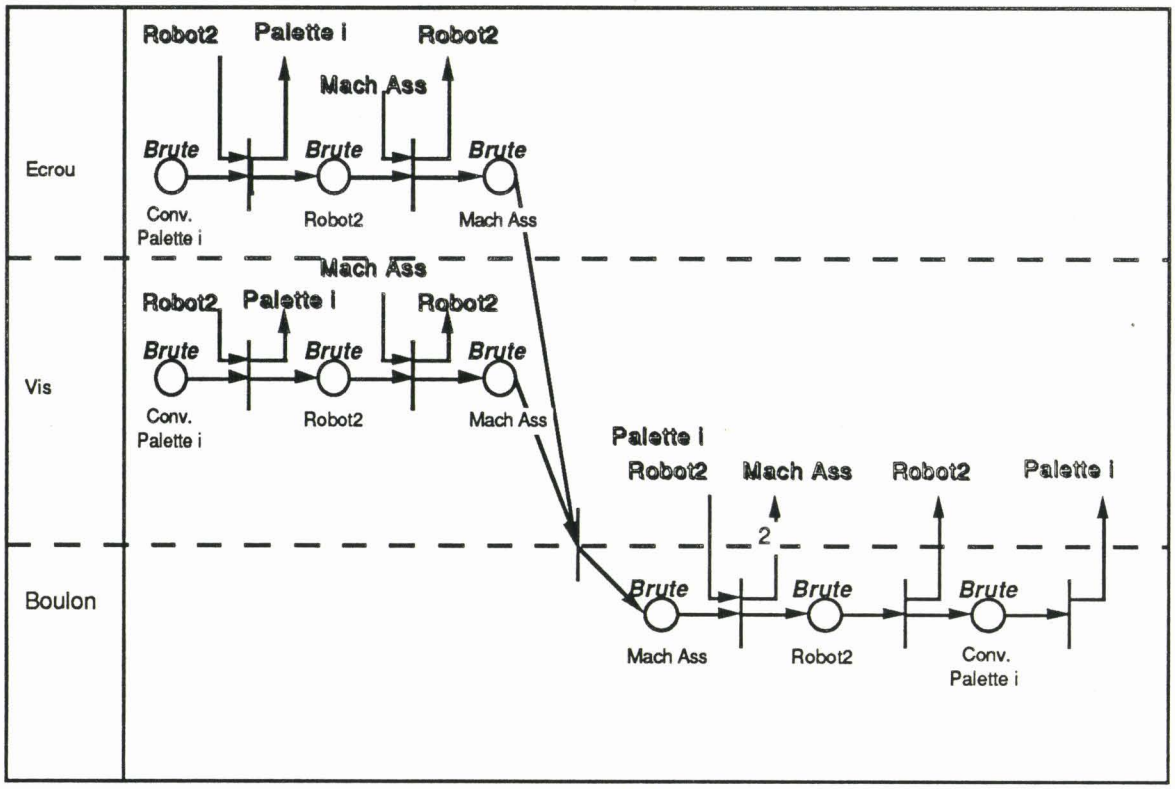


Figure 10

Dans ce cas, il est nécessaire de libérer les deux ES de la Machine d'Assemblage lorsque le Boulon quitte la machine. Les ES des palettes sont libérés immédiatement après leur déchargement.

La démarche de spécification qui vient d'être détaillée sur quelques exemples peut être effectuée pour chaque étape de la génération assistée des gammes opératoires. Associée à la phase de recherche des situations bloquantes, il devient possible de valider le comportement de l'unité de production par affinements successifs.

La spécification des protocoles d'accès pour l'étape de premier niveau de notre exemple est représenté sur les deux figures contigues suivantes :

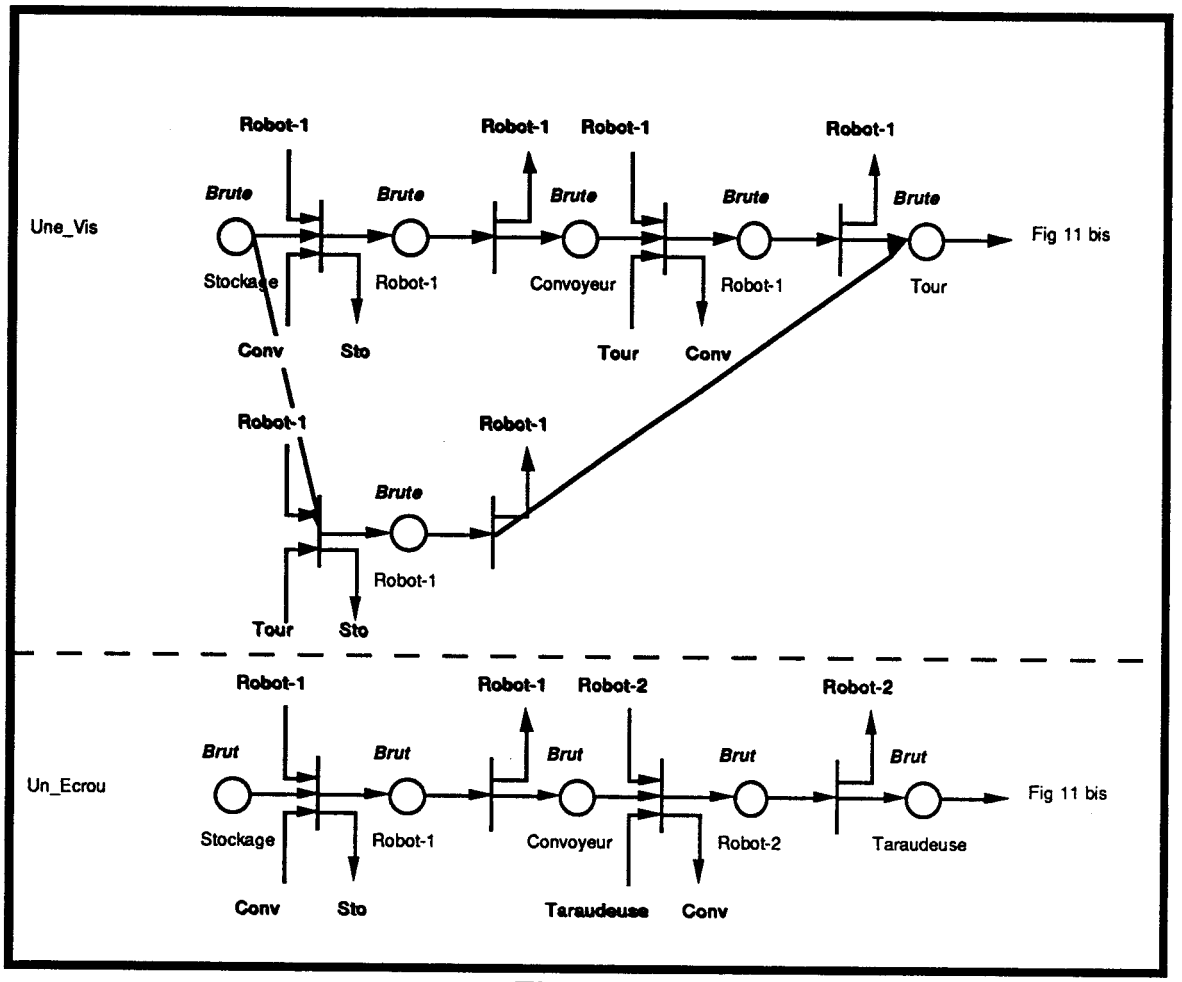

Une_Vis
Robot-1
Brute
Stockage
Conv
Sto
Robot-1
Brute
Robot-1
Brute
Convoyeur
Tour
Robot-1
Brute
Robot-1
Brute
Tour
Conv
Tour
Sto
Robot-1
Brute
Robot-1
Fig 11 bis
Un_Ecrou
Robot-1
Brut
Stockage
Conv
Sto
Robot-1
Brut
Robot-1
Brut
Convoyeur
Taraudeuse
Conv
Robot-2
Brut
Robot-2
Brut
Taraudeuse
Fig 11 bis

Figure 11

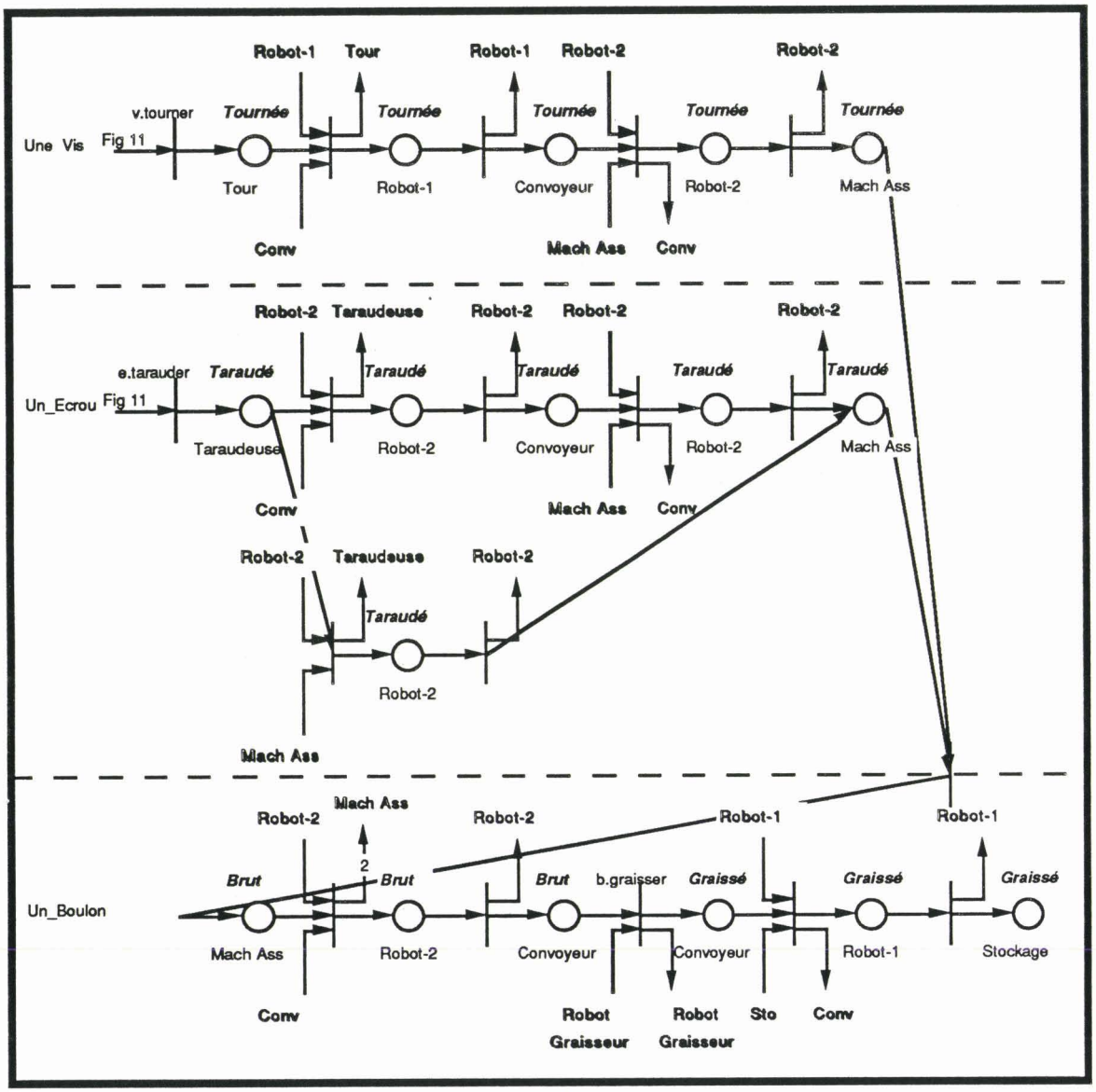


Figure 11 Bis

Cette spécification vérifie les 3 conditions citées précédemment.

La spécification des protocoles est menée à bien de la même façon pour les étapes suivantes de la génération des gammes opératoires.

Il serait possible de l'assister de manière significative au moyen d'un outil informatique, et de faciliter par là même le travail du concepteur en le déchargeant d'une vérification fastidieuse.

Nous allons détailler la mise en œuvre des protocoles les plus utilisés.

V.1.3. Les protocoles usuels

Le protocole d'accès utilisé dans l'ancienne démarche CASPAIM consiste à allouer l'emplacement de la zone de travail de destination avant de quitter la zone de travail de départ. Ce mécanisme est réalisé au moyen du processus dédié au transfert par une liaison Prod/Cons et une liaison Req/Ack (voir § II.2.2.). Cette stratégie est conforme à l'idée d'assurer a priori que l'objet émis sera bien réceptionné, et évite a priori un grand nombre de blocage.

