# 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

Stéphane BOIS
Mastère I.D.N
Maître E.E.A

# INTEGRATION DE LA GESTION DES MODES DE MARCHE DANS LE PILOTAGE D'UN SYSTEME AUTOMATISE DE PRODUCTION

soutenue le 28 Novembre 1991 devant la Commission d'Examen

| | | |
|---|---|---|
| Membres du jury : | M. VERON | **Rapporteur** |
| | M. STAROSWIECKI | **Rapporteur** |
| | J.C. GENTINA | **Examinateur, Directeur de thèse** |
| | P. BORNE | **Examinateur** |
| | E. CRAYE | **Examinateur** |
| | E. CASTELAIN | **Examinateur** |

# AVANT-PROPOS

*Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Informatique Industrielle de l'Ecole Centrale de LILLE sous la direction scientifique de Monsieur GENTINA, Directeur de l'EC LILLE. Je tiens à le remercier vivement de m'avoir accueilli dans son laboratoire de recherche.*

*Je suis très reconnaissant à Monsieur VERON, Professeur à l'Université de NANCY I, et Monsieur STAROSWIECKI, Professeur à l'Université des Sciences et Techniques de LILLE I, de l'honneur qu'il me font en acceptant de juger ce travail et d'être les rapporteurs de cette thèse.*

*Je tiens également à remercier :*

*Monsieur BORNE, Professeur à l'EC LILLE,*
*Monsieur CRAYE, Maître de Conférence à l'EC LILLE,*
*Monsieur CASTELAIN, Maître de Conférence à l'EC LILLE, et Directeur de l'IGII,*

*pour l'honneur qu'ils me font en participant à mon Jury de Thèse.*

*Je voudrais ici remercier tous les membres du Laboratoire d'Automatique et d'Informatique Industrielle de l'EC LILLE, et en particulier Monsieur CRAYE pour l'aide précieuse qu'ils m'ont apportée sur le plan scientifique et sur le plan humain.*

*Enfin, je remercie très sincèrement les personnes qui ont assuré la reprographie de ce mémoire : Messieurs VANGREVENINGE et GERMON.*

# TABLE DES MATIERES

## CHAPITRE I
## Présentation générale

# CHAPITRE II
## Modélisation

# CHAPITRE III
## Elaboration du gestionnaire

# INTRODUCTION GENERALE

La construction d'ensembles automatisés flexibles représente actuellement un challenge industriel de premier ordre.

L'automatisation des processus continus étant très avancée, les techniques éprouvées de l'automatique permettent généralement leurs contrôles et leurs modélisations. Celles-ci sont cependant inadaptées dans le cadre de processus discrets, caractéristiques de l'industrie manufacturière. Cela implique l'emploi de nouveaux outils et méthodes de contrôles. Ajoutons, par ailleurs, que liée à la mutation industrielle engendrée par les progrès technologiques, l'automatisation de processus industriels est devenue plus complexe, tant au niveau de la conception qu'au niveau de la fabrication et de la production.

Le concept de Productique prend son sens par l'intégration des différents outils utilisés dans les fonctions d'automatisation. La structure du système de commande de l'usine du futur repose sur une modularité et une hiérarchisation des automatismes de pilotage correspondant au concept du CIM (Computer Integrated Manufacturing). L'ensemble du système repose sur la combinaison d'outils informatiques qui coordonnent l'ensemble des tâches. La notion d'atelier flexible en est l'aboutissement concret.

La conception de systèmes de commande suscite de nombreux problèmes de recherches. De nombreux travaux sont menés, dans ce sens, au Laboratoire d'Automatique et d'Informatique Industrielle de Lille : **le projet CASPAIM** (Conception Assistée de Systèmes de Production Automatisés pour l'Industrie Manufacturière) vise à élaborer le système de pilotage depuis sa spécification jusqu'à son implantation.

Les fonctions de supervision de la commande et de la surveillance sont naturellement devenues plus complexes à gérer. Les premiers systèmes de supervision étaient en général limités à la mise en forme de valeurs caractéristiques de l'état du procédé. L'utilisation de moyens informatiques permet maintenant d'envisager l'usage de nouveaux outils plus performants, permettant une aide à la décision efficace. Notre approche consiste à utiliser des techniques de type système expert, solution véritablement intéressante pour concevoir un système intelligent de pilotage. La difficulté réside alors dans la définition et l'élaboration d'un modèle cohérent de l'unité de fabrication, respectant son organisation et sa structure interne. Le modèle peut être obtenu à partir d'une analyse qualitative qui permet de prendre en compte les différentes fonctionnalités et connexions entre chaque sous-système élémentaire.

Le choix du modèle a été guidé par l'objectif de la supervision qui est de maintenir une production régulière et constante de l'unité de production, à travers la commande de ses machines. Nous nous sommes ainsi intéressés à la conduite effective des machines en gérant

correctement et intelligemment leurs différents modes de marche. L'intérêt de la gestion des modes de marches d'un système de production est de prendre en compte toutes les contraintes existantes apparaissant lors de l'exploitation d'un système de fabrication. Ceci permet d'assouplir l'usage des automatismes et de minimiser les temps transitoires entre des changements de modes de marche. La difficulté d'exploitation des systèmes automatisés de production résulte des possibilités de plus en plus riches d'évolution, indépendante ou dépendante, de leurs différentes entités : il s'agit de tenir compte tout à la fois des capacités des machines, des ressources humaines et du produit, ceci dans le but final de mieux produire. Ces composantes ont des interactions induisant un comportement du système global, très complexe à appréhender, que ce soit en régime permanent ou en régime transitoire. De plus, le caractère "temps réel" de la production impose d'intégrer rapidement des anomalies de comportement, qui sortent du cadre de fonctionnement initialement prévu. Le système de supervision se doit alors de non seulement détecter les aléas de comportement mais aussi d'exploiter ces informations en vue de prises de décision correctes.

Dans ce cadre, nous avons proposé une méthode d'exploitation en nous intéressant plus particulièrement aux cycles de fonctionnement des machines. De ce fait, nous n'avons pas tenu compte du point de vue "produit", qui est utile, par ailleurs, pour concevoir une unité de production.

Le formalisme que nous serons amenés à utiliser, est basé sur des concepts relevant des tables de contraintes, des graphes d'état et des représentations arborescentes. En effet, plusieurs modèles de description sont nécessaires, chacun d'eux prenant en compte des points de vue complémentaires.

La modélisation permet d'obtenir, en fin d'analyse, une description structuro-fonctionnelle du système, assurant ainsi une décomposition systémique qui caractérise les systèmes de production. Par une démarche ascendante de regroupements, nous construisons une décomposition structurelle et fonctionnelle de l'unité de production, jusqu'à recouvrir le procédé par une seule machine virtuelle. Cette approche permet de dégager les différents niveaux de commande. Cette décomposition est ensuite exploitée pour évaluer l'état global du système. Il dépend de la combinaison de l'état de chacune des machines, dont les comportements non autonomes, sont liés par des contraintes de fonctionnement. Par une démarche ascendante, nous évaluons les états des regroupements de différents niveaux jusqu'à l'obtention de l'état du système complet.

Afin d'exploiter et de valider ces résultats, nous avons conçu, à titre d'illustration, un système de pilotage complet, incluant sa partie commande et son niveau hiérarchique.

L'élaboration de la partie commande, issue de résultats de travaux antérieurs du laboratoire, débute par l'obtention d'un modèle en RdPSAC (Réseaux de Petri Structurés Adaptatifs Colorés), traduit ensuite en langage Grafcet en vue de son exploitation.

La conception du niveau hiérarchique est quand à elle, plus prospective. C'est pourquoi nous nous sommes intéressés à son organisation générale, notamment en nous attachant à la spécifier et à la structurer. Son architecture est basée sur la coordination générale de l'ensemble des fonctions de pilotage par un gestionnaire. Celui-ci assure conjointement, la gestion de modes de marche des différentes machines de l'unité de production.

Le fonctionnement du gestionnaire repose sur la coopération de plusieurs systèmes d'inférence, l'un destiné à un raisonnement déductif et un autre destiné à un raisonnement planificateur, dont les bases de connaissance sont générées à partir du modèle structuro-fonctionnel obtenu précédemment.

Ce mémoire est composé de trois chapitres :

- le chapitre I présente le contexte industriel dans lequel nous travaillons en insistant particulièrement sur les différentes notions qui lui sont associées, telles que la productique et la flexibilité,

- le chapitre II définit la méthodologie conduisant à l'obtention d'un modèle complet du système de production à superviser. Nous introduisons les différents outils retenus lors de la phase de description : **les graphes d'états**, afin de modéliser le comportement dynamique des différentes entités du système, et **les arborescences**, afin de dégager une représentation structurée et hiérarchisée,

- le chapitre III décrit comment nous exploitons cette modélisation pour élaborer le gestionnaire de modes de marche. A travers un exemple d'implantation basé sur l'atelier flexible de l'EC Lille, nous indiquons de quelle façon, nous structurons la commande d'une unité de production et comment nous y intégrons le gestionnaire de modes de marche parmi l'ensemble des fonctions de pilotage. Nous présentons, pour finir, comment l'association d'un niveau hiérarchique et d'une partie commande autorise l'intégration des différents modes de marches et de leurs transitoires.

# CHAPITRE I

## Présentation générale

# INTRODUCTION

Les industries traditionnelles, jusqu'à maintenant, étaient figées au niveau de leur production. De nouvelles contraintes conjoncturelles sont apparues, redéfinissant complètement la conception et l'orientation des usines. De ce fait, le pilotage et la supervision des systèmes modernes de production manufacturière impliquent l'amélioration de multiples opérations, de la conception du système de fabrication, jusqu'à sa conduite effective. Ces opérations concernent aussi la surveillance et la maintenance...

L'organisation structuro-fonctionnelle de l'usine doit être repensée. Traditionnellement, chaque système est conçu comme une entité indépendante. Le concept CIM vise à faire disparaître ces méthodes, afin d'avoir une approche plus générale.

Concilier l'ensemble de ces tâches est un vaste problème et la solution possible est l'intégration par l'informatique de ces différentes fonctions. Cependant, cela nécessite de repenser la nature des apports de l'informatique.

L'informatique ne sera plus seulement vue comme un outil, mais aussi comme un moyen qu'il est nécessaire d'intégrer tout au long de la démarche de conception au même titre que, par exemple, le potentiel humain. L'organisation matérielle est inévitablement modifiée en profondeur. Par exemple, la structure du système de commande sera, par nature fortement répartie, en fonction de la dispersion physique des outils informatiques. Cela suppose l'introduction d'outils nouveaux tels que les réseaux, les bases de données réparties et les systèmes d'exploitation temps réel.

L'intérêt de ce premier chapitre est qu'il présente les systèmes de production, d'un point de vue CIM, dont nous en montrons l'importance et les implications économiques, fonctionnelles et méthodologiques. Nous voulons montrer les difficultés que soulèvent l'élaboration d'un système de production flexible, difficultés rencontrées d'ailleurs lors de la validation et l'implantation de nos travaux sur site. Nous précisons de plus certaines notions, utilisées tout au long de ce rapport.

Quatre parties sont proposées :

I La première partie caractérise les notions liées à la production automatisée et informatisée.

II La seconde partie décrit l'architecture hiérarchisée, matérielle et logicielle, d'un système de production moderne.

III La troisième partie présente les outils qui permettent cette automatisation poussée et leurs caractéristiques.

IV La quatrième partie montre la nécessité d'avoir une méthodologie globale de conception. Nous montrerons en particulier, les travaux développés au laboratoire d'Informatique Industrielle de Lille (LAIL) de l'EC Lille, qui vise à développer une méthodologie de Conception Assistée de Systèmes de Production Automatisés pour l'Industrie Manufacturière (CASPAIM).

# I - LA PRODUCTION

## I . 1 - Présentation

Actuellement, il est possible de caractériser les industries comme :

- **complexes** : cela dépend du type de la production, la taille de l'entreprise, sa structure et son organisation,
- **hétérogènes** : souvent, il y a plusieurs départements de fabrication, situés en des lieux éloignés,
- **spécialisées** : la spécialisation apparaît de plus en plus impérative,
- **hiérarchisées** : une structuration des fonctions de commande en plusieurs niveaux d'organisation,
- **flexibles et dynamiques** : il faut pouvoir réagir et s'adapter rapidement aux évolutions du marché et aux besoins des clients,
- **contraintes** : la production doit se soumettre à des impératifs politiques ou conjoncturels,
- **compétitives** : la notion de productivité est essentielle, en terme de profits et de rentabilité pour les entreprises.

### Les objectifs

En tenant compte de ces critères, un système de production moderne se doit donc de répondre aux impératifs suivants :

- possibilité d'appréhender facilement les variations de la production,
- maintenance d'un taux de charge et de productivité élevée, qui n'est possible que grâce à une automatisation poussée.

La flexibilité sous toutes ses formes est le critère essentiel d'adaptation qui modifie les mécanismes de décision permettant d'assurer des tâches très différentes. C'est la nature discontinue de la demande du marché qui a généré l'introduction de méthodes de fabrication flexible. Une approche flexible permet de rendre continu le flux de produits. L'automatisation permet de réduire la main-d'oeuvre tout en ayant une qualité plus régulière des produits. La densité des moyens et des enjeux est si importante qu'il faut nécessairement appréhender l'automatisation de manière radicalement différente. La conception de systèmes de production est devenue un problème majeur, car il faut pouvoir garantir un outil de production capable de répondre aux conditions initiales imposées, et dont la durée d'existence et de validation sera

au moins assez longue pour pouvoir amortir les investissements. Il faut de plus pouvoir intégrer d'autres éléments au fur et à mesure de l'évolution des besoins.

Auparavant, l'utilisation et la mise en oeuvre d'un système de commande s'appliquaient beaucoup en terme de régime permanent qu'il s'agisse des systèmes continus, ou éventuellement des systèmes à événements discrets. L'apparition de l'ordinateur a rendu possible l'automatisation des processus discrets en régime dynamique et son introduction s'avère désormais indispensable.

La production intégrée ne peut alors être introduite simplement en restructurant et en adaptant l'organisation physique du processus de production (nécessité d'outils d'ingénierie [MOR 90]).

Des notions nouvelles sont de ce fait apparues telles que la productique, la flexibilité et le temps réel.

### I . 2 - La Productique

Ce nouveau domaine d'intérêt est né des besoins exprimés par les industriels. En effet, l'automatisation nécessite de nouveaux outils, de nouvelles méthodologies, de nouvelles techniques qui mettent en jeu l'informatique et l'ordinateur. Le concept de Productique provient de l'amélioration que procure l'informatique aux techniques classiques (mécanique, électronique) dans le cadre de l'automatisation des activités des processus de fabrication (étude, développement, maintenance).

La productique est donc la combinaison de l'automatique et de l'informatique industrielle.

Une définition de la productique a été proposée [FRO 84] :

> "Ensemble des techniques et des moyens tendant à automatiser les activités de la production dans les phases de la vie d'un produit : définition, étude, fabrication, après-vente"

Elle fait bien ressortir le fait que tous les moyens sont intégrés pour augmenter la productivité. L'informatique peut être considérée comme un outil privilégié pour ces objectifs. Elle assistera et remplacera progressivement l'automatique, qui était à l'origine, plutôt destinée à la commande et au suivi de processus continus. La production discrète dans le cadre des industries manufacturières implique de nouvelles techniques adaptées à la spécificité des problèmes abordés.

Au départ, la productique a classiquement pour buts [BEN 86] :

- de diminuer les délais de fabrication et d'attentes,
- de diminuer les stocks et les en_cours,
- de diminuer les coûts,
- d'améliorer la qualité.

L'optimisation des objectifs précédemment cités reste liée à l'aide que peut apporter l'informatique à l'utilisateur et au gestionnaire. Des techniques d'aide et d'assistance par ordinateur en sont issues : XAO, CAO, DAO, FAO...

Cependant un problème d'intégration réside dans l'utilisation complète de ces logiciels. D'une façon générale, il est nécessaire d'avoir une stratégie volontaire d'intégration, en particulier lors de la conception du système.

### I . 3 - La flexibilité

Qu'est ce que la flexibilité ?