Exprimé sur la gamme opératoire d'une pièce, ce protocole prend la forme simplifiée suivante :

Ex 3:

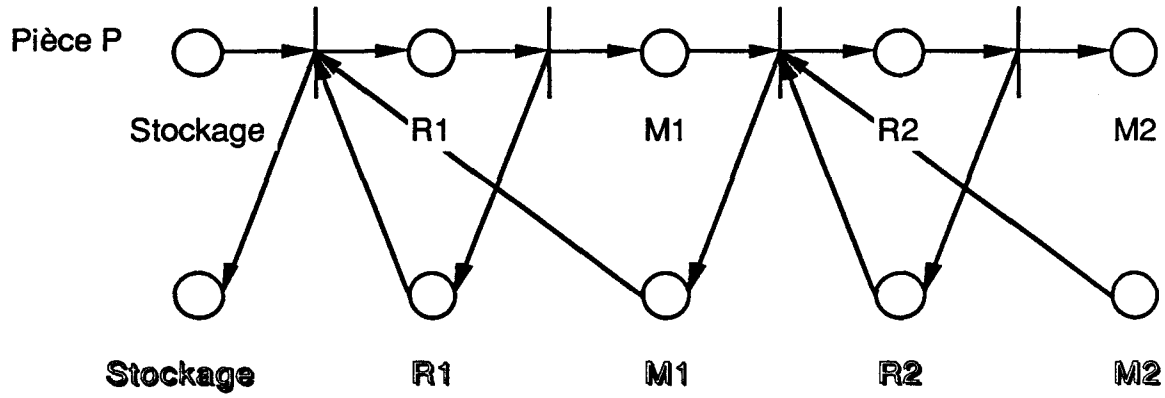


Figure 12

Il traduit le mécanisme dans lequel :

- le transfert du Stockage vers M1 via R1 ne peut se faire que si M1 peut recevoir la pièce,

- la même condition entre les lieux physiques M1 et M2.

Nous verrons que ce protocole d'accès aux lieux de travail M1 et M2 est sûr (non bloquant) vis à vis de la pièce seule, mais qu'il est quelque peu contraignant du point de vue des durées.

Le deuxième protocole d'accès est connu sous le nom d'**accès en temps masqué** (BEA 90). Il consiste à commencer le transfert vers un lieu sans que l'ES du lieu destinataire ne soit alloué. L'allocation ne se fait alors qu'en cours de transfert.

Dans ce cas bien sûr, il se peut que le manipulateur de transport soit bloqué dans l'état chargé, en attendant que l'ES de la zone destinataire se

libère. Il conviendra alors de pouvoir assurer la libération de la zone destinataire.

L'exemple précédent traité en temps masqué vis à vis de M1 et M2 est le suivant :

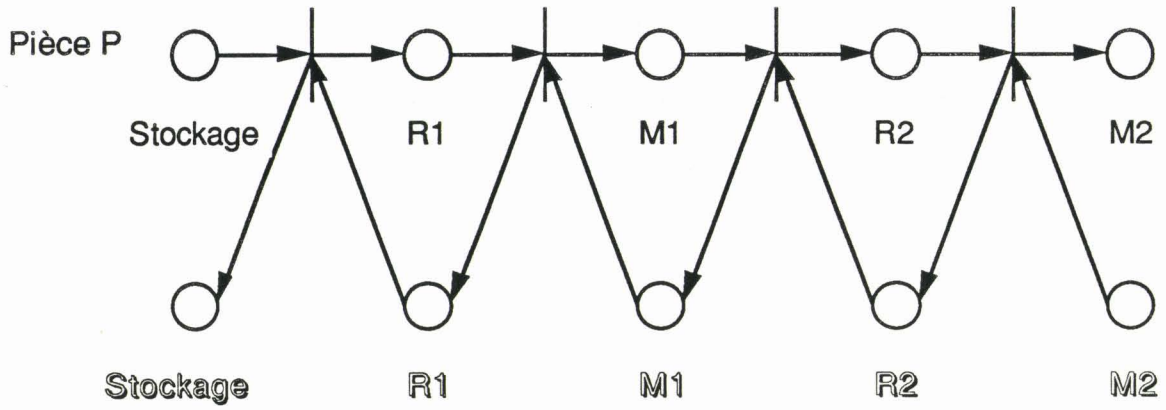


Figure 13

Pour ce protocole, les ressources (ES) sont allouées au plus tard, aussi bien pour les robots que pour les machines. Il s'agit d'un protocole très efficace du point de vue des performances.

Il est cependant plus délicat à gérer. Imaginons en effet que les ressources R1 et R2 soient une seule et même ressource. Le protocole devient alors le suivant :

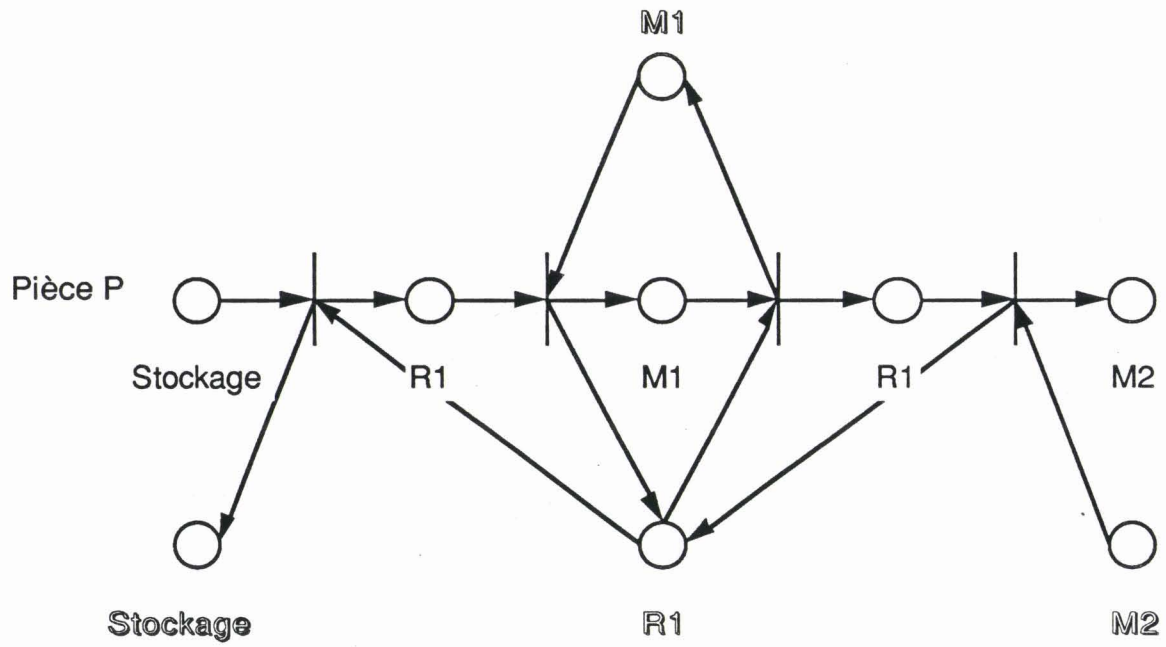


Figure 14

Prenons les hypothèses suivantes :

- R1 possède un ES (une pince),

- M1 possède un ES (un tour).

La configuration dans laquelle le lieu M1 est chargé avec une pièce terminée, et le robot R1 est chargé avec une pièce à destination de M1 se traduit par le marquage suivant :

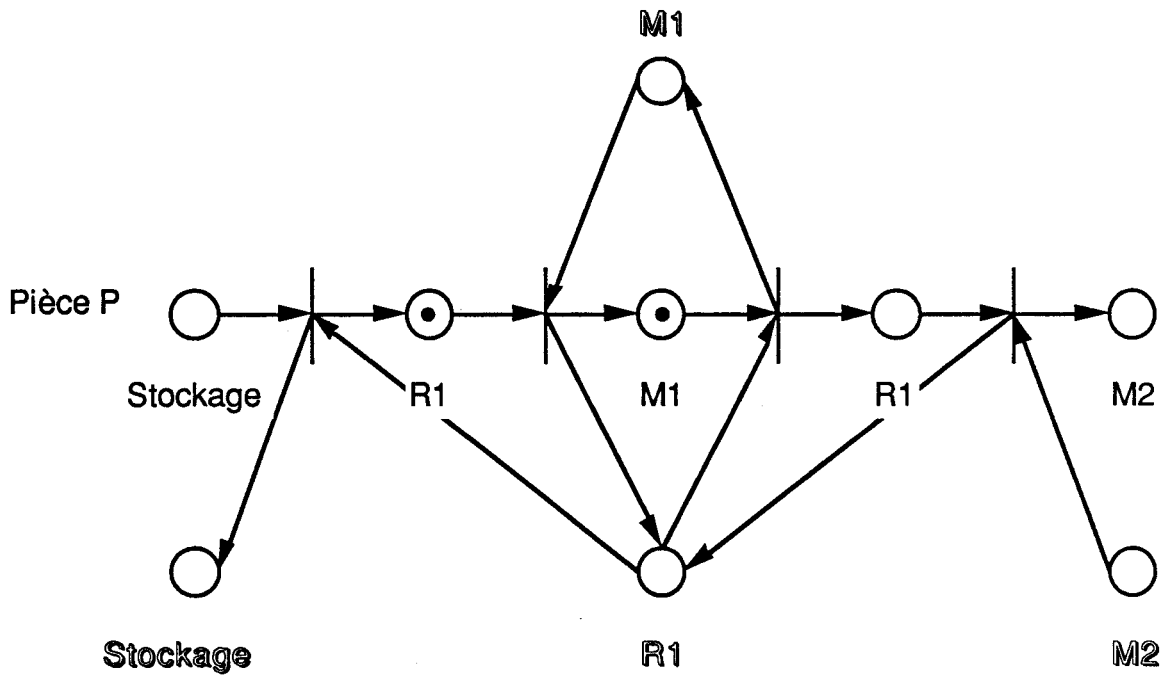


Figure 15

La pièce de M1 ne peut pas être évacuée puisque R1 n'est pas disponible, et la pièce de R1 ne peut pas être évacuée puisque M1 n'est pas disponible.

Il s'agit donc d'un dead-lock ou étreinte fatale pour les deux pièces et les lieux M1 et R1. Toutes les pièces qui transitent, au cours de leur gamme opératoire, par R1 ou M1 seront donc également bloquées.

Les deux protocoles que nous avons présentés recouvrent la majorité des applications traitées à ce jour par le laboratoire.

Voyons en détail une mise en œuvre, spécifique à la productique, de l'identification des blocages.

V.2. Deux techniques d'identification des blocages

Nous avons vu sur l'exemple précédent qu'une mise en œuvre simple de protocole peut conduire à des blocages. Cette section traite de l'analyse statique du RdP généré dans l'objectif d'en extraire les marquages bloquants. Elle traite également de la mise en place d'une gestion visant à éviter les états bloquants à partir de cette analyse.

Les résultats présentés ci-après ne se veulent en aucun cas exhaustifs. Ils ne font qu'illustrer des cas très particuliers de blocages qui, nous semble-t-il, concernent une partie non négligeable des blocages dans les cellules flexibles de production manufacturière.

Des études complémentaires concernant la recherche de verrous en général devraient être menées sur la base des réseaux prenant en compte les protocoles d'accès aux lieux physiques.

V.2.1. Cas particulier de protocole quasi-vivant par construction

Prenons l'hypothèse selon laquelle la gamme opératoire est un réseau normalisé. Nous supposerons de plus qu'il s'agit d'un **graphe d'événement**.

Nous cherchons à exploiter le théorème de Commoner (BRA 83) (cf § IV.3.3.) concernant les graphes d'événement selon lequel :

Le RdP <R; M> est vivant SSI tout circuit élémentaire contient au moins une marque.

Nous pouvons donc définir les protocoles d'accès aux lieux en conservant la propriété de graphes d'événements, et en vérifiant la vivacité du réseau bouclé.

Par définition, une place ne peut avoir qu'un arc en amont et un arc en aval. Par conséquent, une ressource libre ne peut être allouée que pour une seule tâche. Il n'existe pas d'indéterminisme concernant le choix de l'allocation de la ressource.

De plus, les lieux d'entrée et de sortie doivent être à capacité infinie, ce qui permet de ne pas gérer les ES qui leur sont attachés, et donc de ne pas introduire de place ne possédant pas d'arc amont (entrée) ou aval (sortie).

Ainsi, l'exemple 3 muni des protocoles de la Figure 13, excepté les lieux d'entrée et de sortie, est effectivement un réseau à une place d'entrée et une

place de sortie :

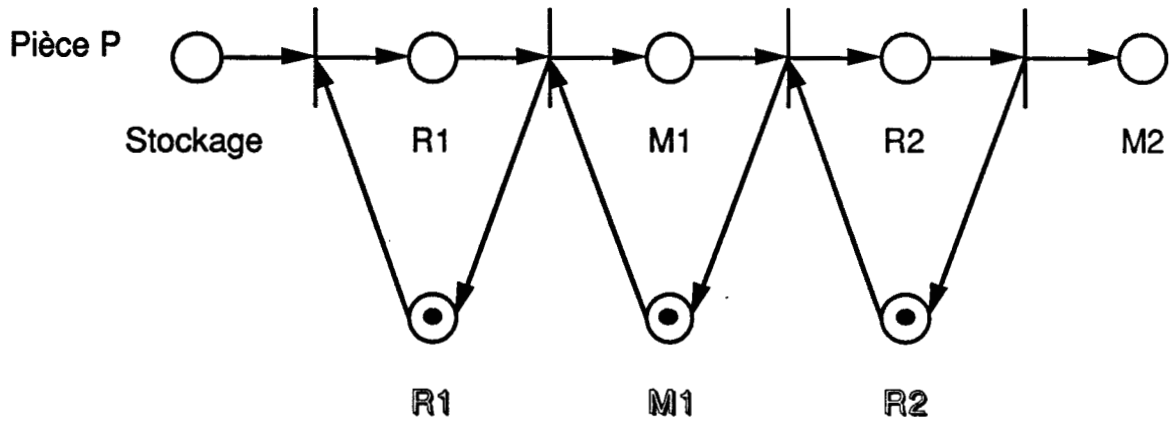


Figure 16

Le réseau bouclé issu de la Figure 16 est un graphe d'événement (M2 et Stockage sont reliées par une transition), et de plus tous les circuits élémentaires contiennent une marque au moins. Ce réseau marqué est donc quasi-vivant. Nous nous sommes ramenés à un RdP à capacité.