Souvent, jusqu'à présent il était possible de trouver des machines dédiées à la fabrication d'un produit. En cas de changement de production, ces machines spécifiques étaient abandonnées car impossibles à reconfigurer et à adapter. Ceci entraînaient d'énormes pertes du fait des mises à jour matériel et système. Les industries ont voulu se montrer beaucoup plus réactives face aux aléas de la demande. En effet, l'exigence des consommateurs a entraîné une dynamique maximale dans la diversification des produits. Cela peut être par exemple :

- un changement de couleurs,
- des options très diverses.

Les petites séries se sont multipliées et il est devenu obligatoire, pour réduire des coûts d'amortissement, de réutiliser et d'optimiser le fonctionnement de ces machines de façon à réagir le plus rapidement aux variations de la demande. Sans remettre en cause le process, les machines sont devenues technologiquement plus performantes pour être capables d'une prise de décision, certes limitée mais qui offre des possibilités intéressantes.

La flexibilité du système global repose sur la flexibilité de ses sous-systèmes physiques et informationnels. Nous pouvons distinguer plusieurs types de flexibilité. En effet, différents points de vue sont proposés [GER 83] :

- **flexibilité de conception :** il s'agit de la possibilité d'adapter le process en fonction de caractéristiques particulières qui peuvent être introduites dans le même produit de base,
- **flexibilité de mélange :** il s'agit de la possibilité de produire simultanément sur une même machine, des produits de même nature mais de type différent,
- **flexibilité du procédé** : cela concerne le fait que l'ensemble du process peut être simplifié et complexifié par retrait et ajout d'opérations productives à des coûts faibles,
- **flexibilité de volume** : qui renvoie à la possibilité du process de faire face à des variations de la demande correspondant à des fluctuations importantes sur les quantités à produire,
- **flexibilité de routage** : flexibilité dans le choix du routage des pièces par multiplications de chemins possibles dans le trajet d'une pièce et d'un produit lors de sa fabrication,
- **flexibilité des opérateurs** : il s'agit d'offrir à un manipulateur la possibilité d'effectuer une séquence d'opérations différentes, simplement par un changement d'outils ou de logiciels de commande,
- **flexibilité dans le séquencement des tâches** : le séquencement des tâches n'est pas figé et peut être modifié afin de prendre en compte d'éventuels dysfonctionnements.

**De nouvelles architectures**

De nouvelles architectures [BON 85] sont issues de ces approches et démarches flexibles. Il est aujourd'hui permis de concevoir des unités de production automatisées et flexibles, telles que la cellule ou l'atelier flexible. L'architecture doit fournir des possibilités d'effectuer des opérations parallèles et hiérarchisées. Les mécanismes de décision doivent assurer plusieurs tâches différentes telles que la coordination des processus de fabrication, l'exécution des tâches ou l'organisation des programmes d'usinage.

Par exemple, **une CFA** (cellule flexible d'assemblage) [BOI 88] réalise potentiellement, non pas une seule tâche mais un ensemble de tâches. Elle doit être flexible et autonome. Elle est généralement destinée à un seul type de fabrication. Un atelier, regroupement de cellules, doit pouvoir usiner un nombre important de pièces de types différents. Le routage des pièces doit être flexible et l'engagement des machines maximal, par la minimisation des temps d'attente et de reconfiguration. Le système flexible idéal doit pouvoir traiter plusieurs types de pièces en utilisant simultanément plusieurs postes avec un lancement indifférent des pièces. Son autonomie est importante car l'homme est absent le plus souvent de ces structures.

## I . 4 - Le temps réel

### Introduction

La haute technicité des applications est souvent associée au concept de temps réel, qui est obligatoire pour la commande de processus. Les applications "temps réel" sont de plus en plus nombreuses, dues au développement des secteurs aéronautiques ou manufacturiers. Le plus souvent, la notion de temps réel n'est qu'une composante particulière de ces applications.

Il est apparu progressivement que le temps réel était mal abordé, résolu souvent par des techniques intuitives. Des normes et des protocoles tendent à apparaître dans le monde industriel, qui homogénéisera les démarches de développement.

Une définition [CNR 88] du temps réel a pu être proposée :

> "C'est une application mettant en oeuvre un système informatique dont le fonctionnement est assujetti à l'évolution dynamique de l'état d'un environnement (en l'occurrence le procédé dans le cas du manufacturier) qui lui est connecté et dont il doit contrôler le comportement."

C'est donc l'asynchronisme de l'évolution de l'environnement par rapport au contrôle automatisé qui est la source des difficultés du temps réel.

### I . 4 . 1 - Caractéristiques

Les applications dédiées à ces systèmes industriels conservent globalement les mêmes caractères que les autres applications. Cependant, elles différent sur certains points fondamentaux :

#### I . 4 . 1 . 1 - Contrainte de temps

En général, ce sont des systèmes destinés à une commande en ligne ou en temps réel qui interagisssent essentiellement avec l'extérieur. Ils réagissent à des occurrences extérieures, imprévisibles tout en étant soumis à des contraintes de temps imposées par le procédé. L'originalité réside dans le fait que les signaux sont émis de manière totalement asynchrone et que le système de commande doit être capable d'y répondre dans un temps assez bref et imposé.

Le temps de réaction est fonction des dynamiques internes du procédé (cas de process continu) ou des cadences de production (processus discrets).

Le temps peut être perçu de deux manières :

- temps logique : le procédé n'est observé qu'à des temps bien définis, par exemple, après un cycle d'évaluation,
- temps physique : le temps est géré de manière continue, par l'intermédiaire d'horloges physiques.

Si la réponse n'est pas assez rapide, l'information peut être perdue ou le système peut évoluer dans un état critique (cas du contrôle de trafic aérien).

#### I . 4 . 1 . 2 - Contrainte de parallélisme

Les applications temps réel sont caractérisées par la simultanéité importante du séquencement des actions ou des détections. Le procédé émet des signaux de manière parallèle, auquels le système de commande doit réagir quasiment instantanément. Il est donc naturel de construire un système d'exploitation capable de gérer le parallélisme des processus en tenant compte de la possible simultanéité des signaux. Le plus souvent, la gestion des tâches est réalisée de manière transparente par le système d'exploitation, chaque tâche ayant un niveau de priorité dépendant de leurs contraintes de temps. La garantie d'une réponse effective et dans un délai bref n'est que rarement assurée et reste difficile à valider.

#### I . 4 . 1 . 3 - Contrainte sur les entrées/sorties

Le système de pilotage a un couplage plus étroit avec l'extérieur et fait intervenir un grand nombre d'E/S pour la conduite effective. Ceci nécessite non seulement un dispositif capable de gérer ces E/S mais de les appréhender en temps réel et avec rigueur, d'où l'utilisation de capteurs précis et fiables.

#### I . 4 . 1 . 4 - Contrainte sur l'environnement

L'environnement industriel peut être agressif. Il nécessite, de ce fait, du matériel apte à assurer un fonctionnement normal quelques soient les conditions, d'où l'utilisation de matériels spécialisés fiables (automates, machines à CN).

### I . 4 . 1 . 5 - Contrainte de fiabilité

La génération d'un ordre erroné ne peut être effectuée. En effet, les conséquences peuvent être dramatiques surtout s'il n'est pas permis de revenir en arrière.

Ces différentes notions aboutissent à une nouvelle approche de l'environnement orientée vers le concept général : **CIM** (Computer Integrated Manufacturing).

## I . 5 - Le contexte CIM

Le CIM vise à intégrer, le plus possible, les différentes fonctions de l'usine par l'intermédiaire de l'informatique. Le but final et théorique d'une méthodologie de conception orientée CIM est l'automatisation complète de l'usine.

Définir un contexte CIM est complexe, car vers où faut-il orienter l'intégration de l'informatique ? en effet, de multiples domaines sont concernés : choix méthodologiques, gestion des ressources humaines et matérielles, gestion de l'information. Une tâche essentielle relevant de l'intégration concerne la gestion optimale des flux d'information dans l'entreprise et l'usine [VER 91]. Une autre tâche est la coopération de systèmes et de matériels informatiques, plus ou moins homogènes. Une autre est le respect des contraintes inhérentes aux notions de flexibilité de la production.

Les enjeux sont importants : amélioration de la productivité avec une diminution du personnel. Cependant, le problème d'une méthodologie orientée CIM est qu'elle ne peut pas être systématique, automatisable et directe car dépendante de paramètres spécifiques à l'usine tels que par exemple :

- le type et la taille de l'entreprise,
- la capacité d'investissements,
- la technologie déjà existante,
- le type de la production,
- la qualification du personnel.

Un contexte CIM pose deux problèmes essentiels de communication :

- interconnecter les machines hétérogènes,
- interconnecter les applications, pour l'échange de données : (EDI : champ de données informatisées).

Le contexte CIM ne sera possible que s'il existe une normalisation, une standardisation concernant tous les domaines impliqués dans le système de production, en particulier les bases de données, les protocoles de communication, les méthodologies de conception.

### I . 5 . 1 - Pourquoi le besoin d'intégration ?

Une vision systémique permet de considérer un système de production comme un ensemble de sous-systèmes indépendants. La connexion de ces systèmes s'avère nécessaire lorsque apparaît le besoin de les faire coopérer, de manière optimale et globale. Néanmoins, étant rarement compatibles, se pose alors le problème de leur branchement sur un support électrique (point de vue physique) ainsi que la pertinence des données à transférer (point de vue applicatif). C'est pourquoi la nécessité d'intégration de ces facteurs se fait ressentir, intégration qui doit débuter dès la démarche de conception.

L'intégration nécessite la prise en compte de tous les systèmes d'information de l'usine :

- contrôle de l'entreprise,
- conception des produits,
- gestion de production,
- surveillance et contrôle qualité,
- stockage et communication.

Un système performant doit rendre accessible toute donnée nécessaire à une prise de décision, et une décision doit être prise au niveau approprié. Sans cela, il ne sera pas possible de parler d'intégration.

Pour résoudre ce problème d'intégration, nous nous intéresserons plus particulièrement aux potentialités qu'offre l'outil moderne qu'est l'informatique. L'ordinateur apparait comme le partenaire et le support idéal de l'intégration.

Lors de la conférence de L'Institut of Mechanical Engineers sur les "systèmes CAO/FAO efficaces" à Cambridge en 1983 une liste a été dressée des 8 principes que requiert une planification d'un système CIM. Nous en citons les principaux :

- toutes les données techniques ou commerciales ne sont introduites dans le système qu'une seule fois au moment où elles sont créées,
- toute modification est intégrée au système le plus tôt possible pour en informer toutes les personnes concernées,
- tous les services ont accès aux informations nécessaires à leurs tâches en temps réel,

- les séquences informatiques de mise à jour sont exécutées automatiquement suivant des règles pré_déterminées.

L'intégration de l'informatique est un processus très lent : citons quelques chiffres publiés dans la revue Robotique Informatique Industrielle de Janvier/Février 91 : 51 % des établissements industriels possèdent au moins un ordinateur, 42 % sont équipées en GPAO et 20 % sont équipées en CAO/DAO.

D'une façon générale, les PME sont beaucoup moins équipées que les grandes sociétés. En effet, la spécificité des entreprises et de leurs tâches rendent difficiles leur automatisation. De plus, les développements matériels ne sont pas assez performants pour remplacer idéalement l'homme. Par ailleurs, l'intégration de l'informatique fait partie d'une démarche générale qui engendre une restructuration complète, difficile à envisager au départ. Une autre raison est le coût important des investissements en automatismes, très difficiles à rentabiliser (70 % des industriels en sont convaincus [BEN 85] ).

### I . 5 . 2 - Que vise l'automatisation poussée ?

Les objectifs de l'automatisation sont multiples :

- elle cherche d'une manière générale, à accroître le potentiel technique du matériel avec par exemple :
  * l'accroissement de la capacité de programmation et de commande des machines,
  * le renforcement de leur flexibilité : capacité à réaliser des tâches différentes,
  * introduction de "l'intelligence" dans les machines pour les aider à traiter des situations imprévisibles.

L'avantage des commandes par ordinateur est que les fonctions de gestion sont fixées par programme et sont en général faciles à modifier.

- cependant, il faut essayer, parallèlement, de diminuer les coûts d'utilisation. En effet, les industriels n'adopteront cette démarche d'intégration qu'à la condition qu'elle permette de réduire les coûts de production, en augmentant la productivité et la qualité. L'assistance par ordinateur peut contribuer à ce mouvement dans l'optique :

  * d'une génération d'une commande plus rationnelle et par l'optimisation de l'allocation des ressources. Il en résulte immédiatement une diminution des

temps d'attente après opérations, bénéfice lié à l'allocation réalisée en temps masqué,

* d'une amélioration de la qualité par une automatisation des tâches manuelles, d'où une réduction du personnel,
* d'une meilleure gestion des stocks avec la diminution des stocks supplémentaires et inutiles.

Les applications de l'informatique sont multiples dans le cadre de la gestion, la conception, la fabrication, la commande ou la communication avec l'introduction du concept XAO.

### I.5.3 - Le degré d'automatisation

L'industriel peut chercher à estimer le degré d'automatisation de son usine mais il est très difficile d'évaluer l'automatisation d'un système de production car les notions qui lui sont liées restent assez subjectives. Une définition a été proposée [SIM 72] :

> "le degré d'automatisation d'un système de production est le quotient du nombre partiel des opérations faites dans le système, selon un programme fixé en déroulement ininterrompu et de la quantité totale de toutes les opérations exécutées dans le programme."

Cette définition demeure applicable dans le cas d'un processus robotisé mais ne l'est pas dans l'évaluation globale d'un système. La difficulté d'évaluation réside dans la prise en compte des opérateurs humains dans le déroulement des opérations.

### I.5.4 - Raison d'une méthodologie orientée CIM

Il ne sert à rien d'automatiser une des fonctions de l'entreprise sans chercher à tout automatiser. En effet, il est possible de mettre en place un système de CAO qui permettrait de générer par exemple un nouveau type de pièce, mais tout son potentiel ne serait pas pleinement utilisé dans le cas où le système de production ne serait pas assez flexible et adaptable pour pouvoir intégrer rapidement cette pièce dans la fabrication.

La production en contexte CIM, nécessite lors de la conception de l'usine, une approche différente car il ne faut pas considérer l'ordinateur comme un simple outil mais comme un moyen d'intégrer les différentes fonctionnalités de l'usine.

Ces fonctionnalités vont de la gestion des informations au pilotage effectif de l'usine.

### I . 5 . 5 - Impact de l'automatisation

Les perspectives et les enjeux de la productique sont nombreux. Les conséquences industrielles sont les améliorations significatives des technologies mises en oeuvre. Les machines disposent aujourd'hui de contrôleurs numériques leurs permettant de changer de programme de fabrication instantanément, certains disposant d'une puissance de calculs largement équivalente à celles de nombreux calculateurs.

Les conséquences sociales sont significatives. Les conditions de travail sont nettement améliorées par le fait que les tâches pénibles et répétitives sont facilement prises en charge par les robots. Les tâches affectées aux hommes sont plus enrichissantes et nécessitent un meilleur niveau de qualifications d'où la nette valorisation de l'opérateur humain.

### I . 5 . 6 - Exemple d'objectif, JIT

Un des buts d'une orientation CIM est le "JUST IN TIME" : il faut pouvoir produire et livrer quelque soient les conditions (en particulier les risques de pannes, le dépassement du temps imparti à des opérations). Le principe est de pouvoir éliminer les temps d'attentes et les interruptions afin que le cheminement du flux de production soit continu. Une étude [ING 88] a montré que les entreprises perdent environ 20% de leurs coûts de fabrication dans des tâches non productives (stockage et transport).

## I - 6 - Exemple d'usine flexible, l'usine BULL de Villeneuve d'Ascq

L'usine Bull a été construite en 1986 à partir de l'idée de concevoir un système de production flexible, plus exactement un système d'assemblage de matériel informatique, capable de concurrencer le marché Asiatique dans ce domaine.

Pour répondre à ces impératifs, une méthodologie de type CIM a été respectée.

L'originalité de cette usine réside dans le fait que les produits à assembler subissent très peu d'opérations manuelles, mises à part deux étapes très difficiles à automatiser. Ces étapes sont le dépotage avec le remplissage des bacs destinés au stockage et l'assemblage effectif des pièces.

Le reste des manipulations est automatisé.

**Description du flux des matières** [VEK 89]

Il est intéressant de regarder la disposition matérielle à travers le flux de matières : on peut voir que le cheminement est cohérent, avec la minimisation des trajets entre chaque emplacement.

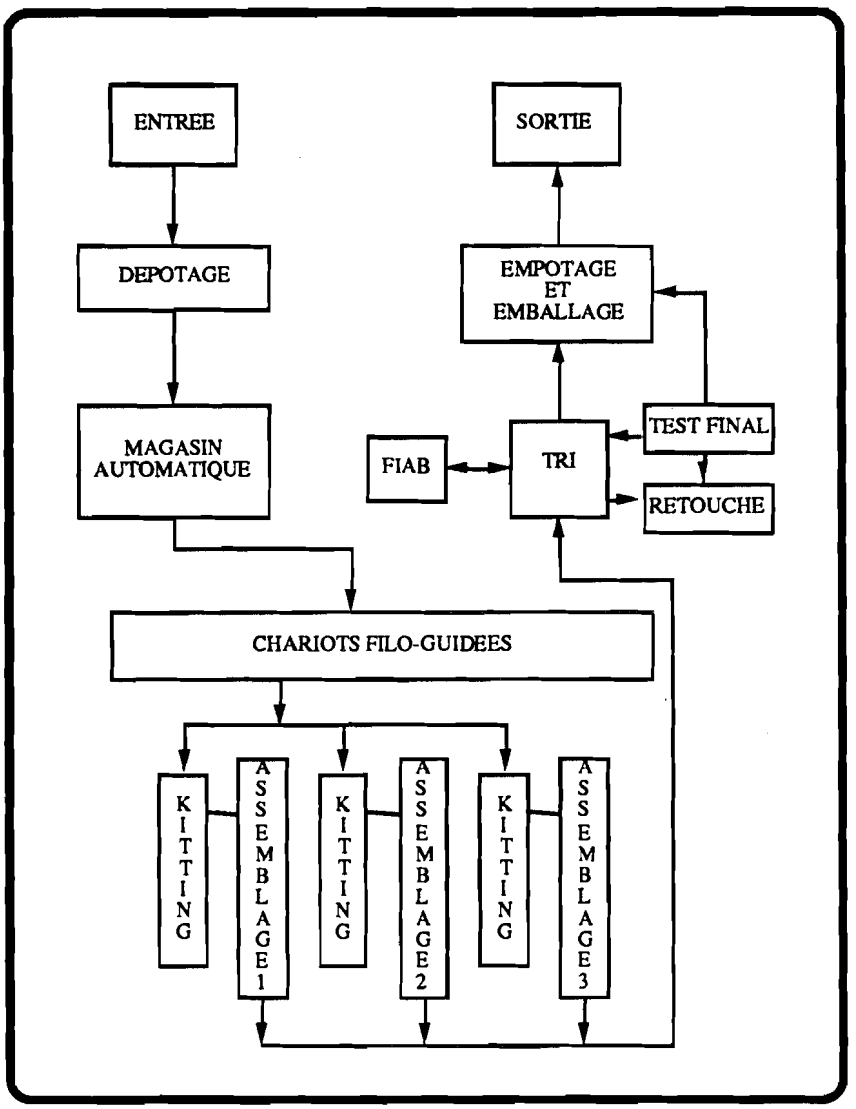



fig 1 : usine BULL, le flux de matières

Description des principales étapes :

Les étapes de kitting et de tests sont automatiques.
**Le kitting** est une étape de préparation des kits destinés à l'assemblage.
**Les tests** sont effectués en trois temps :

- un premier test, dit "**d'intégration**", après le montage, pour vérifier la validité du montage;
- un deuxième test, dit "**de fiabilisation**" pour vérifier la bonne tenue des composants,
- un test final pour vérifier les parties vidéo.

En cas de détection d'un défaut, l'élément est dirigé vers un atelier de retouches, sinon vers la sortie.

L'ensemble des transmissions est supporté par un réseau LAC qui relie les automates et les ordinateurs.

Le CIM fait référence à une approche systématique dans la conception de l'usine : les concepts tels que ordonnancement [HAM 91], planification, pilotage temps réel sont parfaitement intégrés dans ce cadre (GPAO, GMAO...).

D'autre part, le CIM prévoit l'échange des données entre les différents niveaux de temps réel, afin que toute décision puisse être prise en toute connaissance de cause.

L'automatisation n'est qu'une conséquence du CIM, mais ce n'est pas le CIM. Cela fait longtemps que les processus de fabrication sont automatisés sans pour autant donner naissance à une usine intégrée.

## II - ARCHITECTURE DES SYSTEMES DE PRODUCTION

### Introduction

Un contrôle global automatisé de la production est un objectif ambitieux, étant donné la complexité du système abordé. La complexité de l'usine nécessite de la structurer et de la partager en sous-systèmes plus facilement supervisables. Il est possible de séparer les activités et les tâches concernant la fabrication du produit, des opérations de commande et de pilotage des opérations.

Chaque type d'activités peut être structuré en distinguant des sous-systèmes élémentaires :

- activités concernant la fabrication en sous-processus de fabrication (production ou transport),
- activités de contrôle en programmes d'automates ou d'ordinateurs.

Le contrôle général englobe des tâches telles que la planification de la production, le suivi de la production et le pilotage effectif. Elles imposent des conditions très différentes tant au niveau des informations qu'au niveau des délais. Une structuration hiérarchisée de l'architecture du système global s'impose. L'architecture signifie le découpage et les liens entre les divers paramètres existants (données, fonctions, structures).

Plusieurs critères sont pris en compte : nous distinguerons donc dans un premier temps les caractères de la structure physique de ceux concernant la commande. En effet, la structuration de la commande peut, ne pas correspondre à la structuration fonctionnelle du système, certaines fonctions de différents niveaux pouvant être assumées par une même machine.

## II . 1 - Architecture matérielle d'un système

Nous pouvons distinguer trois architectures de système : centralisée, décentralisée et répartie.

Ces architectures sont en fait étroitement liées à l'évolution des outils informatiques. Traditionnellement, l'architecture d'une usine était centralisée. L'utilisation d'une architecture centralisée repose sur l'utilisation d'un gros ordinateur dans l'usine. Il est chargé de toutes les fonctions informatiques de l'usine. La multiplication d'ordinateurs plus petits, plus performants et moins chers permet de décentraliser les différentes tâches qui sont affectées à différentes machines en différents lieux.

L'apparition des réseaux informatiques permet de faire communiquer toutes ces machines, pour passer à une structure dite répartie.

### II . 1 . 1 - Structure répartie

Un système de pilotage est souvent constitué de plusieurs équipements informatiques pour diverses raisons :

- la nature même du process impose la coopération de systèmes de commandes destinés à des unités individuelles. Il s'agit alors de coordonner ces différents systèmes,
- la volonté d'avoir un process sans défaut, amène à une redondance du contrôle ; le système de pilotage est alors réparti sur plusieurs machines dans le but d'appréhender une panne d'un des systèmes de commande,
- des contraintes de temps imposent des équipements de pilotage individualisés, de façon à mieux satisfaire les temps de reprises. Les actions sont parallélisées, d'où l'utilisation de plusieurs unités de traitement.

Par opposition à un système décentralisé, ces équipements sont tous capables de communiquer entre eux.

### II . 1 . 2 - Type d'architectures

Les différents types d'architecture matérielle répartie fondamentaux sont [BEN 86] :

- la configuration répartie sur deux ou plusieurs ordinateurs. La répartition des tâches se fait par incompatibilité. Une communication synchrone permet le basculement des tâches d'un ordinateur vers l'autre en cas de panne (ordinateur fractal/arrière),
- la configuration répartie avec une connexion asynchrone => basculement difficile.
- la configuration répartie avec une liaison synchrone autorisant un échange régulier de fichiers,
- la configuration répartie avec l'utilisation d'un réseau informatique qui permet le dialogue entre plusieurs ordinateurs,
- la connexion plus intime, avec connexion des bus de communication (adresse et données).

La conception de l'architecture découle naturellement lors de la phase de réalisation. En effet, normalement sont pris en compte des paramètres structurels de l'usine.

Un de ces paramètres est donc de gérer de manière optimale les sources d'approvisionnement. Les solutions sont par exemple, le regroupement des opérations successives ou l'adaptation des gammes opératoires.

## II . 2 - Lien avec le système d'informations

La multiplicité des informations est telle qu'il est impossible qu'elles aient la même importance. Par conséquent, la décision doit être hiérarchisée en niveaux, de manière unitaire à la structuration de l'architecture de l'usine. L'impact de la décision et de l'information sera dépendant de son niveau dans la hiérarchie.

## II . 3 - Niveaux de planification

Trois niveaux de planification de produit sont en général proposés : le long terme, le moyen terme et le court terme.

**Le long terme** : c'est le niveau supérieur de la hiérarchie. Ce sont des décisions prises par la direction, qui mettent en jeu la politique globale de l'entreprise. L'horizon se compte en années.

**Le moyen terme** : l'objectif est de construire un plan de fabrication avec la satisfaction des commandes en minimisant les coûts de production et en assurant un fonctionnement normal de la production. L'horizon peut s'évaluer en mois.

**Le court terme** : c'est l'organisation journalière du travail avec le choix d'une solution.

Evidement, chaque niveau se doit de respecter les orientations choisies aux niveaux supérieurs.

La nécessité de l'aide à la décision s'impose : si les objectifs ne sont pas atteints, pour une raison ou une autre, une correction des décisions initiales s'impose.

### II . 4 - Type de décomposition

En général, la décomposition du système de production aboutit à une structure hiérarchisée et répartie. De nouvelles structures fonctionnelles en découlent : l'atelier et la cellule flexible. Ainsi, une usine se décompose en ateliers, un atelier en cellules, une cellule en îlots de fabrication, un îlot en postes de travail, un poste étant un regroupement de machines.

Tout est informatisé et chaque niveau fonctionnel prend les décisions qui le concerne. Au sommet de la hiérarchie, il peut y avoir par exemple un ordonnancement à long terme. Le sommet de la hiérarchie ne coordonne certainement pas les ateliers et les cellules.

La cellule et l'atelier flexible répondent aux mêmes exigences de fabrication (possibilité de changements d'outils, transport automatique), cependant l'atelier conserve un niveau de décision supérieur et est capable d'optimiser les tâches en temps réel et est beaucoup plus autonome.

**La cellule** est un regroupement d'îlots. Elle permet de produire un élément complet au niveau de sa gamme d'usinage. Elle permet en général deux modes de fonctionnement : un mode autonome, géré de manière automatique et un mode dépendant, travaillant avec d'autres sous-systèmes équivalents, l'ensemble étant supervisé et coordonné localement par le niveau cellule.

**Le poste de travail** s'apparente à la commande locale d'un robot auquel il faut ajouter une machine d'usinage. Les outils, les programmes d'usinage sont disponibles. Le pilotage se fait par une commande numérique.

## II . 5 - Un système d'information

### II . 5 . 1 - Présentation

Les objectifs d'un système d'information sont la commande effective des machines, le contrôle, la coordination et la gestion de l'information.

Les systèmes d'information prennent également en charge les communications. Souvent les données sont redondantes et présentées sous différentes formes. L'idée de l'intégration est de générer un système commun de données qui doit permettre une information cohérente et générale dans l'usine.

La grande qualité des systèmes d'information est d'être disponible pour un nombre réduit d'opérateurs, les structures des postes de conduite intégrant plusieurs niveaux de redondance. Les différents niveaux de responsabilités des intervenants sur les postes opérateurs sont autant d'éléments favorables à une surcharge d'informations rendant critique la conduite du procédé.

Une tendance naturelle et actuelle oriente la structure d'un système d'information vers une hiérarchisation répartie des tâches. Chaque niveau ou tâche de cette hiérarchie gère et stocke des informations de nature différente, les traite pour les communiquer aux couches adjacentes.

Cette structure est liée par ailleurs, à la décomposition physique et fonctionnelle du système à commander, ceci pour intégrer de manière cohérente les informations concernant le système.

**Définitions de critères imposées à une architecture de système :**

L'architecture du système doit être [BEN 86]:

- extensible, pour pouvoir appréhender une augmentation des données,
- versatile, capable de s'adapter à des changements,
- disponible, capable d'absorber la charge présente de flux d'informations,
- fiable, capable de supporter les dysfonctionnements du système,
- efficace, capable de réponse effective aux demandes.

Les bases de données permettent le stockage rationnel de toutes les informations relatives au fonctionnement de l'usine.

Les différentes tâches existantes nécessitent un large éventail de matériels et de logiciels, spécialisés dans leurs fonctions de gestion, de contrôle et de commande.

La répartition se fait par centres d'intérêt ou par tâches.

### II . 5 . 2 - Schéma structurel

Le fait de disposer d'un système d'information performant repose sur sa conception dont la méthodologie doit être sûre et fiable. Les systèmes de production dans leurs grandes majorités respectent le même schéma structurel au niveau de leur système de pilotage. Une théorie de J.L Lemoigne définit l'entreprise comme une entité de nature technique, sociale et économique avec des moyens spécifiques et adaptés à l'environnement. A partir de cela, l'architecture du système est réalisée.

Les systèmes d'information se décomposent, généralement, en plusieurs niveaux hiérarchisés. Nous pouvons distinguer quatre niveaux pour un système d'information et de commande.

#### II . 5 . 2 . 1 - Modèle de l'usine intégrée

L'approche générale est de distinguer 4 niveaux, chaque niveau étant identifié à des fonctions et à des données bien spécifiques :

- niveau 0 : niveau machine,
- niveau 1 : niveau cellule,
- niveau 2 : niveau atelier,
- niveau 3 : niveau gestion de production.

Ces niveaux font intervenir des informations bien spécifiques et donc globalement très hétérogènes .

Nous décrivons les tâches affectées aux différents niveaux de commandes :

#### II . 5 . 2 . 2 - le niveau 0 : process, capteurs et actionneurs

Il est appelé plus précisément le procédé ou partie opérative. Il est constitué de la partie opérative (actionneurs et capteurs) et des équipements (machines, pièces, logiciels).

**Les composants de ce niveau sont :**

- **les capteurs** : ils sont l'interface de mesure entre le procédé et la partie commande. Leurs types sont très différents car ils mesurent des entités très

variées : capteurs de proximité, capteurs analogiques de mesure de courant électrique, capteurs de forces... Ils représentent le seul moyen de perception du système de commande,

- **les actionneurs** : ils convertissent un signal de commande en une action effective au niveau du procédé.

Leurs précisions et leurs fiabilités sont des conditions primordiales pour un suivi correct de la production.

### II . 5 . 2 . 3 - Le niveau 1 : commande

Ce niveau est appelé plus précisément la partie commande. Elle assure la coordination et la synchronisation des commandes envoyées vers le procédé. Elle est généralement constituée d'automates de haut niveau. Elle doit permettre par exemple, de choisir les programmes d'usinage par émissions de requêtes vers un calculateur plus spécialisé dans cette tâche.

### II . 5 . 2 . 4 - le niveau 2 : supervision et surveillance

Il constitue un premier niveau réel de prise de décision intelligente. Il est généralement constitué de stations de travail ou de micro_ordinateurs. Il doit permettre la surveillance de machines moins puissantes. Son rôle est de résoudre les indéterminismes de la partie commande au travers de prises de décisions qui peuvent être localisées. Il gère par exemple les transitoires entre les changements de modes de marche.

Sa capacité de calcul doit lui permettre la compréhension de différentes tâches : contrôle d'exécution, surveillance. De plus, on peut considérer que c'est la mémoire et le cerveau de la cellule. Il mémorise l'ensemble des fichiers de commande.