Il le sera pour tout marquage initial pour lequel le stockage contient au moins un jeton.

Il s'agit bien sûr du mécanisme de gestion de postes de travail à la chaîne, dont on sait qu'il n'existe pas de situation bloquante (sauf par famine de pièce).

Le réseau suivant est également quasi-vivant :

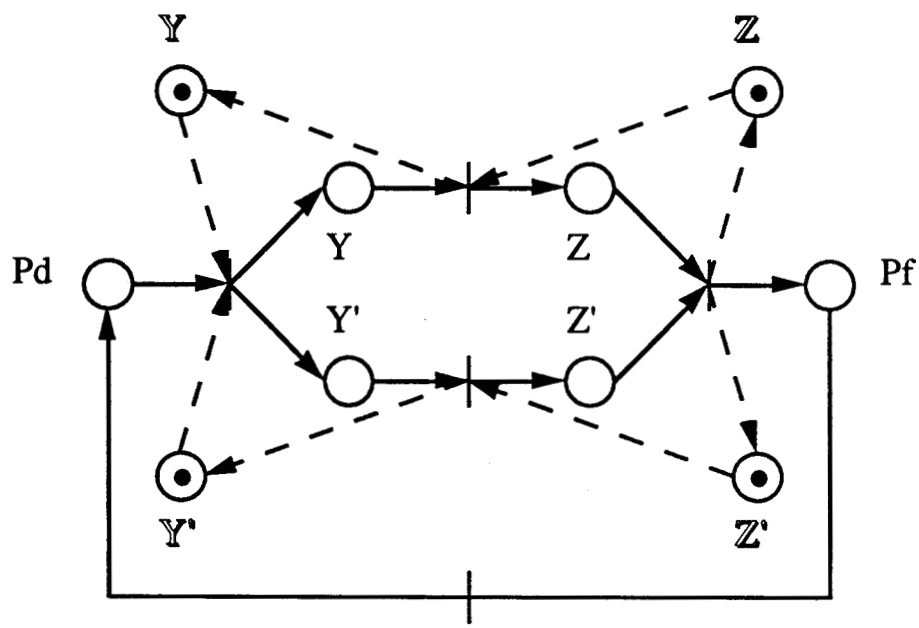


Figure 17

Les hypothèses excluent donc :

- les cas d'alternatives dans le traitement des pièces,

- les cas d'alternatives dans l'utilisation des ressources.

Bien que très contraignantes, elles peuvent convenir dans le cas où le routage des pièces et les machines de traitement sont fixées une fois pour toutes, et sans possibilité de flexibilité.

Notons à ce propos les travaux de COHEN (COH 89) relatifs à l'évaluation temporelle de graphes d'événement, ainsi qu'à des méthodes de réduction de réseau par des méthodes algébriques basées sur l'algèbre des dioïdes. Il s'agit en particulier du dioïde (R, max, +). Cette méthode permet de générer la séquence et la chronologie de sorties des pièces à partir de la connaissance de la séquence d'entrée de pièces, en faisant l'hypothèse de traitement au plus tôt dans le réseau.

Bien que très satisfaisantes (analogie avec la transformée en z dans le domaine des systèmes à état continu), ces méthodes sont d'un intérêt limité dans la réalité malheureusement plus complexe des systèmes industriels étudiés. Aucune flexibilité ne peut être mise en œuvre de cette manière.

Nous avons donc identifié le cas particulier simple d'un protocole quasi-vivant par construction. Il s'agit :

- d'un graphe d'événement,

- pour lequel toutes les ressources ne sont utilisées que pour accomplir une seule opération.

Voyons maintenant une condition suffisante, concernant le RdP, menant à une étreinte fatale locale.

V.2.2. Condition suffisante de dead-lock local sur les circuits

Considérons à nouveau l'exemple 3 associé au protocole de la Figure 14.

Nous avons déjà montré Figure 15 que ce protocole peut mener à des marquages bloquants vis à vis de certaines pièces.

En partant de cet exemple nous proposons d'identifier un grand nombre de situations bloquantes.

Le marquage atteint met en évidence qu'une pièce P utilise la ressource M1 et nécessite la ressource R1 pour évoluer, et une autre pièce P, inversement, utilise R1 et attend M1.

Le marquage atteint est tel que :

- les places $\mathbb{M}1$ et $\mathbb{R}1$ sont vides,

- les deux places $\mathbb{M}1$ et $\mathbb{R}1$ appartiennent à un circuit dans le graphe biparti associé au RdP de la gamme opératoire d'une pièce,

- les deux transitions du circuit ne sont pas tirables puisque les places $\mathbb{M}1$ et $\mathbb{R}1$ ne sont pas marquées,

- ces deux transitions ne seront plus franchissables car les deux ressources ne peuvent pas être restituées.

De manière plus formelle, les types de blocage que nous cherchons à mettre en évidence correspondent à la configuration suivante :

- il existe un ensemble de n lieux physiques, dont les places qui en assurent la gestion forment un circuit dans la gamme opératoire munie des protocoles d'accès, sans qu'aucune place de la gamme opératoire appartienne au circuit,

- toutes ces places sont vides,

- toute ressource ES allouée de ces lieux physiques doit être libérée par une transition qui appartient au circuit.

Il s'agit d'un cas particulier de RdP bloquant. Cependant, dans le contexte de la productique, il recouvre à notre sens la majorité des situations bloquantes rencontrées.

Considérons le cas de deux zones de stockage S1 et S2 fixes dont les objets sont échangés au moyen de deux robots communicants (ils s'échangent des objets entre eux).

On appelle P une pièce qui transite de S1 à S2, via R1 et R2,
et on appelle P' une pièce qui transite de S2 à S1, via R2 et R1.

Les gammes opératoires et les protocoles sont décrites Figure 18 :

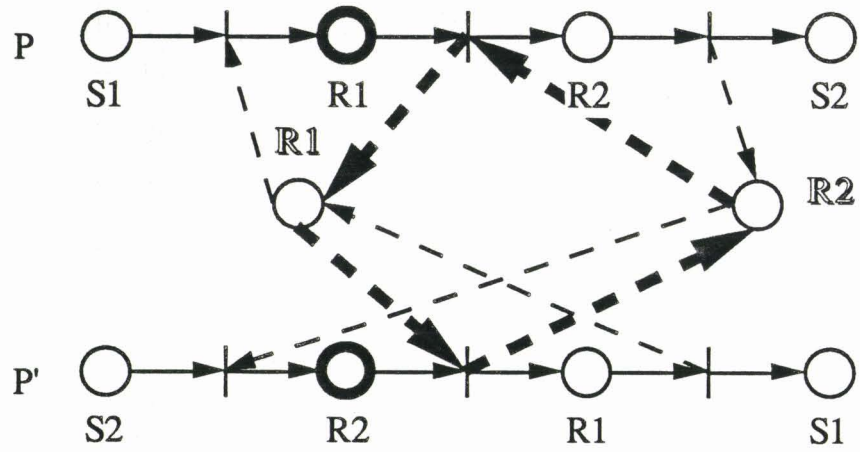


Figure 18

Notons que les ressources sont toujours restituées au plus tôt dans notre démarche. Pour schématiser, le raisonnement nous dirons que deux lieux :

(i) dont les protocoles d'accès sont identiques ou au moins permutables,

(ii) et qui sont utilisés dans deux séquences inversées dans deux gammes ou une seule

sont potentiellement blocables.

Le réseau suivant correspond à ce schéma :

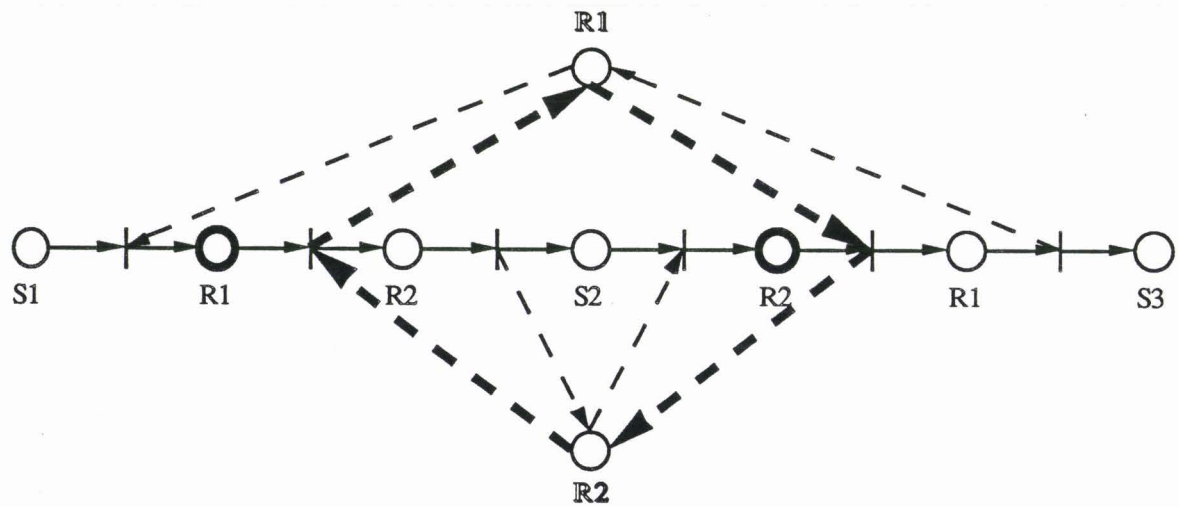


Figure 19

Nous pouvons remarquer que Figure 18 et 19, nous nous sommes ramené à nouveau à un circuit de places de gestion de ressource vides et dont les transitions ne sont pas tirables, ce qui justifie d'autant notre choix de recherche de marquages bloquants.

V.2.3. Méthode d'identification d'une catégorie particulière de blocages

La démarche d'identification consiste à rechercher les configurations du réseau qui remplissent les conditions que nous avons détaillées :

Condition 1

- il existe sur le graphe un circuit constitué des places caractéristiques des états des ressources associées aux lieux physiques, à l'exclusion des places caractéristiques de l'état effectif de ces mêmes lieux (places des gammes opératoires),

Condition 2

- toutes ces places sont vides,

Condition 3

- toute ressource ES nécessairement allouée de ces lieux ne peut être libérée que par une transition qui appartient au circuit.

La recherche des circuits est menée de manière systématique à partir de l'ensemble des gammes opératoires munies de leurs protocoles d'accès, au moyen d'algorithmes itératifs (GON 79), ou récursifs appropriés.

En phase d'exploitation, nous nous attachons à contrôler les circuits identifiés.

Il existe de nombreux travaux concernant la validation formelle des RdP, qu'il s'agisse de méthodes de réduction (BER 83, BER 85, EST 85, EST 87) ou d'analyse de pseudo-vivacité par résolution d'un système linéaire (AGA 87).

Il existe deux approches concernant les blocages en phase d'exploitation.

La première consiste à laisser le système évoluer librement. Lorsqu'un blocage est détecté, un mécanisme de préemption libère certaines ressources afin que le réseau puisse évoluer à nouveau. Cette technique est appelée technique a posteriori (RAY 85).

La seconde, appelée technique a priori, consiste à empêcher la formation de blocages. Elle est mise en œuvre selon deux possibilités :

-1° On définit un ordre total sur les ressources, et un processus doit demander les ressources dont il a besoin selon cet ordre. L'ordre étant total sur les ressources, ce mécanisme empêche la formation de blocages. L'inconvénient majeur est bien sûr que les ressources sont monopolisées plus longtemps que nécessaire, les performances sont donc dégradées.

-2° Par avance, on procède pour chaque processus à la recherche du bilan de leurs besoins maximaux en ressources. Une procédure évalue les configurations futures en fonction des bilans évalués et des modèles de chaque processus afin de calculer les allocations non bloquantes. Ce mécanisme est connu sous le nom de l'algorithme du banquier (CRO 75, BEA 90). Il suppose en particulier une inférence de type chaînage avant avec remise en cause, ce qui est pénalisant dans notre contexte.

Le choix de l'une ou l'autre de ces techniques dépend de la nature du problème. La technique a priori est une technique pessimiste, qui est mise en œuvre lorsqu'on estime que l'occurence des blocages est importante. La technique a posteriori est par contre optimiste et présuppose des interblocages peu fréquents.

En plus de ces deux critères, nous pouvons remarquer que gérer la préemption sur un processus pose d'importantes difficultés sur une architecture distribuée, notamment en ce qui concerne la cohérence et l'intégrité des données (image de l'allocation des ressources).

Nous avons donc choisi une technique a priori.

Elle consiste à contrôler, en temps réel, les circuits qui ont été isolés en phase de conception. Ce mécanisme de prévention des interblocages a pour fonction de s'assurer que les circuits ne vérifient jamais les conditions 2 et 3.

V.2.3.1. Première solution simplifiée : interdire la saturation

La solution la plus simple vise à ce que la deuxième condition ne soit jamais vérifiée. Exprimée dans notre contexte, cela signifie que l'on ne doit jamais saturer les lieux physiques, ou que l'on ne doit jamais allouer l'ensemble des emplacements du circuit. Il est clair que sous ces hypothèses, aucun blocage ne peut se produire.

De plus, la mise en œuvre de ce mécanisme est simple. L'ensemble des allocations du circuit sont gérées simultanément. Les emplacements sont alloués librement jusqu'à ce qu'il n'en reste plus qu'un. Celui-ci ne sera pas alloué afin que l'ensemble des places de gestion d'accès ne soient pas toutes vides.