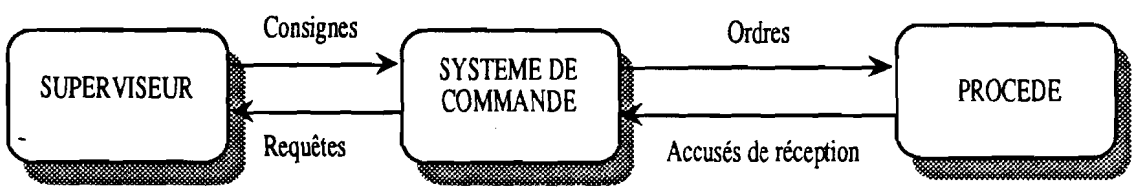



fig 2 : schéma de contrôle

### II . 5 . 2 . 5 - Le niveau 3 : gestion

Il s'apparente à la gestion de production. La portée de ses décisions décide de la stratégie à moyen et long terme de la production. Le niveau 3 en général, est constitué d'un gros ordinateur.

En tenant compte de ces caractéristiques, le modèle habituel de l'usine intégrée est présenté ci-dessous :

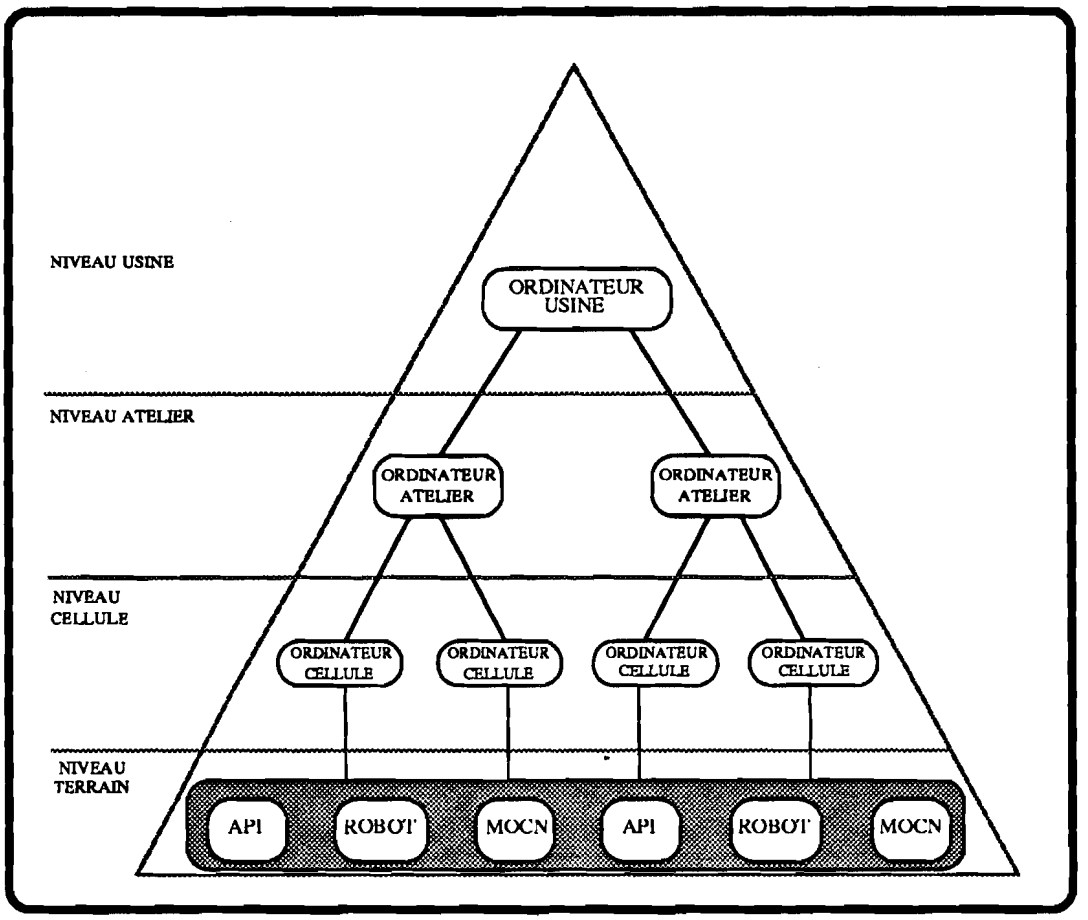



fig 3 : architecture fonctionnelle hiérarchisée d'une usine [LEP 89]

### II . 5 . 2 . 6 - Avantage de la structure hiérarchisée

Les avantages de la structure hiérarchisée sont les suivants :

- le caractère modulaire de l'usine permet une installation et un suivi module par module,
- contrôle redondant : il est toujours possible de maintenir un contrôle de production, lors de panne d'une machine du système de commande,

- à chaque tâche de pilotage est affectée une machine spécialisée adaptée à son niveau de commande et aux logiciels nécessaires,
- elle permet de délocaliser de l'intelligence avec l'avantage de travailler en temps réel. La machine se trouve beaucoup plus proche de l'information et les temps de réponse sont donc améliorés,
- la répartition du système de décision permet un fonctionnement autonome de chaque module.

## II . 6 - Caractéristiques des systèmes d'information industriels

La mise en oeuvre de systèmes temps réel reste difficile. La coopération de l'ensemble des outils utilisés lors d'un système temps réel est mal résolue. Il n'existe pas vraiment de méthodes de conception adaptées au temps réel et surtout à la haute technicité du matériel (environnement multi-processeurs). L'intégration de l'ensemble des contraintes telles que le temps, la gestion de données et la communication entre ressources dans un système réparti reste particulièrement pointue.

De plus, tout modèle capable de modéliser le parallélisme de tâches se ramène le plus souvent, lors de l'implantation effective du modèle, à un pseudo parallélisme (cas typique des langages synchrones qui construisent des automates dépourvus de tout parallélisme) à moins de se trouver dans un environnement multi-processeurs ou multi-machines.

### II . 6 . 1 - Exemple de contrôle

Un contrôle numérique direct (CND) ou contrôle informatique de la supervision (CNS) est très répandu dans les systèmes industriels, basés sur différents niveaux de décisions.

Le CND permet le contrôle d'une variable individuelle de sortie d'un processus. Ses capacités de calculs lui permettent de travailler en multi-tâches de façon à surveiller plusieurs variables de manière séparée.

Le CNS est à un niveau supérieur de contrôle. Le système de surveillance évalue les performances globales du processus pour déterminer quels sont les paramètres d'entrées du processus à modifier pour atteindre la consigne désirée.

### II . 6 . 2 - Problèmes de l'acquisition des données

Un des problèmes du temps réel est l'acquisition des données qui dégradent fortement le temps d'analyse. En effet, non seulement les données à collecter sont nombreuses mais en plus, circulent par des canaux de communication toujours plus lents que les possibilités de calculs des machines d'analyses.

Le processus impose des contraintes de temps de réponse et entre deux instants d'échantillonnage ou de lecture, le système ne doit pas évoluer de manière incontrôlée. Des constantes de temps sont impossibles à diminuer, dues par exemple au transfert de l'information dont le temps d'échantillonnage ou de lecture correspond en fait au temps de traitement du système de pilotage.

Une solution pour résoudre ce problème est de répartir le système de commande sur plusieurs machines qui traiteront les données séparément et parallèlement. Les supports de transmission sont adaptés aux données et permettent de paralléliser la transmission.

Une autre solution est la hiérarchisation des données : certaines variables sont plus importantes que d'autres. Elles doivent être prises tout d'abord en considération lors d'un fonctionnement normal. Mais par exemple, dans le cas d'un dysfonctionnement, le système a besoin de données plus précises et il va alors les chercher de façon à compléter sa base de données. Les grandeurs les plus notables sont celles qui prennent en considération le comportement du système (entrée, sortie). Peuvent être considérées comme auxiliaires les données concernant les flux d'informations sur les matériaux, les outils et les préparatifs.

Cette solution est intéressante dans la mesure où elle permet d'effectuer un pré-filtrage des données qui sont centralisées.

### II . 7 - Gestion des données

La constitution d'une base de données est nécessaire car elle permet de renseigner le gestionnaire sur l'état du système de fabrication, à des niveaux divers de précision. Le traitement de données sert à visualiser l'état du process, en particulier à renseigner sur les grandeurs essentielles. Une base de données doit être organisée autour d'une logique prédéfinie. Cela influe sur la manière dont vont être organisées physiquement les données et leurs vitesses d'accés.

L'avantage d'une base de données réside dans la référence commune que constitue la mémoire de l'entreprise, elle y conserve son savoir faire et son expérience, connaissances qui peuvent souvent être des heuristiques.

La gestion d'une base de connaissances est rendue complexe par l'hétérogénéité des données. Ces données sont de nature très diverse et en très grande quantité. Les différents types de données sont caractérisés d'après leurs niveaux d'apparition :

- **au niveau 0** :
  * les messages sont faibles,
  * le flux est périodique,

    * le temps réel est obligatoire,
- **au niveau 1** :
    * les données sont importantes et de type fichiers,
    * le temps réel est moins obligatoire,
    * les messages sont de type commande,
- **au niveau 2** :
    * le volume d'information est important,
    * le temps différé est possible,
    * les messages sont de type ordres de fabrications,
- **au niveau 3** :
    * il n'y a pas d'impératif de temps, au sens temps réel,
    * les volumes de message sont importants.

Il est impératif que cette base soit facilement accessible pour être exploitable. Un des problèmes pour le suivi est d'avoir des moyens de collectes fiables et rapides, cela nécessite souvent des investissements supplémentaires par l'introduction de réseaux informatiques.

Il est donc nécessaire d'utiliser des logiciels spécialisés dans les bases de données et de concevoir des moyens d'accès à ces informations. Cela permet une gestion dynamique et cohérente des données.

## II . 8 - LES XAO

De plus en plus, la gestion des données est assistée par ordinateur. Le traitement des informations est géré automatiquement par des systèmes informatiques.

Les avantages de l'aide apportée par l'informatique dans la réalisation automatique est la création d'une base de données éventuellement réutilisable pour faire évoluer le produit. De plus, plusieurs logiciels peuvent traiter tout au long du cycle de conception, de la pièce brute au produit final, un fichier correctement généré. L'idée est que tous les logiciels puissent s'interconnecter de façon à ne travailler que sur une base de données commune, avec minimisation du nombre d'opérations à réaliser dans le transfert de données. Les outils de type XAO sont liés à l'importance des fonctions de conception et de gestion. C'est pourquoi les industriels se sont attachés à promouvoir et à développer des logiciels de type DAO, CAO, FAO... Il est possible de complètement automatiser la chaîne de CAO qui générera, par exemple, directement à partir de la forme dessinée à l'écran, le programme de commande numérique destiné à l'usinage de l'objet.

### II . 8 . 1 - Problème des échanges de données

Les échanges se faisaient traditionnellement sous forme de papier avec une perte certaine de la qualité des informations et de temps du fait de l'incompatibilité des logiciels.

De plus en plus, les différents logiciels de l'usine sont amenés à dialoguer avec la CAO par l'intermédiaire d'une base de données commune. L'introduction des XAO doit mener vers une gestion "**zéro papier**". Lorsque tous les systèmes seront intégrés par la construction d'une chaîne logiciel, le bénéfice des XAO sera optimal dans l'élaboration d'un flux homogène de données.

### II . 8 . 2 - Logiciels existants

Actuellement, les produits références proposés [KOR 86] sont tellement performants qu'ils permettent un suivi en temps réel de la production. Les plus anciens progiciels fonctionnent tous en temps différé.

La gamme des produits est très vaste et il est possible de trouver une application pour n'importe quel type de machines.

Chaque logiciel est différent de par les fonctions qu'il utilise ou par le matériel nécessaire.

En général, d'après une étude faite par CAM-I ("Survey of Commercial Available Production Management Systems"), ces outils n'assurent que des tâches de gestion de stocks, contrôle de coûts, contrôles des achats et suivi de fabrication. Par contre, le contrôle de qualité, l'analyse de performances de travail, la gestion des outils et l'ordonnancement, tâches beaucoup plus complexes, sont moins souvent rencontrés.

### II . 8 . 3 - Flux des informations

Dans un système intégré, il est possible de consulter toutes les données de la production qui sont mises à jour en temps réel : état d'avancement des pièces, états des stocks, disponibilité des machines. IAO, CAO et FAO font intervenir les mêmes données. Un contexte CIM vise à rendre accessible toutes sortes de données de l'usine, par n'importe quel utilisateur, de manière automatique et transparente en tenant compte des conditions d'accessibilité. Cette transparence est assurée par l'interconnexion de réseaux et l'intégration de tous les systèmes informatiques de gestion de données.

**II . 8 . 4 -** Exemple de flux d'information dans l'usine

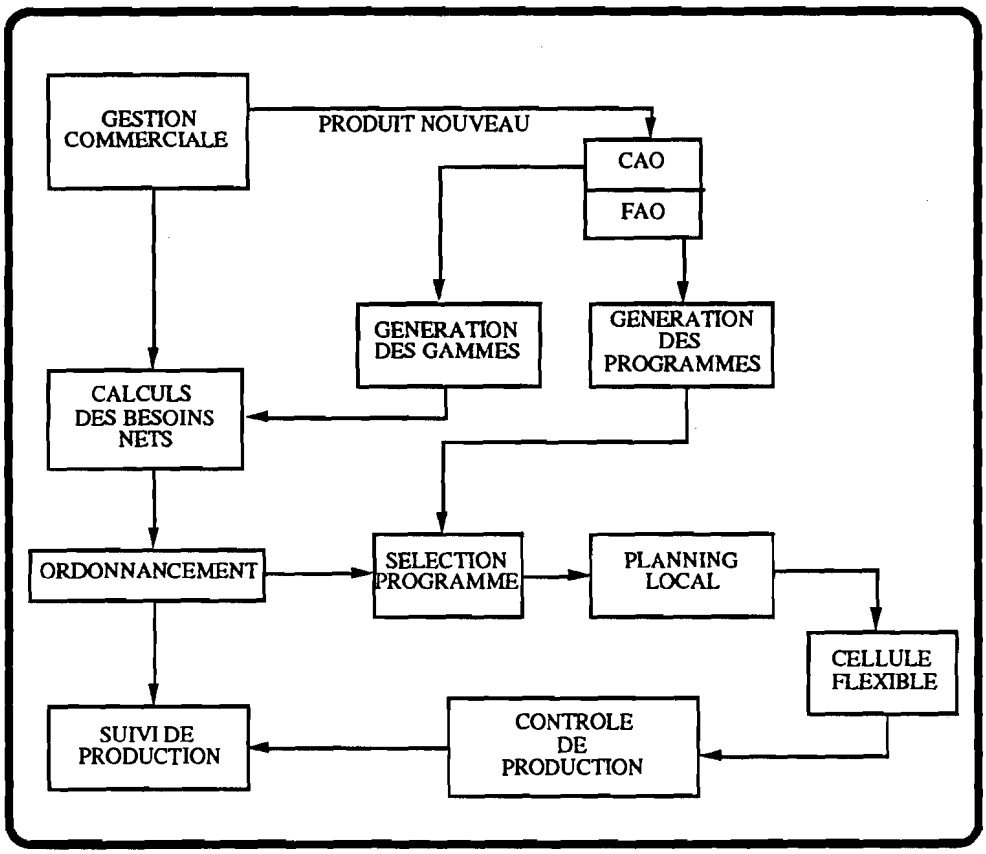



fig 4 : flux d'informations : CFAO/GPAO/ATELIERS [COL 91]

Ce schéma montre les différents et nombreux liens qui existent au sein d'une entreprise et l'importance des communications entre chaque sous-système. C'est pourquoi l'automatisation de ces communications est un facteur important et vital pour la réussite de l'entreprise.

L'analyse de la structure de l'entreprise peut se faire au travers de celle du flux d'information qui transite depuis les entrées jusqu'à leurs sorties (livraison) via leurs traitements (fabrication, production).

Nous présentons les différents logiciels d'assistance. Il faut cependant distinguer la GPAO des autres fonctions XAO. En effet, la GPAO traite les informations concernant le système de production, et permet l'élaboration d'une décision pour la conduite de l'usine. Les autres fonctions XAO, quand à elles, génèrent plutôt des données et des fichiers, et n'interviennent pas directement dans la prise de décision.

### II . 8 . 5 - GPAO (Gestion de Production Assistée par Ordinateur)

C'est une des fonctions importantes de l'usine. Elle relève à la fois des aspects techniques et sociaux. C'est le niveau intermédiaire entre l'informatique de gestion des fonctions administratives et commerciales de l'entreprise et l'informatique de pilotage du système de production.

L'aide de l'informatique à la gestion de production est indéniable par la possibilité qu'elle offre de prendre des décisions beaucoup plus rapidement et plus sérieusement avec la réduction des coûts de fabrication et une meilleure distribution du travail au niveau de l'usine.

#### II . 8 . 5 . 1 - Les fonctions de la GPAO

Une gestion de production satisfaisante, n'impose pas une décision élaborée hors de l'atelier. Elle se propose plutôt de fournir à l'atelier des éléments de décision avec la volonté de minimiser les stocks de fabrication et de respecter les délais de livraisons.

Les fonctions qui en découlent, sont multiples et nous allons décrire les plus importantes. Ce sont en l'occurrence :

- la gestion des informations
  * relatives au procédé (machines, outils),
  * relatives à la fabrication,
- le plan directeur de fabrication,
- le calcul des besoins,
- la gestion des stocks,
- l'ordonnancement,
- le suivi de fabrication.

#### II . 8 . 5 . 2 - Description de ces différentes fonctions

**Le plan directeur de fabrication**

Son rôle est de prévoir à partir des données de type commerciale et technique, et de fixer les impératifs de la production. Ces données sont en particulier les commandes à respecter. A partir de ces commandes sont établies les prévisions de production.

**Gestion des données**

Ces données caractérisent les différents composants de la production tels que les machines ou les pièces. Elles sont :

- relatives aux produits
  * description technique,
  * description sur ces produits,
- relatives aux gammes opératoires
  * description des opératoires,
  * description de l'enchaînement des opérations,
- relatives aux moyens
  * machines
    - capacité,
    - disponibilité,
    - état,
  * outils
    - famille,
    - état,
    - disponibilité,
  * documentation technique,
  * moyens en hommes,
  * historiques.

**Suivi de production**

Ce sont les fonctions de mise à jour du système d'information tout au long de la fabrication. Il s'agit d'intégrer rapidement, toutes informations majeures concernant l'état de la production, des machines, des outils et des hommes.

**Le calcul des besoins**

C'est le calcul tout d'abord des besoins bruts, c'est à dire le besoin des composants nécessaires à une date due. Ensuite à partir de ces informations sont calculés les besoins nets en tenant compte des stocks existants qui peuvent indiquer ce qu'il faut produire exactement.

**Gestion des stocks**

Une bonne gestion de stocks doit permettre de les minimiser tout en assurant un flux continu de production et en évitant des ruptures de stocks, tant au niveau des produits que des pièces détachées ou des composants intervenant dans la fabrication.

Les différentes fonctions se réfèrent à la définition de la politique de gestion, la définition des produits, l'approvisionnement afin de déterminer comment approvisionner, et quoi approvisionner.

La gestion de stocks nécessite une analyse aiguë des besoins, de façon à maintenir un niveau de stocks stable. Des stratégies sont proposées pour son administration. Par exemple, se pose le problème de l'approvisionnement où il est possible, soit de remplacer ce qui sort, soit de ne tenir compte que des besoins réels.

Ceci influe sur le coût réel des stocks et influent sur la politique d'action.

**Le MRP** (Material Requirement Planning) est la méthode de gestion de stocks la plus connue. Elle se base sur le calcul des besoins nets en composants nécessaires à la fabrication, en tenant compte des données commerciales. **MRP II** améliore le calcul des besoins en intégrant les capacités du système de production.

### II . 8 . 5 . 3 - Ordonnancement/lancement

Cette fonction est sûrement la plus complexe et fait actuellement l'objet de multiples recherches. Plusieurs niveaux d'ordonnancement sont possibles, liés aux niveaux de planification associés.

Les opérations d'ordonnancement à court terme se rapprochent plus des fonctions de pilotage de l'usine. Cela consiste à calculer les affectations de tâches aux ressources existantes. L'ordonnancement vise à déterminer à un instant donné, la meilleure pièce à charger sur une machine disponible à partir d'un algorithme compatible avec les délais demandés et les moyens mis en oeuvre. Il faut de plus maximiser le temps d'engagement des machines.

Il faut distinguer l'ordonnancement en temps réel ou en ligne, et hors_ligne. Il est nécessaire d'avoir un modèle qui permettra d'effectuer les calculs.

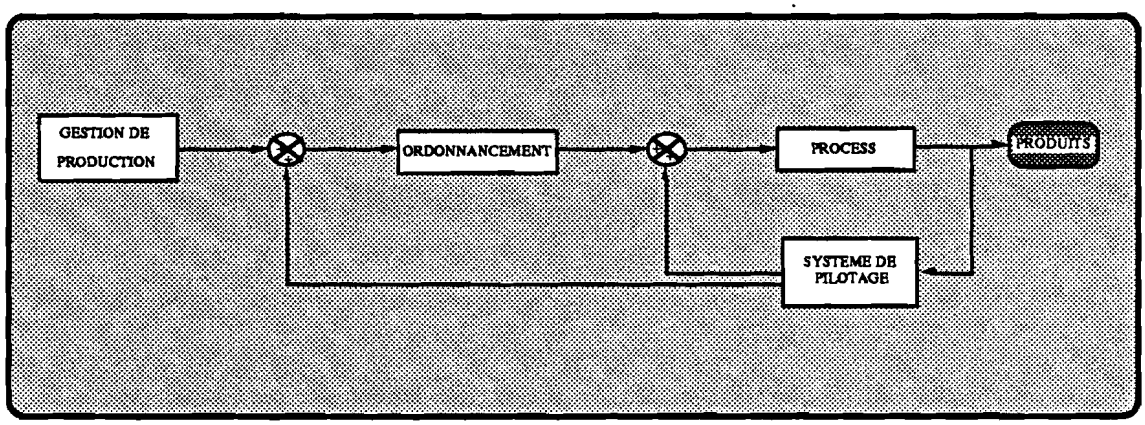



fig 5 : place de l'ordonnancement

Une méthode de lancement est **la méthode KANBAN** qui a pour principe d'analyser l'état des postes pour le lancement des ordres de fabrication. La production en amont n'est déclenchée que lorsque le poste en aval est libre. La production est dite à flux tiré et contrôlée par l'évolution de la demande. Ceci permet de tenir compte rapidement des perturbations possibles de la cellule.

**II . 8 . 6 -** CAO (Conception Assistée par Ordinateur)

Les systèmes de CAO mécaniques traitent des données fournies sous forme de représentations graphiques manipulées de manière interactive, avec les avantages suivants :

- représentation effective de l'objet,
- simulation possible de l'objet ce qui permet de faire des comparaisons,
- concrétisation des résultats théoriques.

**II . 8 . 7 -** MAO (Méthodes assistées par ordinateur)

Cela prend en compte :

- la spécification des opérations et leurs agencements, leurs déroulements,
- l'évaluation des coûts d'une pièce ou d'un produit.

### II . 8 . 8 - IAO (Ingénierie assistée par ordinateur)

L'IAO s'occupe de l'évolution quantitative et qualitative du processus de fabrication. Il offre les possibilités de simuler le système de façon à pouvoir évaluer les performances et le comportement du flux, et de détecter les éventuelles incohérences et blocages.

De plus, il permet l'analyse structurelle et la prévision de la durée de vie des différents éléments.

### II . 8 . 9 - FAO (Fabrication Assistée par Ordinateur)

Elle permet d'optimiser éventuellement les coûts de fabrication à partir de l'analyse des différentes solutions proposées. Elle permet, de plus, de produire de manière complètement automatique les documentations et programmes relatifs aux pièces.

Deux types d'applications sont possibles :

- **en ligne** : c'est le contrôle et la surveillance informatisée des tâches.
  A partir des informations se rapportant à la fabrication de la pièce, l'ordinateur évalue le résultat des opérations pour déterminer ou non la demande d'une correction. Il peut éventuellement déterminer ou diagnostiquer la présence éventuelle de pannes sur une des machines,
- **hors_ligne**, pour la génération des fichiers.

La FAO permet [ING 88] :

- un chargement optimal des machines avec une utilisation plus grande et une meilleure gestion des outils avec la sélection des meilleurs outils,
- une augmentation de la productivité avec une meilleure qualité des programmes de contrôles numériques,
- une amélioration de la qualité des pièces.

Un avantage acquis grâce à l'informatique est la capacité de stockage des programmes à commande numérique sans qu'il soit forcément nécessaire de perdre du temps à les télécharger. La capacité mémoire est suffisante pour stocker les programmes nécessaires. De plus, cela permet une gestion locale des fichiers.

A chaque pièce correspond un parcours particulier avec l'ordre pré_établi des opérations et un montage adapté.

### II . 8 . 10 - Exemple de génération d'un programme de commande

On trouve dans l'unité de contrôle du robot un fichier directement exploitable. Cela nécessite une mise en forme de la tâche CAO qui peut se faire à 3 niveaux :

i génération d'un fichier intermédiaire, stockant les positions cartésiennes, les vitesses, les codes de fonctions. Le fichier est ensuite traité par le post_processeur du robot,
ii génération d'un fichier de valeurs codées,
iii traduction du programme graphique dans le langage de programmation du robot à commander. Des constructeurs ont des interfaces en général, avec les principaux systèmes de CAO.

## III - LES OUTILS D'AUTOMATISATION

### Introduction

L'intégration n'est rendue possible que parce que les différents matériels utilisés sont techniquement assez performants. Ils sont en particuliers, plus "ouverts" vers l'extérieur et plus "intelligents". Ils sont donc beaucoup plus aptes à la communication, facteur-clé d'une production en contexte CIM. Le développement des réseaux a modifié considérablement les caractéristiques des systèmes de production.

Nous décrivons les différents outils majeurs utilisés sous un aspect CIM, en particulier les réseaux, les API et les robots.

### III . 1 - Les réseaux

#### Introduction

Aujourd'hui, l'automatisation des usines passe par l'utilisation de réseaux informatiques. L'interconnexion de matériel devient essentielle pour l'échange d'information entre applications qui sont multiples dans les domaines de : la gestion de l'usine, le suivi de fabrication, la conduite de processus, la commande d'automatismes, ou la CAO/FAO.

Souvent l'usine comprend deux ou plusieurs réseaux locaux, un réseau étant affecté à chaque niveau de structure avec des passerelles entre niveaux.

La notion d'informatique répartie nécessite l'utilisation de réseaux informatiques et il faut les considérer comme des entités à part entière des machines de l'usine. Les conséquences

d'une panne d'un réseau sont aussi importantes, sinon plus, que la panne d'une machine de fabrication. Heureusement la fiabilité d'un réseau est certaine, et souvent les utilisateurs prévoient un réseau de secours.

Les principaux caractères d'un réseau, qui doivent être mis en évidence sont une fiabilité et une sécurité importante, la capacité à intégrer du matériel hétérogène, une capacité de transfert élevée, liée à une vitesse de transmission élevée. Ce sont d'ailleurs des caractères incompatibles.

Nous nous intéresserons plus particulièrement aux réseaux locaux, les autres réseaux étant plus dédiés à des connexions entre entreprises. Les réseaux locaux sont l'outil par excellence de communication interne aux usines

La définition d'un réseau local est liée à l'implantation physique de réseau. Il reste restreint à des communication inférieures à quelques kilomètres (usines, administration).

### III . 1 . 1 - Pourquoi des réseaux différents

Dans de nombreuses entreprises, les réseaux sont très hétérogènes par leurs utilisations à des niveaux différents de l'usine. Les machines interconnectées sont de plus, à un même niveau, souvent de marques différentes et incompatibles entre elles. Les progiciels sont développés au fur et à mesure des demandes. D'une manière générale, chaque type de réseaux assure des fonctions à des niveaux différents de l'entreprise et répondent à des contraintes différentes, physiques (parasites électriques ou électrostatiques ; saleté ; bruit) ou contraintes fonctionnelles (temps réel ou différé ; hétérogénéité des matériels utilisés).

Il est possible de distinguer trois sortes de réseau local :

- réseaux d'entreprise,
- réseaux d'ateliers,
- réseaux de terrain.

### III . 1 . 2 - Description des différents types de réseaux

L'hétérogénéité est ce qui caractérise les réseaux locaux, tant au niveau des informations qu'au niveau du matériel connecté. Les informations véhiculées par un réseau de terrain sont fondamentalement différentes de celles véhiculées par un réseau d'entreprise et ces réseaux interconnectent des machines totalement différentes (par exemple : capteurs et service de gestion). L'hétérogénéité provient aussi des marques de constructeurs de matériel

informatique : souvent l'évolution de l'informatisation de l'usine intègre peu à peu du matériel d'origines diverses pour répondre aux nouveaux impératifs de production.

- **les réseaux d'entreprise** sont dédiés aux communications entre tâches plus orientées vers la bureautique, la messagerie.

- **les réseaux de terrain** font communiquer les capteurs et actionneurs avec les automates ou commandes numériques.
  Le débit est faible car les informations transportées sont de l'ordre de quelques bits par capteurs. Le trafic est déterministe avec un balayage acyclique des capteurs et des actionneurs. Les informations gérées sont donc d'étapes périodiques. Ce flux d'information est appelé **FIP** (Flux d'Information en provenance de et vers le Processus).

- **les réseaux d'ateliers** sont à un stade intermédiaire entre les réseaux d'entreprise et de terrain. Ils permettent le passage des informations de milieux techniques et industriels vers le niveau gestion de production. De par sa position, le réseau d'atelier doit être nécessairement capable d'absorber des informations très différentes : états de machines, de capteurs, niveaux de stocks, avancement de production... Un bon réseau doit être flexible et ouvert au niveau de ses informations transportées. Chaque machine étant très spécialisée dans sa fonction, le réseau d'atelier doit connecter du matériel très hétérogène. De plus, le flux d'information est beaucoup plus apériodique. Il est essentiel dans la gestion et dans le pilotage de l'usine que ce type de réseaux soit rapide et fiable.

Cependant ces différents types de réseaux peuvent offrir des services extérieurs à leurs cadres initiaux. Certains équipements orientés vers la bureautique, sont utilisés comme réseaux industriels mais il résulte souvent des problèmes de communications dus à des parasites occasionnés par les machines industrielles.