Considérons un exemple simple. Il s'agit du chargement et du déchargement en temps masqué d'une zone de stockage (de 10 ES) par un robot à un préhenseur. Le RdP modélisant les gammes opératoires et les protocoles d'allocation est le suivant :

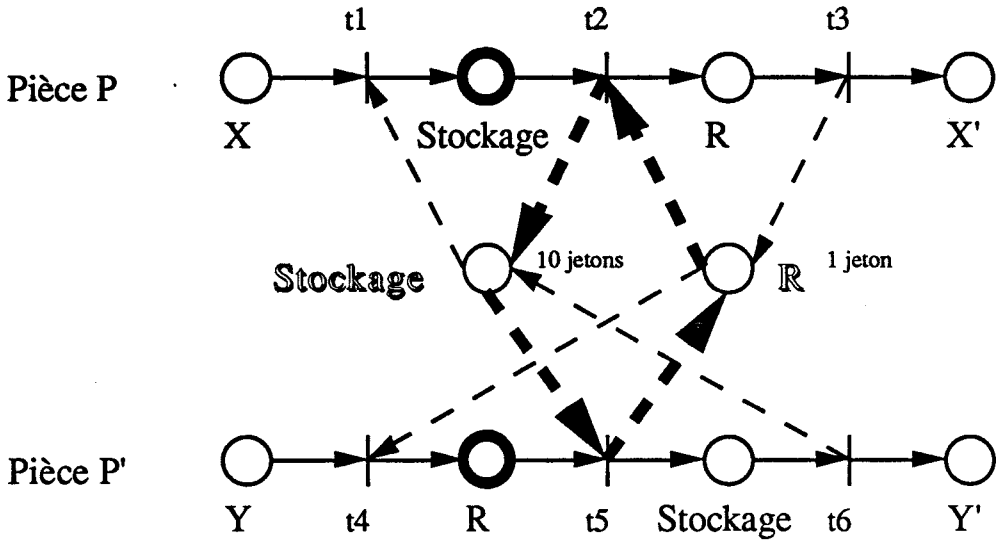


Figure 20

Les deux places de gestion de ressources Sto (pour le lieu Stockage) et R appartiennent à un circuit. Le circuit est matérialisé en gras sur la Figure 20. Une solution simple permettant de ne pas saturer Sto et R simultanément est d'ajouter une place Accès au réseau précédent. Cette place a pour objet de limiter le nombre de jetons alloués dans Sto et R simultanément à la capacité cumulée de Sto et R moins 1, soit 10 jetons au maximum.

Un jeton de la place Accès doit alors être consommé lorsqu'un des jetons de Sto et R l'est, et restitué dans les mêmes conditions. Les transitions t1 à t6 sont donc concernées. Cependant les transitions t2 et t5, qui font partie du circuit contrôlé, ne modifient pas le nombre total de jetons dans les places Sto et R. La place Accès n'est donc reliée qu'aux transitions qui font évoluer le nombre total de jetons dans l'ensemble des places Sto et R, soit les transitions t1, t3, t4 et t6.

Le contrôle d'accès devient alors le suivant :

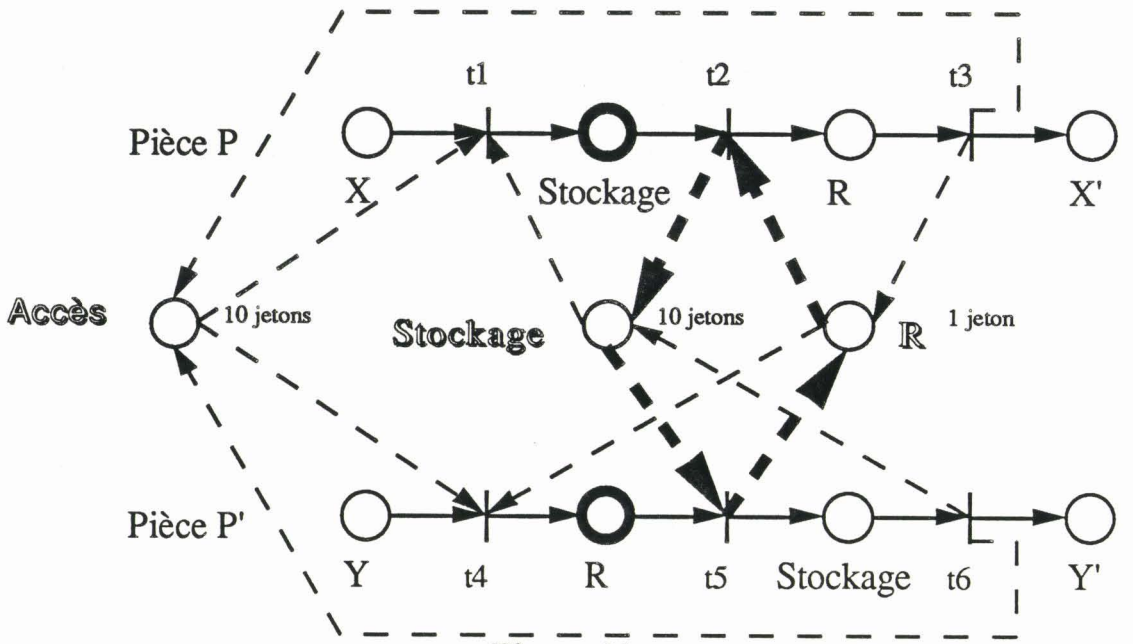


Figure 21

Ce mécanisme est très simple à mettre en place. Nous pouvons remarquer de plus qu'il limite l'utilisation des 11 ES réels à 10 à tout instant. Les dégradations de performances induites par cette limitation sont donc faibles. Il suffirait en effet d'accroître d'une unité la capacité de stockage maximale pour satisfaire une contrainte de performance atteinte pour un stockage de capacité n, le surcoût induit par cette surcapacité n'est pas ici très important.

Il est cependant clair que si le nombre total d'ES du circuit est faible, 2 ou 3, alors cette gestion grossière risque de devenir pénalisante du point de vue de l'efficacité.

Nous sommes alors conduits à tolérer que la condition 2 soit satisfaite (saturation du circuit), tout en garantissant que la condition 3 ne soit pas.

V.2.3.2. Deuxième solution plus fine : contrôle d'accès des circuits bloquants

Examinons la troisième condition menant à un blocage :

Toute ressource ES appartenant aux lieux concernés par le circuit ne peut être libérée que par une transition qui appartient au circuit.

Dans cette hypothèse, ne savons que tous les ES sont alloués. Considérons le cas du lieu Stockage. Tous les ES du Stockage sont alloués, donc l'ensemble des jetons sont dans les places Pièce P lieu Stockage, et Pièce P' lieu Stockage. Or les jetons de la place Pièce P' lieu Stockage peuvent libérer un ES lors du

tir de t6, qui ne fait pas partie du circuit. La condition 3 nous permet d'affirmer que l'ensemble des ES alloués sont dans la place Pièce P lieu Stockage.

Suivant le même raisonnement, il apparaît que le jeton R ne peut être que dans la place Pièce P' lieu R. Le marquage bloquant est donc le marquage pour lequel :

10 jetons sont dans la place Pièce P lieu Stockage,
1 jeton est dans la place Pièce P' lieu R.

De façon plus générale, le marquage bloquant est tel que :

- l'ensemble des jetons des lieux sont alloués,

- l'ensemble de ces jetons sont répartis sur toutes les places qui sont en amont des transitions du circuit considéré.

Ce marquage est effectivement bloquant pour deux raisons conjointes :

- toutes les ressources sont allouées,

- toutes les ressources sont en amont des transitions du circuit, donc aucune ressource ne peut être libérée par une transition autre que celles du circuit lui-même.

Prenons un exemple plus complexe que le précédent.

La pièce P doit transiter par X, Sto, R et X',
la pièce P' doit transiter par Y, R, M, R', Sto et Y'.

Schématiquement, les échanges sont les suivants :

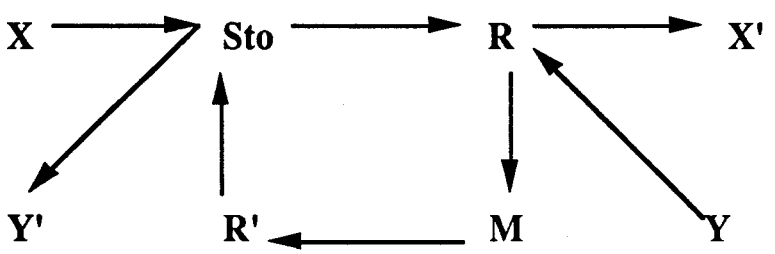


Figure 22

Les protocoles sont définis de la manière suivante :

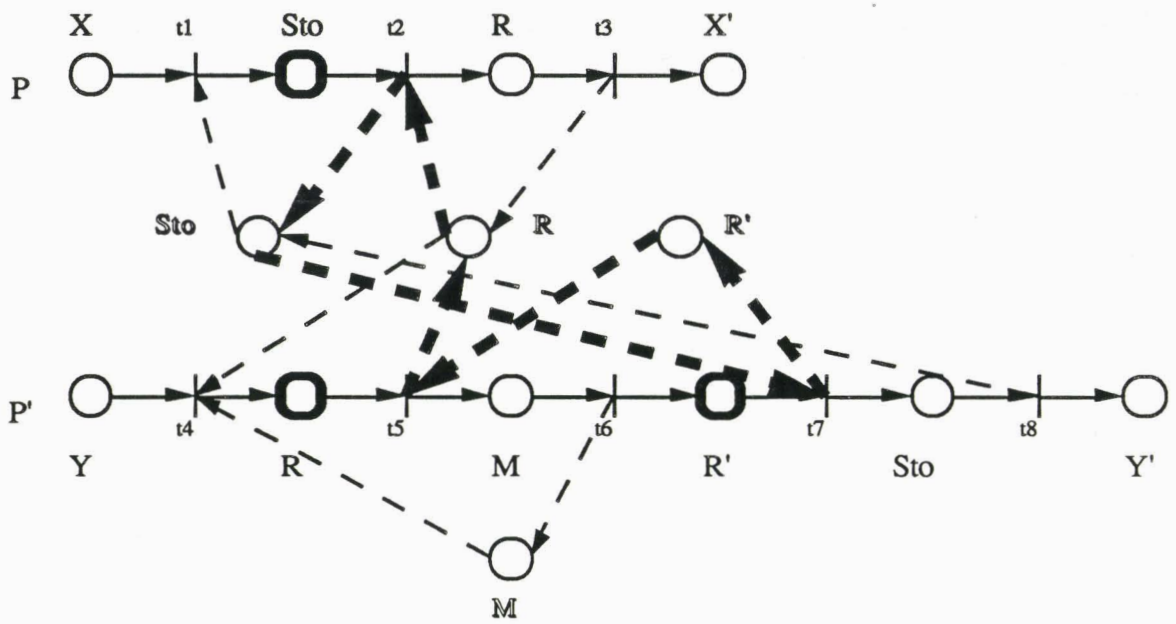


Figure 23

De tels protocoles peuvent être rencontrés dans des cas pratiques industriels particuliers.

Pour mettre en évidence la situation de blocage, il suffit de repérer sur chaque gamme les places amont du circuit, et de considérer la possibilité d'un marquage global de ces places vidant le circuit.

Nous faisons l'hypothèse que Sto a 10 ES, R a 1 ES, R' a 1 ES et M au moins 2. Le marquage suivant est effectivement bloquant :

la place Pièce P lieu Sto a 10 jetons,
la place Pièce P' lieu R a 1 jeton,
la place Pièce P' lieu R' a 1 jeton.

Le mécanisme que nous cherchons à mettre en place pour éviter ces blocages consiste à interdire la saturation des places RdP immédiatement en amont des transitions du circuit.

De même que précédemment, nous ajoutons une place Accès qui contient 10+1+1-1=11 jetons, soit la somme des ES du circuit moins 1.

Pour contrôler la place Pièce P lieu Sto, il faut contrôler la première transition en amont de t2 qui modifie la somme des jetons des places de gestion de protocole Sto, R et R'. Il s'agit de t1 qui consomme 1 jeton de la place Sto.

Pour la place Pièce P' lieu R, il faut contrôler t4 qui consomme 1 jeton de la place R.

Pour la place Pièce P' lieu R', il faut contrôler t4 également puisque t5 ne modifie pas la somme des jetons Sto, R et R'.

Les transitions aval de la place Accès sont donc t1 et t4.

Les transitions amont de la place Accès sont les dernières transitions du circuit vis à vis de chaque pièce, soit t2 et t7.