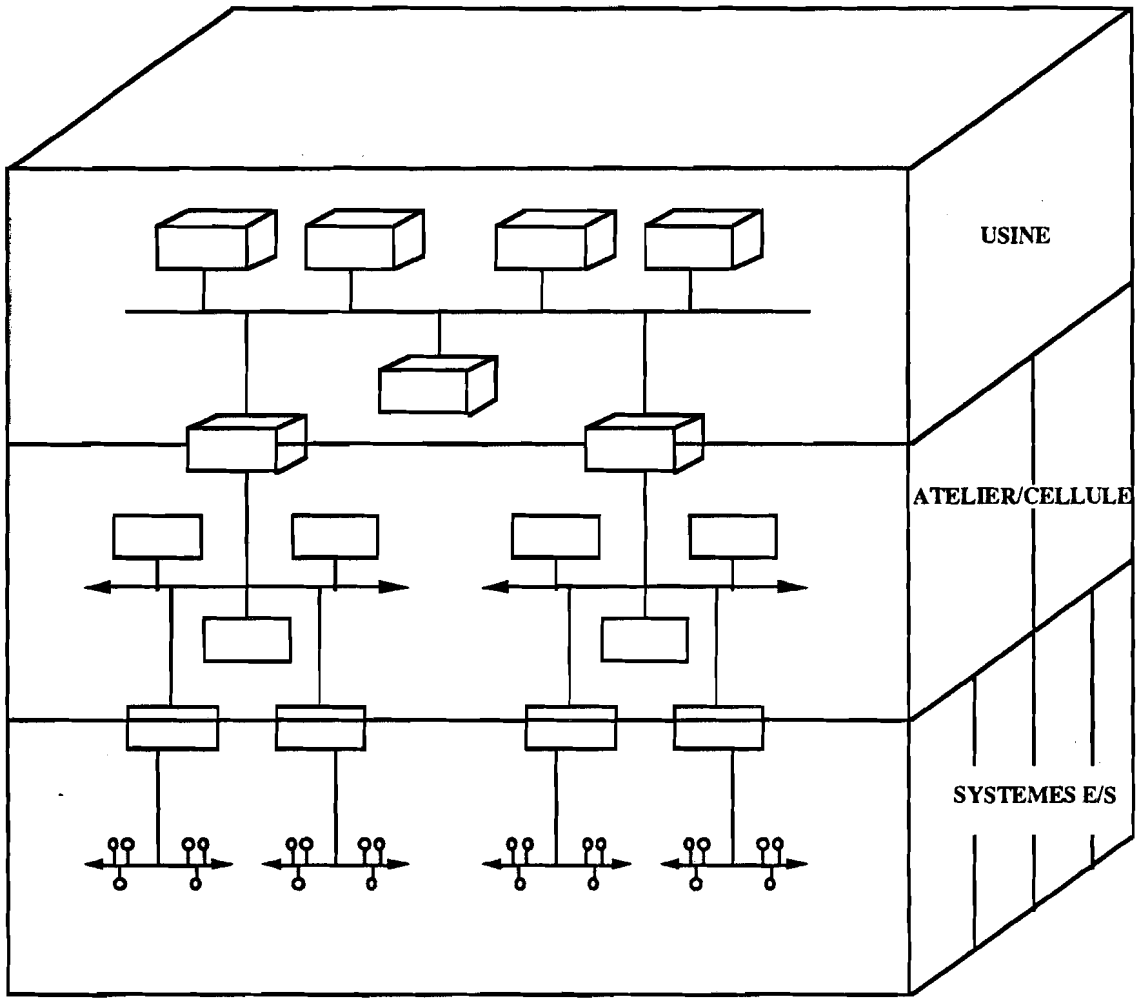



fig 6 : usine vue par les réseaux [COM 89]

**Nécessité d'une norme**

Dans ce secteur, plus qu'ailleurs, il est nécessaire de normaliser les topologies et les supports de transmission qui sont très spécifiques aux applications. Une référence est nécessaire pour normaliser l'interconnexion des différents systèmes.

Le modèle OSI précise des normes et des protocoles entre les équipements à respecter. Un système est dit ouvert s'il respecte ces normes. La norme OSI est un modèle en couches. Chaque couche est responsable d'obligations de nature très diverses vis à vis des couches de niveaux supérieurs et inférieurs mais aussi vis à vis des couches de mêmes niveaux. Ces obligations peuvent concerner le type des supports de transmission, le type et le format des données. 7 couches sont définies :

| Couches | | fonctions |
|---|---|---|
| N° | Nom | |
| 7 | Application | Elle définit les mécanismes communs utiles aux applications réparties et la signification des informations échangées. |
| 6 | Présentation | Elle définit la forme des informations échangées : structure, syntaxe, codage. |
| 5 | Session | Elle fournit des outils de synchronisation et de gestion du dialogue entre entités communicantes. |
| 4 | Transport | Elle fournit des moyens de transport d'information d'un bout à l'autre entre deux utilisateurs situés dans des systèmes différents, indépendamment des caractéristiques du ou des réseaux réellement utilisés. |
| 3 | Réseau | Elle réalise l'acheminement des informations au travers du réseau pour des modules non directement connectés. |
| 2 | Lien ou liaison | Elle permet le transfert fiable d'information entre des systèmes directement connectés. |
| 1 | Physique | Elle décrit les interfaces mécaniques et électriques pour l'échange des signaux porteurs des informations. |

fig 7 : norme OSI [LEP 89]

Le problème de cette norme est qu'elle n'est pas adaptée aux services temps réel, par la superposition de multiples couches, d'où introduction de nouvelles approches MAP et TOP, plus simples.

Pour faciliter l'intégration de matériel hétérogène, les constructeurs se sont mis d'accord sur des normes à respecter. Deux protocoles apparaissent actuellement MAP et TOP et sont en voie de normalisation.

**MAP** (Manufacturing Automation Protocol) développé par Général Motors, et **TOP** (Technical Office Protocol) sont en fait des dérivés de la norme de modèle OSI en adaptant les différentes couches aux problèmes spécifiques rencontrés. Ces projets sont soutenus par les groupes d'utilisateurs tant aux Etats unis qu'en Europe afin de contribuer à la normalisation et à la promouvoir. Cependant, MAP et TOP restent attachés à la norme OSI pour l'interconnexion de systèmes ouverts. MAP prévoit un maximum de protocoles d'échange dans le but d'être totalement ouvert aux ateliers, plus rapides mais simplifiés, (Mapway version développée par la Télémécanique).

MAP et TOP sont pratiquement identiques sauf au niveau des couches basses et hautes du modèle OSI et vont devenir rapidement intégrés ensemble (publication de MAP/TOP 3.0) avec possibilité de faire communiquer des machines totalement hétérogènes.

D'autres protocoles existent, beaucoup plus adaptés aux problèmes de communication industrielle en temps réel tels que les protocoles MODBUS et JBUS : ils permettent d'interconnecter des automates industriels, des terminaux d'ateliers et des ordinateurs. Ces réseaux sont utilisés dans la commande et le pilotage d'ateliers. Cependant leurs protocoles, qui sont de type maître esclave, ne sont pas très performants.

**Le projet FIP** prévoit l'interconnexion de matériel de bas niveau : capteurs et actionneurs, cela permettrait d'éviter la multiplication de liaisons série de type RS232 utilisable pour la connexion avec les capteurs et de remplacer ainsi les liaisons points à points.

Les réseaux utilisant des techniques d'accés avec jeton (Token Ring) autorisent leur emploi dans un contexte déterministe à la différence de réseaux employant une technique de type CSMA/CD (Ethernet)

Des réseaux industriels **LAC/LAC2** [COM 89] sont des solutions possibles. Le protocole est de type CSMA/CD et les débits d'information sont beaucoup plus rapides. De plus il est possible de relier toutes sortes de machines, du capteur au gros ordinateur. De nombreuses passerelles sont possibles vers le monde SNA d'IBM et Unitelway de la Télémécanique.

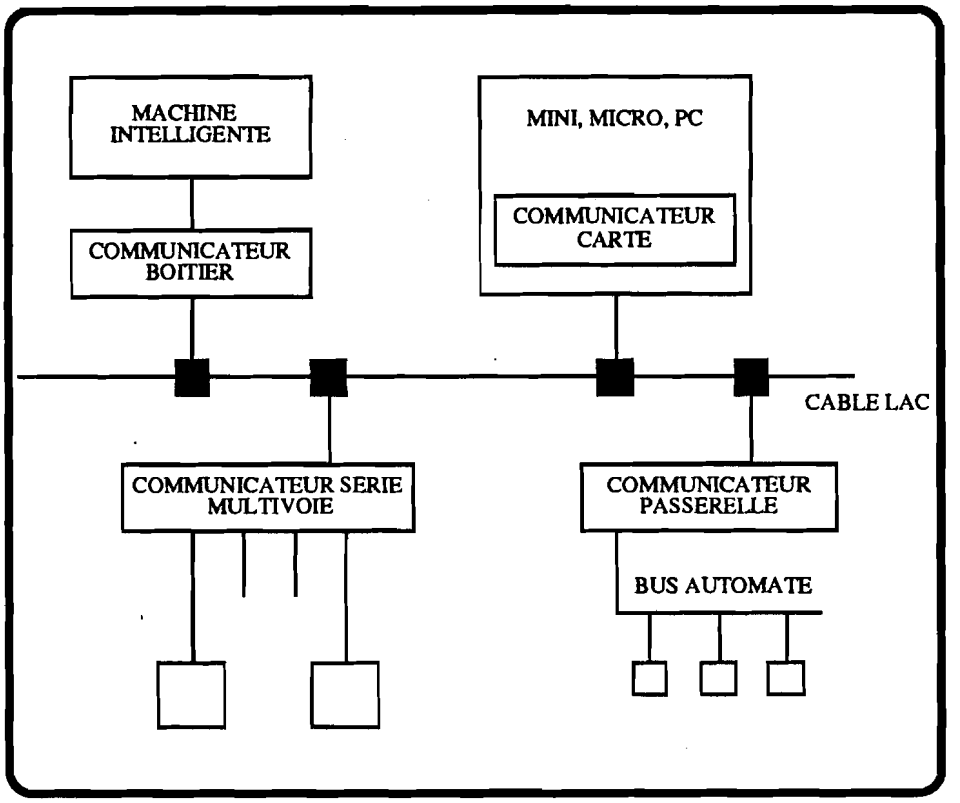



fig 8 : exemple de réseau LAC [COM 89]

### III . 2 - Les automates programmables [CHA 87]

A l'origine, les automatismes étaient essentiellement de type logique ou séquentiel. Leurs fonctions étaient peu adaptables à l'évolution du matériel et du processus. Ils sont parvenus actuellement à un stade intermédiaire entre cet état d'origine et un calculateur classique. Ils conservent toujours la possibilité de programmer des automatismes mais ils peuvent d'autre part, communiquer avec l'extérieur ou assurer des tâches plus intelligentes. En effet, leurs langages de programmation sont beaucoup plus performants et adaptables. La qualité des interfaces de communication est devenue certaine et permet l'ouverture vers d'autres automates ou d'autres mondes. Par exemple, la gamme d'automates proposée par TELEMECANIQUE est particulièrement significative de l'évolution des automates. Le premier langage proposé fut le langage à relais, suivi par le langage Grafcet et littéral. La puissance de calcul a notablement augmenté. Des coupleurs intelligents sont proposés pour la commande de matériel et autorisent la communication par des réseaux, adaptés évidement aux milieux industriels. La convivialité est meilleure qu'auparavant.

#### III . 2 . 1 - Les avantages des API

Leurs possibilités au niveau des entrées est grande en nombre et en variété. Ils peuvent être installés près de l'endroit des machines à commander d'où la nécessité d'être capables de

supporter les contraintes d'un environnement industriel ingrat (problèmes de pollution, perturbations électriques). Leur utilisation est souple : programmation en ligne, possibilité de mise au point dynamique et programmation simplifiée. Elle ne nécessite pas de connaissance informatique spéciale et est directement exploitable par l'automaticien. Par contre, elle est limitée dans la complexité de programmation.

### III . 2 . 2 - Les classes des API

Les classes d'automates sont nombreuses et adaptées aux différentes applications industrielles.

Une première classe concerne ceux qui nécessitent un nombre important d'entrées/sorties, lié à de sérieuses capacités de calcul.

Une seconde classe implique les automates destinés à des tâches fonctionnelles bien précises. Certains sont dédiés au contrôle de l'évolution séquentielle de processus. D'autres par exemple, tels que les commandes numériques, consistent à commander une machine_outil par un programme et non par un opérateur, avec en particulier, un accès aux données internes des machines, de manière transparente.

Cette diversité d'automates implique l'usage de plusieurs langages et de fonctions, différentes et spécifiques.

La systématisation des automates autorise le pilotage global par un ordinateur. La conception possible des programmes en CAO, sur un ordinateur plus puissant, libère ces machines des tâches pénibles de conception avec un gain de temps et de précision.

### III . 2 . 3 - Cas des machines-outils

L'avantage des machines outils sous commande numérique est qu'elles s'adaptent facilement au travail atelier et à la petite fabrication groupée car elles peuvent être facilement reprogrammées. Ces machines répondent à des contraintes d'automaticité (gestion automatique des outils), de fiabilité ou encore de précision. Les NUM sont connectées à un ordinateur central qui stocke les programmes avant téléchargement en cas de besoin. L'organisation, de type DNC (Direct Numerical Control) permet, de plus, un suivi en temps réel de l'avancement des travaux et de relier plusieurs MOCN entre elles.

## III . 3 - Les systèmes de manutention

La flexibilité au niveau du routage et du transport de pièces passe par l'utilisation de systèmes de manutentions performants et surtout flexibles. Ils visent à charger, décharger les pièces. Nous nous intéresserons plus particulièrement aux robots car très utilisés.

Le choix d'un processus de manutention est vaste :

- les chariots filo-guidés, très souples d'utilisation,
- les chariots sur rails, assez lourds de fonctionnement,
- les manipulateurs, les télémanipulateurs,
- les convoyeurs à chaînes.

Le choix se fait souvent en tenant compte des spécificités de chaque appareil. Souvent, ces systèmes de transport servent de lieu de stockage et cela peut jouer au niveau de la sélection.

### III . 4 - Les robots

Les robots deviennent de plus en plus performants tant au niveau de leurs possibilités, qu'au niveau de leur autonomie. Ils intègrent de plus en plus de capteurs, de systèmes de reconnaissance et de meilleures fonctions de manipulation. Ils sont donc plus facilement intégrables à l'usine (flexibilité).

#### III . 4 . 1 - Avantages des robots

Les avantages des robots sont multiples d'un point de vue technique et fonctionnel.

L'intérêt des robots par rapport à un opérateur humain est sa faculté à travailler en milieux hostiles, dus en particulier à la chaleur ou au bruit. Ces conditions sont difficilement supportables par un humain à l'inverse d'un robot. La construction de robots nécessite donc l'étude de leurs milieux de travail. En effet, la multiplicité des dangers doit rendre le robot approprié et spécifique à son milieu. Par exemple, il peut être nécessaire de rendre un robot ininflammable de façon à ce qu'il puisse travailler à température élevée. L'utilisation d'un robot permet d'améliorer la sécurité.

De plus, la simplification et la programmation du contrôle offre une amélioration de la qualité de la tâche et par conséquent, une amélioration de la qualité du produit.

L'introduction des robots favorise la flexibilité du système de production et la réduction des coûts de production, d'où une augmentation de la productivité.

#### III . 4 . 2 - Disposition de la cellule vis à vis de robots [ENG 81]

L'importance des robots est si grande qu'il faut optimiser l'agencement du robot dans l'atelier et le travail peut être conçu de cette manière :

- disposer les machines de travail autour du robot,
- apporter du travail au robot,
- faire passer le travail devant le robot,
- amener le robot au travail.

Les zones d'activités nécessaires sont disposées dans la zone de travail du robot (disposition la plus utilisée). Le robot a la charge de plusieurs machines-outils à des endroits différents (exemple courant : robot monté sur un rail). Le robot doit pister la pièce ou le produit à transformer lors de son transfert. Cela permet de garder un transfert en cours et de limiter les opérations de déchargement/chargement

### III . 4 . 3 - La surveillance du robot

La qualité s'est nettement améliorée, liée à l'augmentation des possibilités de surveillance telles que la régulation, les techniques de recherches ou la commande par retour d'efforts.

La commande par retour d'efforts est la plus classique et consiste à prendre en compte les interactions de contact entre un robot et son environnement pour lui faire exécuter une tâche donnée. On la distingue de la commande en force où l'on contrôle les couples moteurs des actionneurs en prenant en compte la dynamique du système : les interactions de contacts sont supposées largement prépondérantes par rapport aux autres forces auxquelles peut être soumis le manipulateur (forces inertielles par exemple).

## IV - METHODOLOGIE CIM

### IV . 1 - Besoins méthodologiques

La mise en place d'une structure CIM repose sur une démarche classique, mais qui est rendue plus complexe par les multiples aspects concernés.

Les étapes de la démarche d'intégration sont classiques et répondent aux conditions imposées par une gestion de projet correcte. Cependant, le nombre de paramètres est élevé et de nature hétérogène. La démarche comporte 4 étapes:

- spécifications des besoins,
- mise au point d'un plan directeur,
- intégration,

- informatisation progressive.

Le plan directeur est en fait l'ensemble des règles à respecter lors de l'informatisation.

**Notion de conduite de projet :** la conduite de projet est fondamentale dans la réussite des objectifs. Cela permet de déterminer exactement la tâche de chacun et de vérifier que chaque phase du projet converge vers les objectifs. Cela permet de maîtriser l'avancement du projet en terme de coûts et de délais fixé par un plan.

### IV . 2 - Nécessité d'une méthodologie

Il est possible d'automatiser une partie des procédures existantes. Cependant cela rajoute beaucoup plus de problèmes que cela n'en résoud, de par la complexité du nouveau système.