Le contrôle d'accès est donc le suivant :

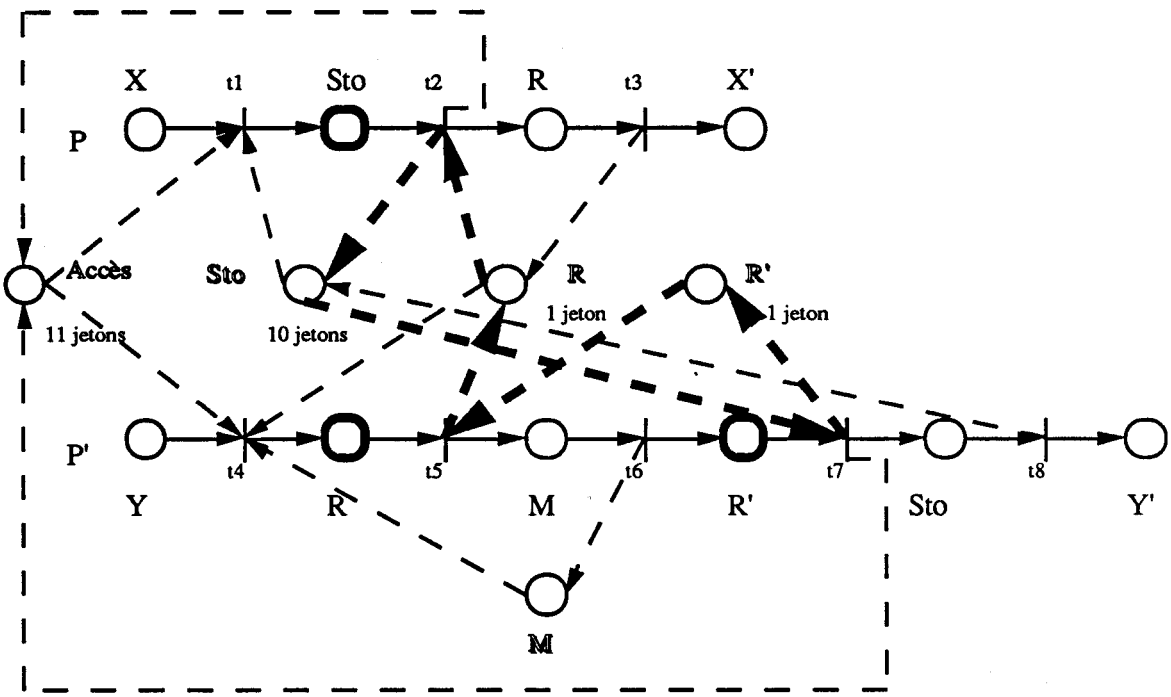


Figure 24

Le marquage bloquant n'est effectivement plus atteignable.

Nous pouvons remarquer que ce contrôle d'accès peut mener à la saturation des 3 lieux Sto, R et R'. En effet, le marquage dans lequel

Pièce P lieu R a 1 jeton,
Pièce P' lieu R' a 1 jeton,
Pièce P' lieu Sto a 10 jetons.

est accessible. Il est cependant non bloquant (R peut être déchargé sur X').

En revanche, il interdit un marquage pour lequel :

Sto est saturé et toutes les pièces de Sto doivent aller sur le lieu R,
R est saturé et la pièce de R doit aller sur R' via M,
R' est saturé et la pièce de R' doit aller sur Sto.

Il est aisé de concevoir qu'une telle configuration est bloquante. Le rôle de la place Accès est de contrôler les arrivées et les sorties de pièces des 3 lieux Sto, R et R' afin qu'une telle situation ne puisse se produire.

Nous avons présenté dans cette section une démarche visant à construire, dès la phase de conception, un mécanisme de gestion à priori des circuits potentiellement bloquants.

Le choix entre les deux solutions précédemment proposées pour la gestion des accès dépend essentiellement de la capacité totale du circuit en ES.

Nous allons détailler à titre d'illustration complémentaire le mécanisme du banquier appliqué au partage de deux ensembles de ressources.

V.2.4. Blocage sur allocations croisées

Considérons l'exemple qui est repris dans plusieurs ouvrages de référence (CRO 75, BEA 90). Il consiste à allouer successivement deux ressources R1 et R2 à deux processus P et P' mais selon deux séquences inversées. Dans la mesure où le concepteur a toutes libertés pour la spécification de ses protocoles d'accès, il est possible qu'il se place dans une telle configuration.

Les protocoles sont les suivants :

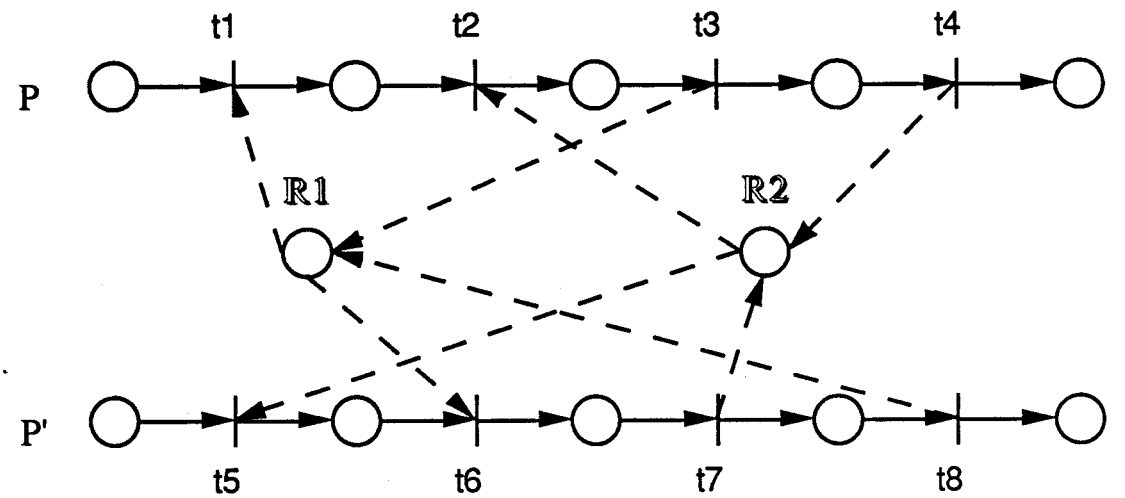


Figure 25

R1 a une capacité de 1 et R2 de 3.

Il s'agit d'un cas ne présentant pas de circuit qui ne mette en jeu que des places de gestion, mais il est bloquant. Nous cherchons à mettre en place un mécanisme inspirée de l'approche proposée dans le paragraphe précédent. Il s'avère qu'il existe également un circuit, mais qu'il fait intervenir des places des deux gammes opératoires.

Le blocage se produit lorsqu'une pièce P s'est allouée le moyen R1 et attend R2, et trois pièces P' se sont allouées R2 et attendent R1.

Une place Accès est ajoutée au réseau. Le mécanisme doit garantir le fait suivant :

- lorsqu'une pièce P s'alloue R1, on doit garantir qu'il subsistera un ES de R2 tant que P ne se l'est pas allouée,

- lorsqu'une pièce P' s'alloue R2, on doit garantir qu'il subsistera un ES de R1 tant que P' ne se l'est pas allouée.

La place Accès doit donc perdre un jeton lorsque la pièce P transite par t1

(le jeton consommé traduit la diminution de la capacité en pièce à destination de R2), il s'agit en quelque sorte d'une préallocation d'un emplacement de R2.

La place Accès gagne un jeton lorsque la pièce P transite par t2 : R2 est allouée en t2,
de même pour P' : la place Accès est reliée à t5 et t6.

Le marquage initial de la place Accès est la pondération cumulée de R1 et R2 moins un, soit 3 jetons. De la sorte, on peut garantir que pour chacune des 3 pièces présentes dans les places potentiellement bloquantes, il existe un exemplaire de la ressource manquante à au moins l'une des trois.

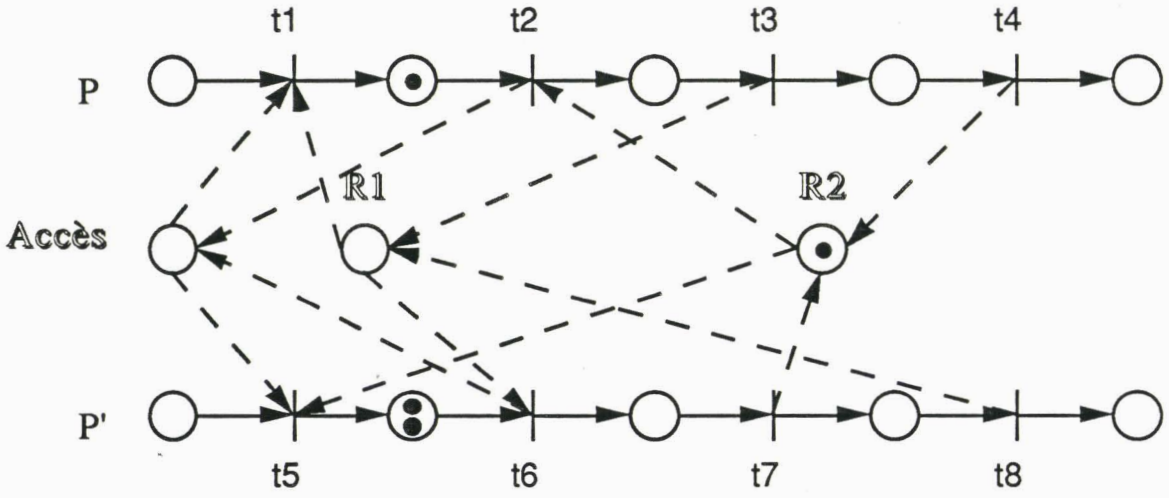


Figure 26

Le marquage de la Figure 26 traduit le fait que :

- les 2 pièces de P' attendent R1,

- aucune pièce P' ne peut s'allouer R2,

- la pièce de P peut s'allouer l'ES restant de R2.

Le marquage suivant :

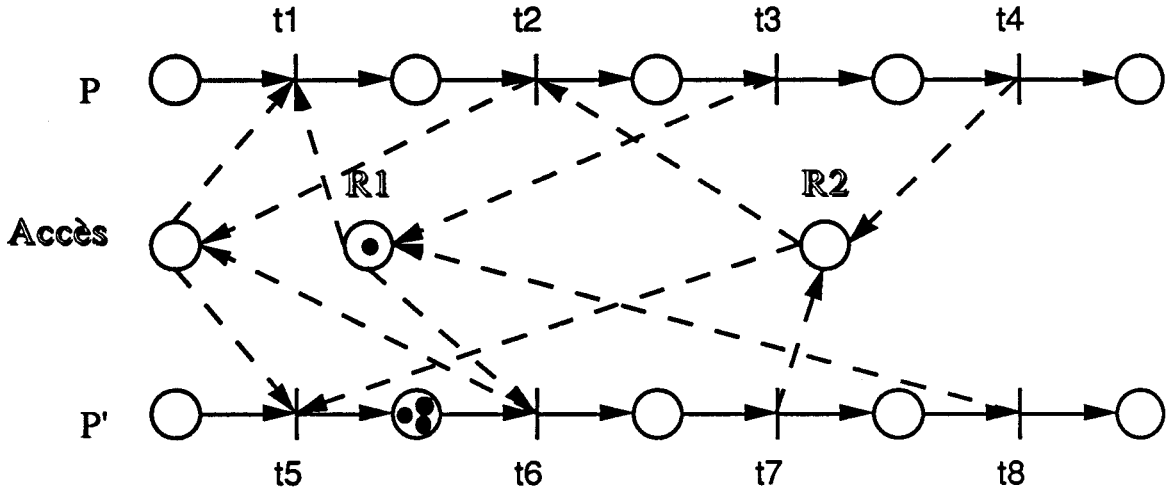


Figure 27

traduit le fait que :

- aucune pièce P ne peut s'allouer R1,

- chacune des 3 pièces P' peut potentiellement utiliser R1.

Ce mécanisme permet de garantir que les 2 places bloquantes ne seront jamais saturées simultanément. Il est simple à mettre en œuvre dans le cas général.

Il ne s'applique qu'aux allocations croisées, soit aux réseaux dans lesquels deux ressources sont allouées successivement à deux processus mais en sens inverse. Il faut de plus que la première ressource vis à vis d'un processus ne soit pas libérée tant que la seconde est allouée.

Le cas d'allocations croisées concernant un nombre plus important de ressources est une extension qui peut être prise en compte.

Nous pouvons remarquer de plus que la recherche des allocations croisées est similaire à la recherche des circuits (§ V.2.3.), mis à part que dans la recherche des allocations croisées, nous asseyons d'identifier des circuits qui passent par des places des gammes opératoires en plus des places de gestion de moyens.

Nous avons présenté en détail des mécanismes de gestion d'accès qui sont simples à mettre en œuvre. Ils permettent de traiter la majorité des situations bloquantes dans le domaine de la production flexible manufacturière discrète.

Les deux techniques présentées s'inscrivent dans le cadre plus général de la recherche de verrous dans les RdP.

Les résultats présentés dans ce mémoire ne sont qu'une esquisse du problème, basée sur l'expérience. Les méthodes de recherche de verrous et de trappes telles que celle présentée dans ALA 85 constituent des axes de recherche à part entière, et devront être intégrées de manière rigoureuse dans la démarche CASPAIM-2.

Nous détaillons leur identification sur la base de l'exemple proposé dans ce mémoire au chapitre III.

V.2.5. Mise en œuvre : contraintes d'accès, contrôle des flux de palettes

L'identification des configurations bloquantes est tout d'abord menée sur la première étape de la génération des gammes opératoires présentée au chapitre IV. Le procédé est donc assimilé à l'ensemble des lieux suivants :

Stockage, Robot-1, Convoyeur, Tour, Robot-2, Mach Ass, Taraudeuse.

La troisième étape, dont la gamme opératoire est présentée au chapitre IV Fig 78, est issue du troisième niveau de la description du procédé, soit l'ensemble des lieux :

Stockage, Robot-1.Tronc.Bras, Convoyeur.Rail-i.Palette-j, Tour.Mandrin.Machoires, Robot-2.Pince-i, Mach Ass et Taraudeuse.

Ce modèle sera utilisé comme second exemple de mise en œuvre.

Nous verrons dans ce cadre que le flux des palettes, dont la gestion est assurée par l'Interface, doit être contrôlé à partir de la troisième étape et les suivantes.

Reprenons le modèle des gammes opératoires et des protocoles d'accès présentés en début de chapitre. Les transitions ont été numérotées afin de repérer les circuits et les allocations croisées.