Le CIM repose sur le fait que tous les outils de conception par ordinateur doivent pouvoir communiquer à n'importe quel stade de vie d'un projet, ceci pour faciliter la diffusion du produit et sa production.

Evidemment cela entraîne des coûts supplémentaires au départ, mais qui seront amortis à long terme par la baisse des prix généraux.

Souvent, il n'existe pas de méthodes bien définies pour concevoir un système d'information. En général, ils sont réalisés au coup par coup suivant les applications.

L'absence de méthodes est due aux frais de recherche, frais qui ne peuvent être supportés que par de grandes entreprises.

Nous allons nous intéresser aux méthodologies de conception globale de l'entreprise et celles plus spécialisées de l'usine. Une méthodologie de conception doit s'appliquer d'une part aux fonctions de gestions de production et d'autre part aux fonctions dites plus techniques.

La démarche générale reste identique à celle d'un projet informatique, mais évidement elle est orientée vers la conception de systèmes automatisés avec les contraintes et les impératifs inhérents à ce domaine. Une réflexion productique est l'étape obligatoire de spécifications des objectifs et des contraintes à respecter.

Une démarche habituelle dans la conception d'un système industriel de commande, est la réalisation de la partie opérative suivie de celle de la partie commande. Cependant, cette démarche n'exploite pas au maximum les possibilités de simulation et d'évaluation qu'offrent par exemple les outils informatiques. La partie opérative ne peut être implantée qu'à la suite de la validation de la partie commande. L'élaboration de la partie commande reste identique à celles appliquées lors d'une démarche génie logicielle. Les différentes étapes telles que spécifications, étude, simulation et validation sont communes.

Une méthodologie globale est nécessaire. En effet, une méthodologie s'impose et est l'étape obligée pour concevoir un système d'information et de pilotage. Cela permet de prendre en compte toutes les contraintes, budgétaires ou technologiques, pouvant intervenir.

Ces prises en compte initiales évitent de redéfinir complètement la structure et l'environnement de l'entreprise en cours lors de changements de stratégie avec limitation des frais.

### IV . 3 - Spécificité des systèmes flexibles [BON 85]

Ils nécessitent une méthodologie de conception plus marquée. Les contraintes à prendre en compte restent particulières : gammes opératoires, temps d'usinage, type des produits, caractéristiques et potentiel des machines. Cela va déterminer le type d'architecture et les solutions de pilotage.

Différentes phases sont distinguées :

- spécifications,
- analyse et conception,
- développement d'un modèle et évaluation,
- implantation,
- exploitation,
- maintenance.

De façon à avoir une base de travail cohérente, un modèle de représentation s'impose qui permettra par la suite de valider et de mesurer les résultats obtenus. Ce modèle peut avoir plusieurs niveaux d'abstraction par rapport à l'évolution et à la nature du projet.

### IV . 4 - Phase de spécifications

L'étude débute par une phase de spécifications, travail d'ailleurs assez long car il nécessite les connaissances de nombreuses personnes.

Cette phase permet de recueillir les informations spécifiques à la réalisation du système. Les informations sont très spécifiques : par exemple, il peut s'agir des gammes opératoires, de types de pièces produites. Pour un système de gestion, il peut s'agir des logiciels et de la structure des logiciels à utiliser. Les objectifs de cette phase de spécifications sont tout d'abord de fournir des supports de documentation qui serviront de base tout au long du projet, avec le rejet des ambiguïtés. Cela peut aussi mener à un prototypage rapide du système.

## IV . 5 - Méthodes d'analyse et de conception

La phase initiale, étape d'analyse et de conception est la plus importante avec la structuration du système et son analyse fonctionnelle. L'analyse sert à préciser les différentes tâches et fonctions de la cellule jusqu'à obtention des tâches élémentaires. Cette analyse peut être associée à une étude de faisabilité pour vérifier la cohérence du projet. Il ne faut pas oublier de prendre en compte le produit. En effet, c'est lors de l'étape de conception que doivent être intégrés les problèmes liés à la fabrication ou l'assemblage, par le fait d'utiliser de préférence, des pièces standardisées, ce qui influe notablement sur les gammes opératoires finales.

L'analyse du processus de fabrication reste primordiale. Le système de fabrication doit être capable de construire des pièces, non encore prévues. Il faut en déduire les modes de fonctionnement les plus variés possibles, dans le cadre des tâches de stockage, fabrication ou transport. Cela permettra de regrouper ainsi les opérations successives.

### Importance d'une démarche globale

De façon à rationaliser et systématiser une démarche d'implantation de système de gestion de production, une démarche ordonnée s'impose.

Une démarche classique de conception consiste à générer l'organisation fonctionnelle à partir des moyens de production déjà existants. Au contraire, une démarche plus rationnelle serait de tenir compte des possibilités technologiques, des objectifs de production pour proposer une organisation matérielle et logicielle suivie d'une implantation effective. Cette démarche doit intégrer tous les aspects de l'usine.

Une approche génie logiciel s'impose. Elle permet de résoudre les problèmes globalement. Une approche globale permettra de simplifier la structure générale en évitant une trop grande hiérarchisation, frein à la flexibilité envisagée. Les méthodes d'analyse permettent de passer du système réel à un modèle structuré et donc décomposé de ce réel.

Tout d'abord une analyse descendante dégage la structure hiérarchisée du système de conduite. Elle permet de mettre en relation chacun des modules de décision dégagés.

Cette démarche méthodologique permettra de générer une ou plusieurs solutions de schéma de structures d'usines. Des modèles sont proposés et sont éventuellement simulés de façon à savoir si le cahier des charges a été respecté et si la solution n'aboutit pas à des incohérences de conception.

### IV . 6 - Phase de simulation

Phase importante du cycle de vie d'un projet, elle permet d'étudier et de valider le modèle obtenu de façon à vérifier son exactitude vis à vis du cahier des charges initial. Les systèmes de CAO sont une aide précieuse, car capables de modéliser et d'évaluer très rapidement plusieurs solutions possibles. Les résultats de la simulation peuvent mener à revoir les spécifications initiales, par un retour arrière dans la conception, dans le cas où les performances seraient insuffisantes.

Une simulation idéale permet d'arriver progressivement vers une solution optimale. Un affinement des différents composants du système conduit vers un dimensionnement et une définition des algorithmes de commande. Plusieurs niveaux de simulation sont proposés [BEN 85] :

- dimensionnement des grandeurs caractéristiques des postes de travail,
- affinage des résultats et dimensionnement des moyens de manutentions,
- optimisation fine du fonctionnement en temps réel de l'atelier,
- simulation du pilotage de l'atelier en fonctionnement normal.

### IV . 7 - Phase d'implantation

Ensuite les phases de réalisation et d'exploitation permettent de valider la conception.

La solution finale n'est acceptée qu'après la validation et le respect du cahier des charges avec la garantie d'une solution viable.

La mise en route est une opération délicate car généralement, c'est à ce stade de la démarche qu'est détectée la majorité des erreurs possibles de fonctionnement.

Les bénéfices réalisés après une démarche globale sont par exemple [ING 88] :

- la réduction du coût de la conception technique de 15 à 30 %,
- la réduction du délai d'industrialisation de 30 à 60 %,
- l'augmentation de la productivité des opérations de fabrication : 40 à 70 %,
- l'augmentation de la rentabilité des investissements de 2 à 3 fois,
- la réduction des en_cours de 30 à 60 %,
- la réduction des coûts de personnels.

**IV . 8 - Sécurité, sûreté et fiabilité** [NUS 88]

Tout système moderne de production doit être doté d'un système de détection de défauts et de surveillance afin de pallier et de prévenir les défaillances possibles. Celles-ci peuvent être dues à des erreurs d'installation, de manipulation ou de commande.

Les causes de défaut sont par exemple dues à :

- une erreur de conception du système : le cahier des charges n'ayant pas été respecté,
- un défaut d'implantation : il peut se produire lors du passage du modèle à l'implantation effective,
- une défaillance de composants due à des phénomènes d'usure,
- des perturbations extérieures qui peuvent entraîner des erreurs.

La sûreté de fonctionnement est donc obligatoire dans la commande d'un processus. Cette notion de sûreté s'associe à celles de sécurité et de disponibilité.

**La sécurité** est une notion plus restrictive que la sûreté : le but est l'absence totale de circonstances susceptibles d'occasionner une action irréparable (blessure, destruction définitive d'un équipement présentant un danger...).

**La disponibilité** est relative au temps d'utilisation possible d'une machine, ce qui dépend de sa fiabilité et de sa maintenabilité. La fiabilité est aussi la probabilité pour qu'un équipement accomplisse une fonction requise dans des conditions précises pendant une durée déterminée. La maintenabilité permet de réagir à une défaillance.

La fin d'une période de bon fonctionnement est marquée par une panne due à une défaillance et par des opérations de reprise. Celles-ci ont pour but de rétablir un fonctionnement normal après l'apparition d'un défaut.

En général, la fiabilité du système de pilotage est très importante par rapport à celle du matériel. Après la période de tests et de mises en route, un système de pilotage tombe rarement en panne. Ceci est d'ailleurs vital de par l'importance de son rôle. D'ailleurs, un système de pilotage bien conçu autorise une certaine autonomie de fonctionnement en cas d'incidents ou de difficultés.

les techniques pour améliorer la fiabilité sont:

- l'amélioration des méthodes de développement,

- le contrôle de la qualité des composants,
- la mise en place d'une redondance des éléments,
- la détection des défauts de manière rapide = > reconfiguration.

**Fiabilité du logiciel**

C'est un problème différent car il n'est pas possible de parler d'usure du logiciel. Les problèmes peuvent venir surtout d'une erreur de conception ou du non respect des spécifications de départ. Ces erreurs sont d'ailleurs très difficiles à détecter et à envisager.

La prise en compte de ces défaillances logicielles est évitable pour peu que l'on utilise une démarche de conception rigoureuse, avec différentes étapes successives d'évaluation et de vérification du logiciel.

## IV . 9 - Méthodes existantes

### IV . 9 . 1 - Présentation de quelques méthodes

Les solutions les plus souvent retenues sont le modèle SADT, les réseaux GRAI ou la méthode IDEF-0. Ces méthodes engendrent des architectures d'ateliers adaptées aux besoins de l'exploitation avec l'intégration de parties opératives structurées, cependant avec quelques particularités.

Le modèle IDEF-0 est retenu en général pour étudier le système de fabrication.

Les réseaux GRAI sont utilisés dans la formalisation de systèmes décisionnels.

#### IV . 9 . 1 . 1 - SADT [ROS 77]

SADT est basée sur la notion d'analyse structurée de systèmes complexes. Cela permet de partitionner, de structurer et de présenter les besoins au moyen d'une représentation graphique (datagramme et actigramme). Elle ne prend en compte que les aspects d'organisation structurelle des activités et données associées de manière essentiellement statiques.

#### IV . 9 . 1 . 2 - Réseaux GRAI [DOU 84]

La méthode GRAI (Graphes à Résultats et Activités Intercalées) a été développée au laboratoire d'Automatique de l'Université de Bordeaux.

Cette méthode s'appuie sur un modèle conceptuel élaboré à partir des théories sur les systèmes hiérarchisés et des théories des systèmes d'organisation.

Elle comporte une "phase d'analyse et une phase de conception".

Le modèle de conduite regroupe un sous-système physique, un sous-système de décision et un sous-système d'information.

- le sous-système physique est en fait la décomposition structurelle du procédé,
- le sous-système de décision est une décomposition hiérarchisée des décisions en tenant compte de leur horizon de décision,
- le sous-système d'information est le lien entre les deux autres sous-systèmes.

### IV . 9 . 2 - Exemple de méthodologie de conception

**Le programme ICAM** (Integrated Computer Aided Manufacturing) : ce projet mené par le Ministère de la Défense Américaine a pour but de "faire avancer la frontière de technologie".

Les objectifs sont la promotion des outils de type FAO avec l'idée d'augmenter la productivité.

Le principe est de proposer des structures hiérarchisées de modules de fabrication reposant à la fois sur des structures logicielles et matérielles. Ces modules sont capables de gérer des fonctions de conception, de fabrication et de contrôle.

Une décomposition structurelle est réalisée : niveau station, niveau cellule, niveau machine.

Une méthode de représentation a été mise au point avec 3 langages :

- IDEF-0 - modélisation des fonctions,
- IDEF-1 - modélisation de l'information,
- IDEF-2 - modélisation dynamique pour la simulation de fonctions du système et les tests de validité.

Les critères de flexibilité, d'automatisation, de production en série limitée, de contrôle global sont intégrés au maximum.

Une normalisation des produits issus de cette méthodologie est actuellement réalisée.

#### IV . 9 . 2 . 1 - Présentation de IDEF-0

Cette méthode se rapproche de SADT. Elle a pour but de présenter une approche structurée des systèmes de production. Elle permet de délimiter le système à analyser (contexte, critère d'analyse, objectifs de l'analyse), de hiérarchiser les tâches réalisées par le système de façon à bien représenter chaque domaine d'activités.

Un modèle graphique est utilisé, ce qui permet une bonne compréhension générale.

La phase de structuration est une analyse descendante pour la construction du modèle à partir des informations fournies par l'utilisateur. Chaque activité est décomposée en activités plus précises.

Cependant, cette méthode exprime peu la notion de dynamique. Une solution pour y remédier est SPEX [TIX 89], combinaison de IDEF-0 et du Grafcet.

### IV . 9 . 3 - Classement des méthodes existantes [GOR 87]

Une décomposition des méthodes d'analyse des systèmes de production est possible suivant trois axes, la nature du modèle, le niveau d'abstraction et le cycle de vie du projet.

Il existe plusieurs méthodes qui s'attachent à ces problèmes et qui tentent de concevoir un système d'information : Merise, Axial, GRAI. Ces méthodologies s'intéressent surtout aux problèmes de la gestion globale d'une usine.

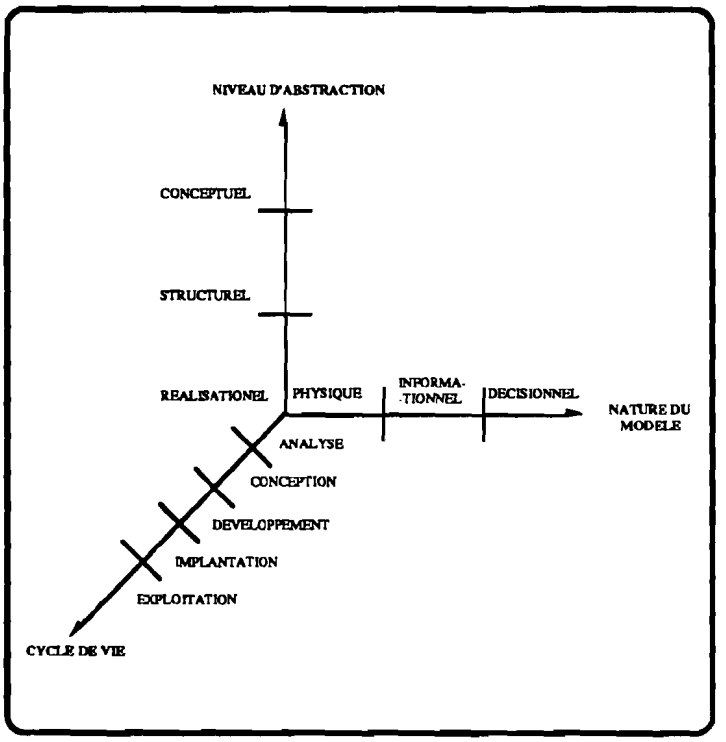



fig 9 : décomposition suivant trois axes

### IV .9 . 4 - Modèles informatiques de la commande [GOR 87]

Les méthodes existantes n'ont pas la possibilité de traiter chaque situation de la même manière. Une classification est proposée :

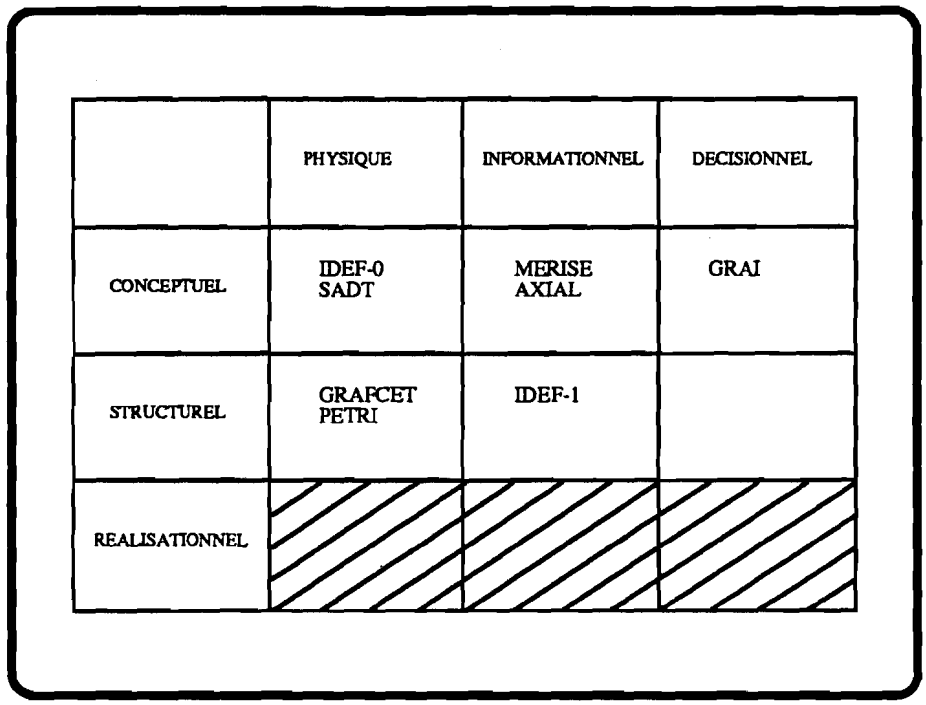



fig 10 : classification de modèles de commande

Cette classification permet de délimiter les différents domaines de modélisation. Pour modéliser complètement un système de production, il faut combiner plusieurs modèles (cas du CIM-OSA).