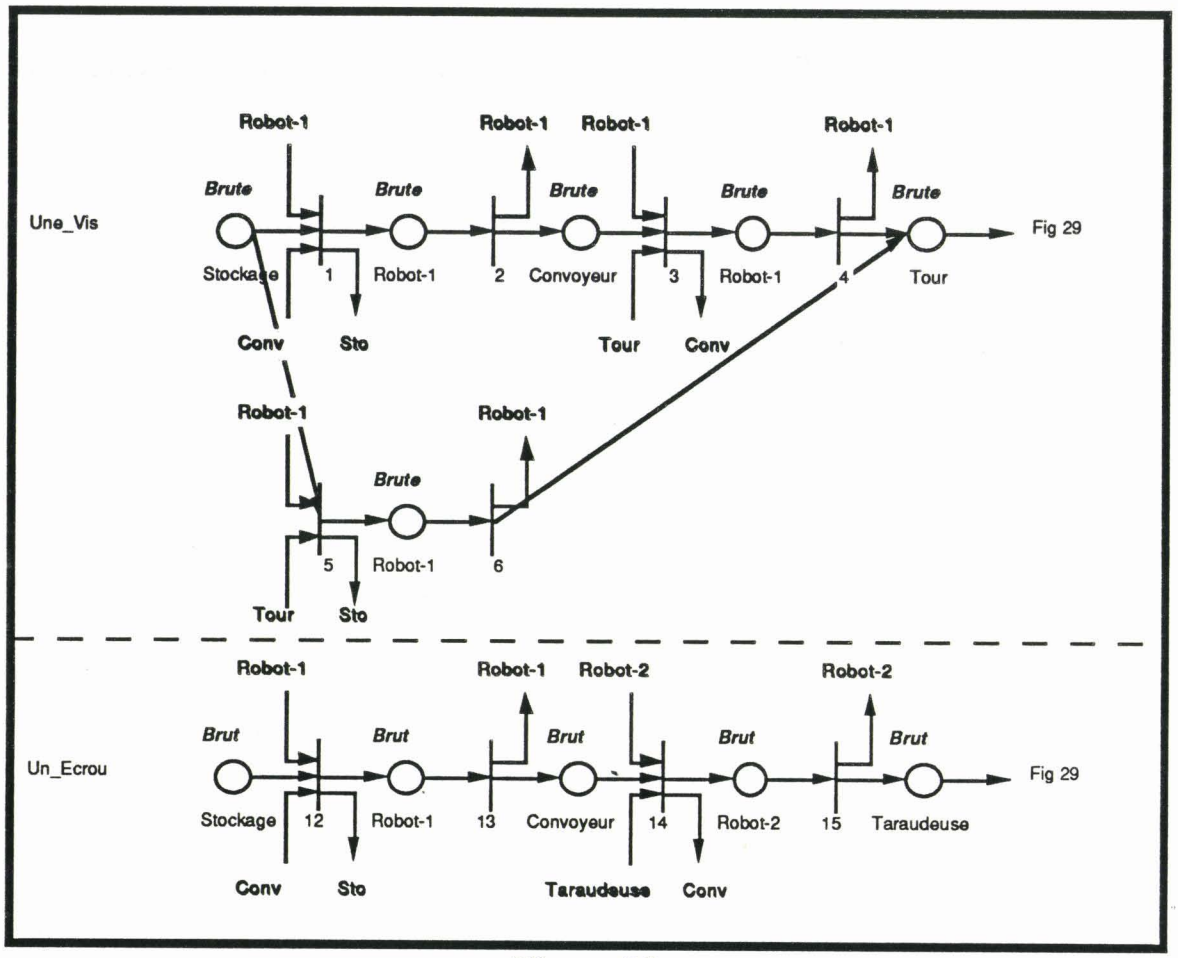

Une_Vis
Robot-1
Brute
Stockage
1
Robot-1
Brute
Robot-1
2
Convoyeur
Brute
Robot-1
Robot-1
3
Robot-1
Brute
Robot-1
4
Tour
Brute
Fig 29
Conv
Sto
Tour
Conv
Robot-1
Brute
5
Robot-1
Robot-1
6
Tour
Sto
Un_Ecrou
Robot-1
Brut
Stockage
12
Robot-1
Brut
Robot-1
13
Convoyeur
Brut
Robot-2
Robot-1
14
Robot-2
Brut
Robot-2
15
Taraudeuse
Brut
Fig 29
Conv
Sto
Taraudeuse
Conv

Figure 28

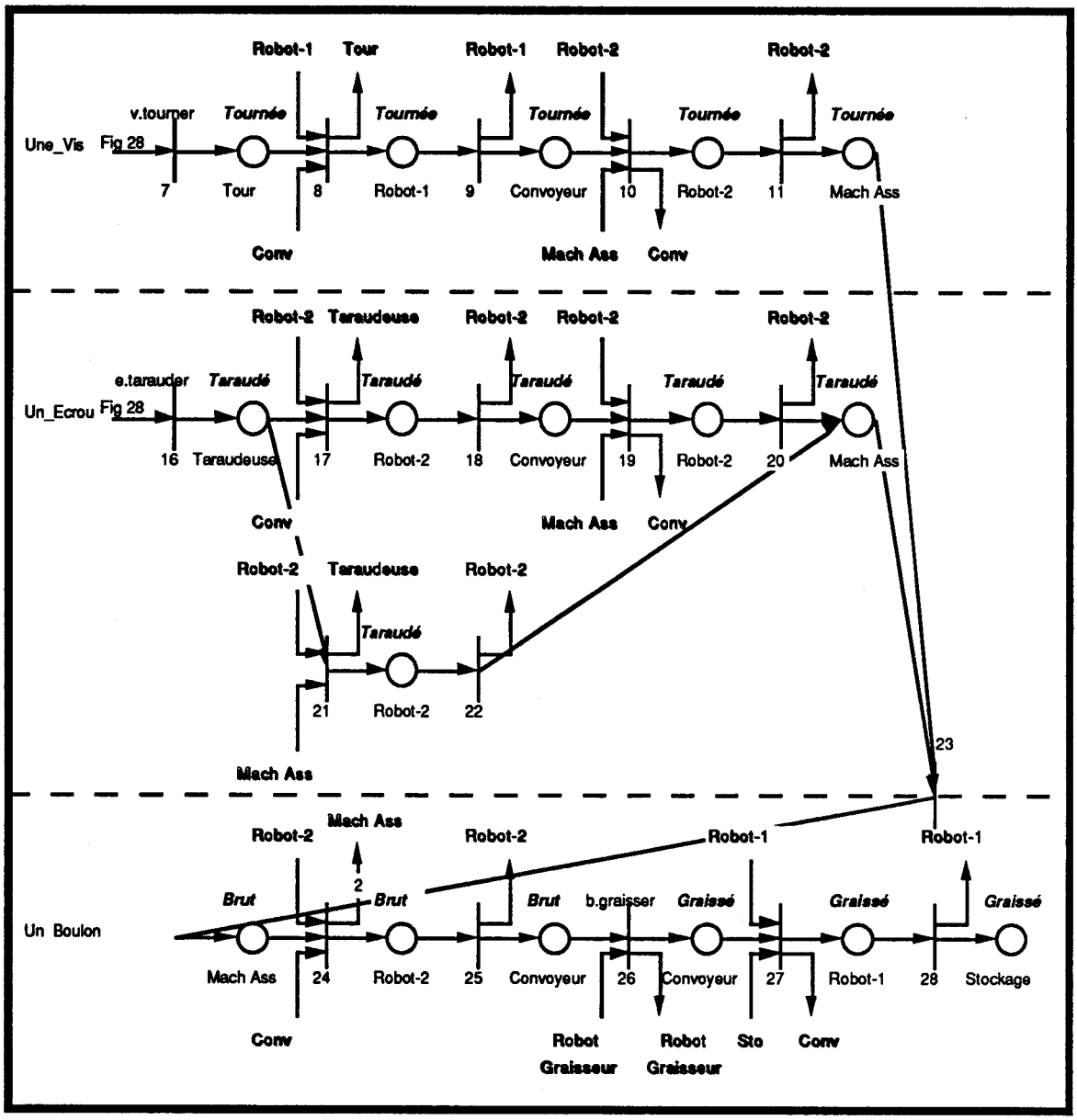


Figure 29

Recherche des circuits

La recherche des circuits passant par les places de gestion de ressources et les transitions de ce réseau a été informatisée au moyen d'un algorithme récursif, et a permis d'en trouver neuf :

1. t3 → Conv → t8 → Tour

2. t10 → Conv → t24 → Mach Ass

3. t14 → Conv → t17 → Taraudeuse

4. t14 → Conv → t24 → Mach Ass → t21 → Taraudeuse

5. t19 → Conv → t24 → Mach Ass

6. t26 → Graisseur

7. t1 → Sto → t27 → Conv

8. t12 → Sto → t27 → Conv

9. t8 → Tour → t5 → Sto → t27 → Conv

Le circuit 6 ne fait intervenir que le moyen graisseur. Dans la mesure où il n'est pas partagé par plusieurs gammes opératoires, il ne produit pas de blocage.

Les huit autres circuits sont potentiellement bloquants. Puisque le convoyeur apparait dans chacun des huit circuits, et qu'il a une capacité importante en ES, la solution simplifiée est satisfaisante pour chacun de ces huit circuits. En effet, le fait qu'un ES reste vacant à tout instant sur l'ensemble des ES des palettes du convoyeur ne modifie que très peu les performances globales du système de production.

Une place RdP est donc ajoutée pour chaque circuit. Cette place permet de garantir que le circuit ne sera jamais vide de jetons. Sa capacité vaut la somme des capacités des places de gestion moins 1. Cette place est reliée à toutes les transitions du réseau qui modifie la somme des jetons de l'ensemble des places de gestion.

Le circuit 1 concerne les lieux Convoyeur et Tour. La place de gestion d'accès, que nous appelons Accès Conv-Tour, est reliée

en amont des transitions t1, t5, t12, t17, t24,

en aval des transitions t10, t14, t19, t27.

Elle contient Conv + Tour -1 jetons.

Le tableau suivant dresse le bilan des circuits ainsi que la liste des transitions amont et aval des places de gestion d'accès et leur marquage initial.

Circuit	Nom place de gestion	Marquage initial	en amont des transitions	en aval des transitions
1	Accès Conv-Tour	Conv + Tour -1	t1, t5, t12, t17, t24	t10, t14, t19, t27
2	Accès Conv-Mach Ass	Conv+Mach Ass -1	t1, t8, t12, t17, t21	t3, t14, t24, t27
3	Accès Conv-Taraudeuse	Conv + Taraudeuse -1	t1, t8, t12, t24	t3, t10, t19, t21, t27
4	inutile			
5	id circuit 2			
6	inutile			
7	Accès Conv-Sto	Conv + Sto -1	t8,t17, t24, + autres de Sto	t3, t10, t14, t19, + autres de Sto
8	id circuit 7			
9	inutile			

Figure 30

Il est inutile de gérer le circuit 4 puisqu'il contient Conv et Mach Ass qui sont gérés par le circuit 2, ainsi que Conv et Taraudeuse qui est géré par le circuit 3.

Le circuit 5 est identique au circuit 2.

Le circuit 8 est identique au circuit 7.

Le circuit 9 contient Conv et Sto qui sont gérés par le circuit 7 donc il est inutile de le gérer.

La transition t24 apparait dans la liste des transitions amont de la place Accès Conv-Mach Ass puisqu'elle consomme 1 jeton de la place Conv et restitue 2 jetons dans la place Mach Ass. t24 modifie donc le marquage global de Conv et Mach Ass, bien qu'elle fasse partie du circuit 2.

Identifions les blocages par allocations croisées.

Recherche des allocations croisées

Cette recherche se décompose en 2 étapes :

- identification des séquences de transitions allouant pour chaque gamme successivement 2 ressources, sans que la première ne soit libérée :

Conv en t1 puis Tour en t3,

Tour en t3 puis Conv en t8,

Tour en t5 puis Conv en t8,

Conv en t8 puis Robot-2 en t10,

Conv en t8 puis Mach Ass en t10,

Mach Ass en t10 puis Conv en t24,

Conv en t12 puis Taraudeuse en t14,

Taraudeuse en t14 puis Conv en t17,

Conv en t17 puis Mach Ass en t19,

Mach Ass en t19 puis Conv en t24,

Mach Ass en t21 puis Conv en t24,

Conv en t24 puis Robot Graisseur en t26.

- puis recherche des allocations croisées proprement dites :

Première ressource	Seconde ressource	Allo 1ère ressource	Alloc seconde ressource	Alloc seconde ressource	Allo 1ère ressource
Conv	Tour	t1	t3	t3	t8
Conv	Tour	t1	t3	t5	t8
Conv	Mach Ass	t8	t10	t10	t24
Conv	Mach Ass	t8	t10	t19	t24
Conv	Mach Ass	t8	t10	t21	t24
Mach Ass	Conv	t10	t24	t17	t19
Conv	Taraudeuse	t12	t14	t14	t17
Conv	Mach Ass	t17	t19	t19	t24
Conv	Mach Ass	t17	t19	t21	t24

Figure 31

Nous remarquons d'après ce tableau que, dans la mesure où les circuits identifiés précédemment ne sont pas saturés, aucune allocation croisée ne sera bloquante. Il est donc inutile, dans ce cas précis, de les gérer explicitement.

En définitive, du point de vue des gammes opératoires, il suffit de garantir la non saturation des circuits suivants :

Convoyeur, Tour

Convoyeur, Machine d'Assemblage

Convoyeur, Taraudeuse

Convoyeur, Stockage.

Gestion des palettes

Les gammes opératoires de la troisième étape font apparaître les palettes en tant que support des pièces. La gestion des allocations des ES des palettes est assurée au moyen de la place de gestion d'accès. Cependant, le trajet effectif des palettes sur leur support (les rails) ne figure pas dans la PC. Chaque palette est gérée dans le module Interface. Le modèle des états atteints par chaque palette traduit les lieux par lesquels elle transite.

Nous supposons que les palettes sont utilisables en permanence. Aucune maintenance n'est envisagée, bien que cette gestion puisse être prise en compte. Nous pouvons ainsi définir l'équivalent d'une gamme opératoire, dans laquelle l'objet manipulé serait une palette. Le graphe d'état obtenu est bien cyclique. Les phases de validation (quasi vivacité et terminaison propre) n'ont plus de sens. La propriété de vivacité du réseau est cependant à vérifier, et permet de garantir que chaque palette peut évoluer librement sur le graphe.

Les palettes évoluent dans le réseau suivant :

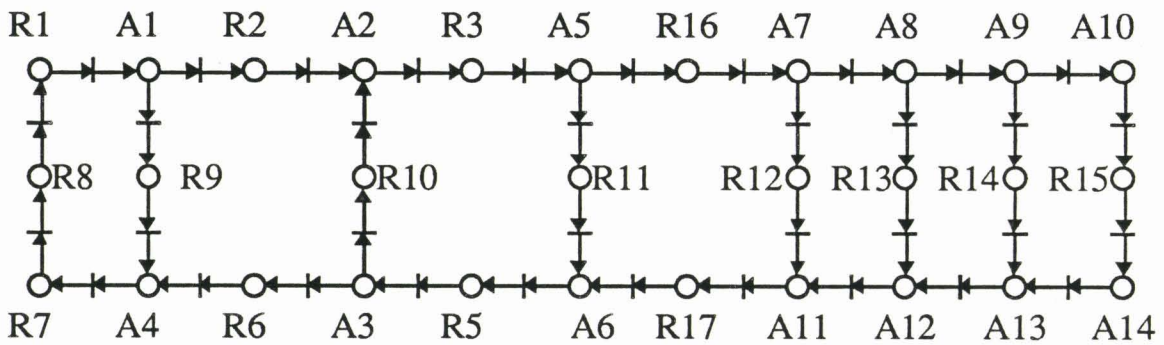


Figure 32

Nous choisissons de spécifier les protocoles d'accès sous la forme d'accès en temps masqué pour chaque tronçon de rail à capacité finie. Ils sont donc de la forme suivante (seuls quelques accès ont été explicités, les autres se présentant sous le même format) :

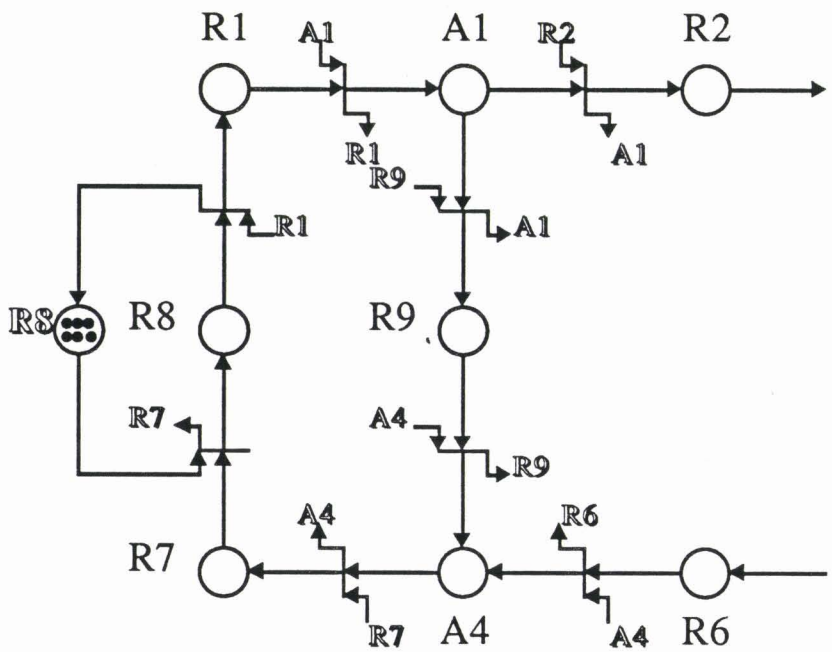


Figure 33

La recherche des configurations bloquantes se limite à la recherche des

circuits potentiellement bloquants. En effet, puisque tous les accès se font en temps masqué, les allocations croisées coïncident avec les circuits.

Le nombre de circuits indépendants du réseau de la Figure 32 est égal au nombre cyclomatique du graphe biparti G qui lui est associé (GON 79), soit

Nombre de circuits indépendants = M - N + p

avec **M** = nombre d'arcs du graphe G,
N = nombre de sommets du graphe G,
p = nombre de composantes connexes.

Du fait de la forte connexité du RdP de la Figure 32, le graphe sous-jacent a une seule composante connexe.

D'après la topologie de RdP (graphe d'état), il est équivalent de calculer le nombre cyclomatique du graphe G' dont

- un sommet est un aiguillage de la Figure 32,

- un arc est associé à l'ensemble des transitions et places reliant deux aiguillages.

Le graphe G' est donc le suivant :

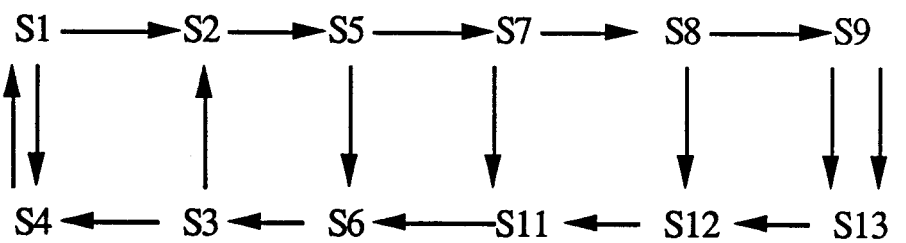


Figure 33

Il y a donc 18-12+1 = 7 circuits indépendants dans le RdP, qui sont les suivants :

R8,R1,A1,R9,A4,R7

R8,R1,A1,R2,A2,R3,A5,R11,A6,R5,A3,R6,A4,R7

R10,A2,R3,A5,R11,A6,R5,A3

R10,A2,R3,A5,R16,A7,R12,A11,R17,A6,R5,A3

R10,A2,R3,A5,R16,A7,A8,R13,A12,A11,R17,A6,R5,A3

R10,A2,R3,A5,R16,A7,A8,A9,R14,A13,A12,A11,R17,A6,R5,A3

R10,A2,R3,A5,R16,A7,A8,A9,A10,R15,A14,A13,A12,A11,R17,A6,R5,A3

Le contrôle d'accès à chacun de ces circuits est mis en œuvre selon les modalités présentées au paragraphe précédent par la solution simplifiée. Une place de gestion d'accès est associée à chaque circuit identifié. Le marquage initial d'une place d'accès vaut la capacité cumulée de toutes les places du circuit auquel elle est associée moins un.

Pour conclure, nous pouvons remarquer que ces considérations sur les configurations bloquantes permettent de proposer des choix pertinents quant au nombre de palettes à mettre en circulation.

Il suffit en effet que le nombre total de palettes en circulation soit inférieur strictement à la capacité totale du plus petit circuit (par exemple R1, A1, R9, A4, R7, R8) pour qu'aucun des circuits ne puisse être saturé. On se dispense ainsi de la gestion a priori des blocages par un choix judicieux du nombre de palettes en circulation, bien que les performances réduites en terme de débit de palettes puissent interdire une telle solution.

CONCLUSION

Ce dernier chapitre nous a permis de mettre en évidence que, sans contraindre a priori le concepteur sur les choix de protocoles d'accès aux ressources, ce dernier peut assez facilement être assisté dans ses choix par deux outils complémentaires permettant, dans le contexte spécifique de notre étude, d'extraire les situations bloquantes. Il est alors possible, plutôt que de contraindre le concepteur à imaginer d'autres solutions moins satisfaisantes, de compléter le graphe au sens des deux méthodes proposées ici.

Dans ce dernier chapitre, nous avons présenté deux techniques de gestion a priori des blocages. La mise en œuvre dans le cadre de la productique de ces techniques permet de couvrir la majorité des situations bloquantes rencontrées à ce jour.

Nous proposons d'appliquer ces techniques au niveau de la Partie Commande (point de vue des pièces) et également au niveau du module Interface (point de vue des supports mobiles des pièces).

Elles introduisent des indéterminismes supplémentaires dont la gestion par le Niveau Hiérarchique vient s'ajouter aux indéterminismes dus aux gammes opératoires munies des protocoles d'accès.

Elles garantissent cependant un comportement non bloquant du système dès la phase de conception. Les performances devront cependant être évaluées de façon complémentaire.

Notons au passage que notre solution nécessite une allocation centralisée de toutes les ressources.

L'analyse des réseaux obtenus par notre démarche (blocages), constitue un axe prospectif important.

Par manque de temps, nous n'avons qu'esquissé deux techniques de prévention des blocages, qui viendront en complément de recherches ultérieures.

CONCLUSION GENERALE

Nous avons présenté dans ce mémoire une extension de la méthodologie CASPAIM dont le principal objet est l'assistance à la conception de la commande d'ateliers flexibles complexes.

Cette conception consiste à réaliser ou à garantir l'adéquation entre deux points de vue complémentaires :

- les fonctionnalités que doit remplir l'unité de production vis à vis de la production souhaitée (gammes de pièces, quantités, délais),

- et les moyens de production de la cellule considérée.

Dans l'esprit d'une véritable gestion de projet de conception de la commande, il nous est apparu indispensable de mener en parallèle :

- la conception et la validation d'un modèle Réseaux de Petri décrivant la suite des opérations que doivent subir chacun des objets manufacturés, que nous avons appelé gamme logique,

- la spécification hiérarchisée des moyens de production.

Puis nous avons présenté une méthode originale et pragmatique dont le but est d'assister et de contrôler systématiquement la phase délicate qui consiste à réaliser l'adéquation entre les fonctionnalités souhaitées et les moyens disponibles. Dans l'élaboration de cette méthode, nous avons tenu à privilégier l'aspect gestion de projet afin de faciliter considérablement la gestion de l'évaluation de multiples variantes de conception.

Il ne s'agit en aucun cas de contraindre le concepteur à limiter ses choix par rapport au potentiel de l'atelier flexible, mais au contraire d'expliciter cette potentialité afin qu'il réalise son choix de conception. Il est clair que la difficulté de la méthode se situait dans l'identification de cette potentialité.

Le modèle RdP obtenu est enfin contraint afin de prendre en compte la capacité des ressources du système. C'est ce modèle qui est évalué (blocages, performances) pour conduire ensuite à la phase finale d'implantation.

Le modèle de conception est construit initialement dans l'hypothèse d'un parallélisme d'exécution maximal, puis est contraint peu à peu, à mesure que les moyens opérationnels sont pris en compte.

Les principaux axes de recherche prospectifs qui apparaissent dans la suite logique de notre travail concernent :

(i) la mise en œuvre des deux systèmes décisionnels découplés du Niveau Hiérarchique : au niveau du suivi de pièces d'une part, et au niveau de la gestion temps réel des moyens de production d'autre part (comme par exemple la gestion du système de transport),

(ii) la méthode de mise en œuvre du module Interface,

(iii) la gestion de bibliothèques de gammes prédéfinies, ou de moyens prédéfinis,

(iv) la phase ultérieure de spécification et de conception de l'architecture informatique qui sert de support à l'implantation du graphe de commande.

Le travail présenté dans ce mémoire constitue en résumé une base de spécification d'un outil informatisé d'assistance à la conception du contrôle réparti et hiérarchisé d'unités flexibles de production dans l'industrie manufacturière.

Sans remettre en question les résultats antérieurs proposés par le LAII qui restent valables dans le contexte d'une large gamme d'applications, nous avons ainsi contribué à résoudre le problème posé par la conception de systèmes de production de grande complexité qui tendent dès à présent à apparaître dans les grandes Entreprises Manufacturières.

REFERENCES BIBLIOGRAPHIQUES

BOU 84 A. BOURJAULT

Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires.

Thèse d'état, Université de Franche-Comté, Besançon, 1984

BOU 88 J.P. BOUREY

Structuration de la partie procédurale du système de commande de cellules flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

BOU 90a J.P. BOURRIERES

Contribution à la modélisation intégrée des systèmes robotisés d'assemblage.

Thèse d'état, Université de Franche-Comté, Besançon, 1990

BOU 90b A. BOURJAULT, J.M. HENRIOUD

Conception des systèmes flexibles d'assemblage.

CIM Productica 90, p 497, Bordeaux

BOU 90c J.P. BOURRIERES, F. LHOTE

Modélisation hiérarchisée des tâches d'assemblage.

CIM Productica 90, p 505, Bordeaux

CAM 89 J.P. CAMPAGNE, G. CAPLAT

Elaboration automatique de gammes d'assemblage.

A.P.I.I. Vol 23, N°1, 1989

CAS 87 E. CASTELAIN

Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1987, Lille

CHA 88 D. CHAPPE, A. BOURJAULT

Une méthodologie de détermination de l'organisation d'un atelier d'assemblage.

Congrès AFCET Automatique, p 147, Oct 88, Grenoble

CRA 89 E. CRAYE

De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1989, Lille

KAP 88 M. KAPUSTA

Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

Chapitre I

BOU 87 J.P. BOUREY, D. CORBEEL, E. CRAYE, J.C. GENTINA

Utilisation des réseaux de Petri structurés adaptatifs colorés dans l'analyse et la synthèse du contrôle hiérarchisé de processus discontinus. Partie A : les modèles de description.

A.P.I.I. Vol 21, N° 4, 1987

BOU 88a J.P. BOUREY

Structuration de la partie procédurale du système de commande de cellules flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

BOU 88b J.P. BOUREY, J.C. GENTINA

Structuration de la partie procédurale du système de commande de cellules flexibles de production

Congrès AFCET Automatique, p 233, Oct 88, Grenoble

CAS 87 E. CASTELAIN

Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1987, Lille

CRA 89 E. CRAYE

De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1989, Lille

GEN 87 J.C. GENTINA, D. CORBEEL

Coloured Adaptative Structured Petri-Net : A tool for automatic synthesis of hierarchical control of flexible manufacturing systems.

IEEE International Conference on Robotics and Automation March,31/April,3, 1987, Raleigh, North Carolina

KAP 88 M. KAPUSTA

Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

BEA 90 J. BEAUQUIER, B. BERARD

Systèmes d'exploitation : Concepts et algorithmes

Mac Graw-Hill, Paris 1990

BOO 88 G. BOOCH

Ingénierie du logiciel avec ADA

Addison-Wesley Europe. InterEditions, Paris, 1988

BOU 88 J.P. BOUREY

Structuration de la partie procédurale du système de commande de cellules flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

CRA 89 E. CRAYE

De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1989, Lille

CRU 90a D. CRUETTE, J.P. BOUREY, J.C. GENTINA

Method of structural and functional analysis of complex processes, and aided design of command under constraints in the manufacturing.

MIM $S^{2'}$ 90 IMACS-IFAC International Symposiun on Mathematical and Intelligent Models in System Simulation. Session IV.C.1. Brussels, September 3-7, 1990.

CRU 90b D. CRUETTE, J.P. BOUREY, J.C. GENTINA

A hierarchical specification and validation of the operating sequences in the flexible manufacturing systems context.

A paraître en novembre 90 dans la revue Computer-Integrated Manufacturing Systems, Ed Butterworth

KAP 88 M. KAPUSTA

Génération assistée d'un graphe fonctionnel destiné à

l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

LHO 88 P. LHOSTE, J.M. TIXADOR, G. MOREL, M. ROESCH

Outils de Conception de Systèmes Automatisés de Production.

Congrès AFCET Automatique, p 331, Oct 88, Grenoble

MOR 88 G. MOREL, M. ROESCH, M. VERON

Génie Productique, Génie-X

Congrès AFCET Automatique, p 319, Oct 88, Grenoble

ROS 75 D.T. ROSS, J.B. GOODENOUGH, C.A. IRVINE

Software Engineering : Process, Principles, and Goals

Computer : 65, Mai 1975

TIX 89 J.M. TIXADOR

Une contribution au génie automatique : la spécification, exécutable des machines et systèmes automatisés de production.

Thèse de Docteur Ingénieur, Université de Nancy I, 1989

Chapitre III

ALA 90 J.T. ALANDER, M. FRISK, J. TUOMINEN,M. VUOLTEENAHO

An outline of a modular and multiarm assembly cell with object-oriented control.

IECON '90. 16th Annual Conference of IEEE Industrial Electronics Society. Nov 27-30, 1990, Pacific Grove, California

AMA 90 S. AMAR, E. CASTELAIN, J.C. GENTINA

Modélisation des moyens de production par langages orientés objet en vue de la conception de la commande d'un système de production flexible.

CIM Productica 90, p 323, Bordeaux

CRU 90 D. CRUETTE, J.P. BOUREY, J.C. GENTINA

Method of structural and functional analysis of complex processes, and aided design of command under constraints in the manufacturing.

MIM $S^{2'}$ 90 IMACS-IFAC International Symposiun on Mathematical and Intelligent Models in System Simulation. Session IV.C.1. Brussels, September 3-7, 1990.

LHO 88 P. LHOSTE, J.M. TIXADOR, G. MOREL, M. ROESCH

Outils de Conception de Systèmes Automatisés de Production.

Congrès AFCET Automatique, p 331, Oct 88, Grenoble

MOR 88 G. MOREL, M. ROESCH, M. VERON

Génie Productique, Génie-X

Congrès AFCET Automatique, p 319, Oct 88, Grenoble

TIX 89 J.M. TIXADOR

Une contribution au génie automatique : la spécification, exécutable des machines et systèmes automatisés de production.

Thèse de Docteur Ingénieur, Université de Nancy I, 1989

Chapitre IV

BER 83 G. BERTHELOT

Transformation et analyse de réseaux de Petri : Application aux protocoles.

Thèse d'Etat, Université de Paris VI, 1983

BER 85 G. BERTHELOT

Transformations de réseau de Petri

T.S.I. Vol 4, N°1, 1985

BOU 84 A. BOURJAULT

Contribution à une approche méthodologique de l'assemblage automatisé : élaboration automatique des séquences opératoires.

Thèse d'état, Université de Franche-Comté, Besançon, 1984

BOU 87a J.P. BOUREY, D. CORBEEL, E. CRAYE, J.C. GENTINA

Utilisation des réseaux de Petri structurés adaptatifs colorés dans l'analyse et la synthèse du contrôle hiérarchisé de processus discontinus. Partie A : les modèles de description.

A.P.I.I. Vol 21, N° 4, 1987

BOU 87b A. BOURJAULT, D. CHAPPE, J.M. HENRIOUD

Elaboration automatique des gammes d'assemblage à l'aide de réseaux de Petri.

A.P.I.I. Vol 21, N°4, 1987

BOU 88a J.P. BOUREY

Structuration de la partie procédurale du système de commande de cellules flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

BOU 88b J.P. BOUREY, J.C. GENTINA

Structuration de la partie procédurale du système de commande de cellules flexibles de production

Congrès AFCET Automatique, p 233, Oct 88, Grenoble

BOU 90a J.P. BOURRIERES

Contribution à la modélisation intégrée des systèmes robotisés d'assemblage.

Thèse d'état, Université de Franche-Comté, Besançon, 1990

BOU 90b A. BOURJAULT, J.M. HENRIOUD

Conception des systèmes flexibles d'assemblage.

CIM Productica 90, p 497, Bordeaux

BOU 90c J.P. BOURRIERES, F. LHOTE

Modélisation hiérarchisée des tâches d'assemblage.

CIM Productica 90, p 505, Bordeaux

BRA 83 BRAMS

Réseaux de Petri : Théorie et pratique. Tome 1. Théorie et analyse

Masson, 1983

CAM 89 J.P. CAMPAGNE, G. CAPLAT

Elaboration automatique de gammes d'assemblage.

A.P.I.I. Vol 23, N°1, 1989

CAS 87 E. CASTELAIN, D. CORBEEL, J.C. GENTINA

Partie B. Apports de l'Intelligence Artificielle dans la réalisation d'un simulateur de Réseaux de Petri adaptatifs colorés : application à la conception et à la validation de la commande des processus industriels.

A.P.I.I. Vol 21, N°4, 1987

CHA 88 D. CHAPPE, A. BOURJAULT

Une méthodologie de détermination de l'organisation d'un atelier d'assemblage.

Congrès AFCET Automatique, p 147, Oct 88, Grenoble

CRA 89 E. CRAYE

De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1989, Lille

EST 85 P. ESTEBAN

Sur la recherche d'algorithmes simplifiés d'analyse des Réseaux de Petri.

Thèse de Docteur Ingénieur, Université P. Sabatier, Toulouse, 1985

EST 87 P. ESTEBAN

Algorithmes simplifiés d'analyse des RdP.

A.P.I.I. Vol 21, N°6, 1987

GEN 87 J.C. GENTINA, D. CORBEEL

Coloured Adaptative Structured Petri-Net : A tool for automatic synthesis of hierarchical control of flexible manufacturing systems.

IEEE International Conference on Robotics and Automation March,31/April,3, 1987, Raleigh, North Carolina

GON 79 GONDRAN, MINOUX

Graphes et algorithmes, Ed Eyrolles, 1979

HEN 90 J.M. HENRIOUD, A. BOURJAULT

Détermination des arbres d'assemblage

A.P.I.I. Vol 24, N°6, 1990

KAP 88 M. KAPUSTA

Génération assistée d'un graphe fonctionnel destiné à l'élaboration structurée du modèle de la partie commande pour les cellules de production flexibles dans l'industrie manufacturière.

Thèse de Doctorat d'Université de USTLFA, 1988, Lille

SIB 85 SIBERTIN-BLANC

High Level Petri Nets with data structure

6th Eutopean Workshop on Applications and Theory of Petri Nets. Helsinki, Finland 1985

SIB 88 SIBERTIN-BLANC

Le prototypage des applications interactives à l'aide de réseaux de Petri

Séminaire d'Informatique de l'I.I.E., Réseaux de Petri, 26-27 Oct 1988

SIB 90 SIBERTIN-BLANC

L'approche à objets pour la surveillance et la mise en œuvre des SED

Réunion S.E.D., GT2, CNRS du 11 janvier 1990

SIB 91 SIBERTIN-BLANC

Object Oriented Structuring for High Level Petri Nets.

A paraître dans Advance in PN 91, LNCS, Springer en collaboration avec R. BASTIDE

TAR 72 TARJAN

Depth-first Search and Linear graph algorithms, Siam J. Computing 1, pp 253-268, Tarjan 1972

Chapitre V

AGA 87 S. AGALOUA, P. LADET

Une méthode pour l'analyse des Réseaux de Petri. Analyse de la pseudo-vivacité des Réseaux de Petri par résolution d'un système linéaire.

A.P.I.I. Vol 21, N°6, 1987

ALA 85 H. ALAIWAN, J.M. TOUDIC

Recherche des semi-flots, des verrous et des trappes dans les réseaux de Petri

T.S.I. Vol 4, N°1, 1985

BEA 90 J. BEAUQUIER, B. BERARD

Systèmes d'exploitation : Concepts et algorithmes

Mac Graw-Hill, Paris 1990

BER 83 G. BERTHELOT

Transformation et analyse de réseaux de Petri : Application aux protocoles.

Thèse d'Etat, Université de Paris VI, 1983

BER 85 G. BERTHELOT

Transformations de réseau de Petri

T.S.I. Vol 4, N°1, 1985

BOU 90 J.P. BOURRIERES

Contribution à la modélisation intégrée des systèmes robotisés d'assemblage.

Thèse d'état, Université de Franche-Comté, Besançon, 1990

BRA 83 BRAMS

Réseaux de Petri : Théorie et pratique. Tome 1. Théorie et analyse

Masson, 1983

COH 89 COHEN

Qu'est-ce-qu'un système à événements discrets ? Le point de vue du conférencier.

Réunion S.E.D., GT1, CNRS du 7 décembre 1989

CRO 75 CROCUS

Systèmes d'explotation des ordinateurs. Principes de conception.

Dunod, 1975

EST 85 P. ESTEBAN

Sur la recherche d'algorithmes simplifiés d'analyse des Réseaux de Petri.

Thèse de Docteur Ingénieur, Université P. Sabatier, Toulouse, 1985

EST 87 P. ESTEBAN

Algorithmes simplifiés d'analyse des RdP.

A.P.I.I. Vol 21, N°6, 1987

GON 79 GONDRAN, MINOUX

Graphes et algorithmes, Ed Eyrolles, 1979

RAY 85 M. RAYNAL

Algorithmes distribués et protocoles.

Eyrolles, 1985

ANNEXE

Quelques propriétés des réseaux de Petri

Quasi-vivacité :

Soit $<R ; M_0>$ un réseau marqué.

Une transition t de ce réseau est **quasi-vivante** si il existe un marquage accessible $M \in A(<R ; M_0>)$ tel que $M(t>$.

Le réseau marqué $<R ; M_0>$ est dit **quasi-vivant** si toutes ses transitions sont quasi-vivantes.

Vivacité :

Une transition t d'un réseau marqué $<R ; M_0>$ est **vivante** si pour tout marquage accessible $M \in A(<R ; M_0>)$ t est quasi-vivante pour le réseau $<R ; M>$.

Un réseau marqué $<R ; M_0>$ est vivant si toutes ses transitions sont vivantes.

Un réseau R est vivant si il existe un marquage M tel que $<R ; M>$ soit vivant.

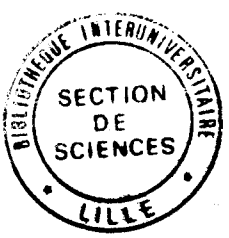
RESUME

La première phase de la conception d'une cellule flexible de production discontinue consiste à mener en parallèle la conception d'un modèle réseau de Petri décrivant la suite des opérations que doit subir chacun des produits manufacturés, et la modélisation hiérarchisée des moyens de production.

La seconde phase consiste à mettre en œuvre une méthode itérative permettant de réaliser progressivement l'adéquation entre les fonctionnalités souhaitées et les moyens disponibles, en privilégiant l'aspect gestion de projet, ainsi que l'évaluation de multiples variantes de conception.

Le modèle complet ainsi obtenu, dont le degré de parallélisme est maximal, est enfin contraint afin de traduire la concurrence d'accès aux ressources.

L'approche proposée est illustrée par un exemple de traitement de regroupements de pièces de dimension industrielle.



MOTS CLEFS

Méthodologie de conception
Gamme logique, opératoire
Systèmes de production flexible
Traitement de lot
Réseaux de Petri à Objets
Modélisation du Procédé
Modèle fonctionnel