### IV . 9 . 5 - CIM-OSA

CIM-OSA (Computer Integrated Manufacturing - Open System Architecture) est une architecture proposée dans le cadre d'un Projet Esprit avec le but de fournir une approche globale pour la conception et l'implantation d'un système CIM.

Deux sous-domaines se dégagent [GAC 90] [CHE 90] :

- une partie modélisation : il faut pouvoir modéliser toutes les composantes d'un système industriel,
- une partie infrastructure : elle est destinée à l'implantation effective.

L'idée générale est de proposer un modèle général d'architecture de référence qui sera adaptable à des systèmes particuliers (**Instantiation**).

Différents modèles sont proposés, chacun d'eux étant dédié plus spécifiquement aux différents domaines de l'entreprise étudiée (**Génération**).

Ces domaines sont ceux de **l'Organisation, des Ressources**, de **l'Information** et des **Fonctions.**

**La Dérivation** modélise la description de l'implantation à partir tout d'abord de la modélisation des besoins de l'entreprise puis de celle des spécifications de modélisation.

## Cadre de modélisation de CIM-OSA

Ce cadre permet de définir les différentes phases et vues de modélisation d'un projet.

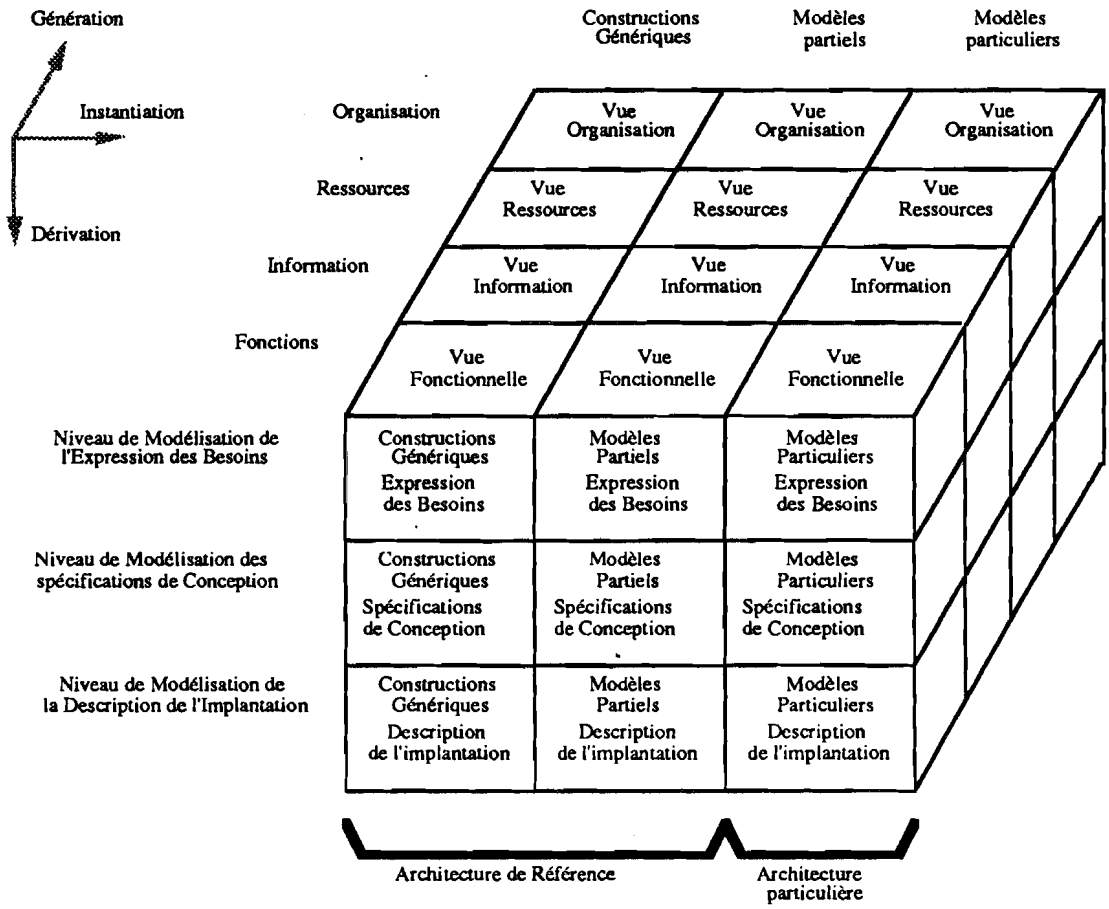



fig 11 : cadre de modélisation de CIM-OSA

Il existe peu de méthodologies pour la conception de graphes de commande et de coordination qui mettent en oeuvre les différentes fonctionnalités des machines, les types de pièces, leurs gammes opératoires...

Le laboratoire LAIL de l'EC Lille s'est attaché au développement d'une méthodologie de conception des niveaux de commande et de supervision : **le projet CASPAIM**, présenté ultérieurement, vise à définir une méthodologie complète de la conception d'un système de production flexible. Elle intègre la génération automatique de la partie commande d'un Système à Evénements Discrets (SED) en industrie manufacturière et de son architecture, les fonctions de GPAO avec les problèmes d'ordonnancement, et la prise en compte des fonctions de supervision et de sûreté de fonctionnement.

## IV . 10 - Conception des systèmes de commande

Les systèmes de commande de par leur complexité et leurs spécifications ont des exigences élevées au niveau de leur modélisation et de leur représentation. Ceci constitue d'ailleurs, une des problématiques importantes du génie automatique qui vise à proposer des méthodes et des outils pour la conception de systèmes de commande temps réel.

Souvent, les démarches de développement s'apparentent à celle d'un "cycle de vie en V" [AFN 87] qui détaille les différentes phases d'élaboration de la commande et ses différentes phases de validation.

Les outils utilisés doivent pouvoir prendre en compte les diverses relations qui peuvent ainsi apparaître entre les processus.

Les réseaux de Petri se sont avérés être un outil intéressant de représentation. Les recherches basées sur les RdP sont multiples dans le domaine des protocoles, systèmes de production. Les réseaux de Petri permettent de caractériser aisément les systèmes à haut degré de parallélisme et à évolution asynchrone.

La mise en oeuvre de systèmes temps réel reste difficile. La coopération de l'ensemble des outils utilisés lors de la conception d'un système temps réel, est mal résolue. Il n'existe pas vraiment de méthodes de conception adaptées au temps réel et surtout à la haute technicité du matériel (environnement multi-processeurs). La communion de l'ensemble des contraintes telles que le temps, la gestion de données et la communication entre ressources dans un système réparti reste particulièrement délicate.

### IV . 10 . 1 - Rôle du système de coordination

A partir d'une image du procédé fournie par les capteurs, le système de pilotage va émettre des commandes via les actionneurs. L'essentiel est de pouvoir garantir une image exacte car si la vitesse d'évolution du procédé est trop rapide, le système de commande n'arrivera pas à assumer sa tâche. Le système de commande doit pouvoir appréhender toute variation significative de l'état du procédé. Un système de commande temps réel est en général un système non autonome qui réagit aux occurrences externes. Sa description précise les relations entre les occurrences des événements extérieurs et les occurrences des événements internes qui doivent en découler. Nous nous intéresserons plus particulièrement aux systèmes à événements discrets (cas du manufacturier).

### IV . 10 . 2 - Structure du système de commande

L'implantation sera forcément répartie. La structure du processus à commander peut tout d'abord suggérer une décomposition naturelle et un découpage de l'automatisme séquentiel en

sous-machines ou de l'algorithme de commande en sous-programmes. Cela nécessite d'inclure ou de distinguer, dans la représentation graphique, des sous-ensembles représentatifs des sous-machines ou des sous-programmes. De plus, il convient de régler les conflits d'utilisation de ressources communes si elles existent et d'étudier les couplages implicites ou explicites entre sous-machines et d'éviter les commandes contradictoires. Il s'agit aussi de synchroniser la présence des outils, des pièces et des palettes dans le processus de fabrication. Par exemple, cela peut être de prévoir le stockage d'une palette pour éviter un blocage dans le cheminement du reste des palettes. Un modèle initial du graphe de commande s'impose afin d'évaluer ses performances. Ce modèle doit appréhender le parallélisme avec un découpage possible en sous-ensembles en tenant compte de mécanismes d'échange entre processus. Le temps réel caractérise ces modèles, avec leurs aspects non déterministes et un temps de réponse non garanti.

### IV . 10 . 3 - Prises en compte des événements externes [MOA 85]

Les occurrences externes sont totalement asynchrones et ne sont pas maîtrisées par le système. C'est pourquoi il est difficile d'établir des règles de gestion de ces occurrences.

Un événement est dit **externe** s'il résulte de l'activation d'un opérateur externe au système. En général, une interface est chargée de traiter simultanément les occurrences externes, qui peuvent être disjointes.

### IV . 10 . 4 - Les modèles

Il est important de pouvoir modéliser le parallélisme. Plusieurs approches sont possibles :

- les représentations graphiques,
- les langages de programmation
  * langages classiques (procéduraux ou déclaratifs),
  * langages synchrones.

Ces approches différent en particulier, par le caractère de synchronisme ou non. L'approche graphique avec les Réseaux de Petri [BRA 83], le Grafcet [AFC 77] ou les States-Charts [HAR 87] est actuellement la plus utilisée. Un défaut, par exemple des Réseaux de Petri est que lors de la modélisation d'un problème complexe, avec un nombre important d'événements à gérer, la description et l'analyse sont lourdes. Par ailleurs, ils intègrent difficilement la notion de temps. Ils sont utilisés pour modéliser des systèmes à évolutions asynchrones.

### IV . 10 . 4 . 1 - Les langages asynchrones

L'approche asynchrone est courante lors de l'élaboration des systèmes de commande. Une programmation asynchrone permet de gérer des systèmes industriels, dont les applications sont d'ailleurs assez nombreuses.

Ce sont plutôt des langages transformationnels avec des tâches de traitement d'informations structurées et hiérarchisées, chaque tâche étant décomposée en activités exécutées séquentiellement sous la supervision d'un moniteur temps réel. Les actions sont générées lors d'occurrences d'événements émis par le process ou par d'autres processus internes de commande.

Le défaut des langages asynchrones est d'être peu adaptés à l'intégration du temps et d'être parfois pauvres en primitives de communication. Une solution pour remédier à ce problème, serait l'utilisation de techniques type "boîtes aux lettres" ou interruptions dont il est malheureusement difficile d'avoir une estimation exacte du temps de réponse.

Les langages de programmation parallèle de type ADA sont une solution mais sont non déterministes quand au temps de traitement. Ils incluent des mécanismes de communication entre processus et sont très peu réactifs.

### IV . 10 . 4 . 2 - Cas de Sceptre

Sceptre est un exécutif temps-réel qui est proposé comme norme des langages temps-réel [BNI 84].

Il permet de manipuler des objets permettant la communication et la synchronisation entre tâches, avec l'intégration du temps (introduction de time-out). Les états d'une tâche sont bien spécifiés ( En_cours, Prêt, En_attente, Hors_service...). Il offre de plus, des primitives de manipulation de tâches (Démarrer, Arrêter, Continuer...).

### IV . 10 . 4 . 3 - Approche synchrone

Une évolution est l'intégration explicite de nouvelles spécifications telles que le synchronisme ou les contraintes temporelles. Une approche possible s'oriente vers les langages synchrones [BER 86]. Les hypothèses de départ sont très simplificatrices par l'introduction de la synchronicité (prise en compte immédiate et traitement en temps nul des informations) et de la réactivité.

**Les systèmes réactifs** sont des systèmes temps réel particuliers : ils réagissent à des stimuli issus du procédé en lui renvoyant des signaux. Ils sont caractérisés par leur déterminisme car ils répondent toujours de la même manière aux mêmes signaux. Les entrées sont les signaux et les capteurs auxquels ils réagissent instantanément.

Leurs avantages sont de pouvoir générer des programmes structurés et fiables et constituent en premier abord, un outil correct de spécifications. Cependant, l'hypothèse de synchronicité constitue un obstacle à leurs utilisations et leur faisabilité reste à prouver (idée de machine infiniment rapide).

Le modèle d'exécution est rendu déterministe ce qui autorise :

- une compilation du programme en un automate à états finis, déterministe, équivalent sémantiquement à un langage parallèle,
- une estimation et une maîtrise des temps de réponse,
- tout signal est considéré comme une occurrence externe ce qui permet une prise en compte du temps multi-formes.

Les langages synchrones autorisent une gestion intéressante du temps par le fait de le manipuler sous différentes formes et comme un signal quelconque.

Le temps de traitement étant considéré comme nul, les signaux d'entrées et de sorties sont simultanés. Cela modifie considérablement les habitudes de programmation (il est possible et très facile d'implanter des communications entre processus, sans utiliser des techniques classiques telles que la mémoire commune grâce à la lecture instantanée des variables spécifiques à chaque processus : principe de diffusion simultanée). Ceci rend ces programmes très réactifs.

#### IV . 10 . 4 . 4 - Exemple : le langage Esterel

Ce langage synchrone [BER 86] répond à toutes les caractéristiques citées auparavant.

Les primitives du langage sont assez complètes. Certaines sont classiques (affectation, appel de procédure), d'autres sont plus riches telles que les primitives temporelles (chien de garde ou test de présence). Une compilation du programme en un automate permet lors de l'exécution un gain appréciable. Le source généré peut, par exemple, être un module C, langage fortement utilisé dans des systèmes temps réel.

#### IV . 10 . 4 . 5 - Approche mixte

Les approches asynchrones et synchrones, prises individuellement, n'étant pas totalement satisfaisantes, une combinaison des deux peut être intéressante. L'idée est de distinguer la partie transformationnelle de la partie réactive, en les implémentant chacune dans un langage adapté, par exemple, l'intégration d'Esterel et d'un dérivé de Sceptre [AND 89]. Ceci a l'avantage de séparer la partie transformationnelle ne nécessitant pas un temps de réponse immédiat de celle demandant un temps de réponse plus rapide.

## IV . 11 - Présentation du Projet Caspaim

Le projet Caspaim a débuté en 1985. Il vise à proposer une méthodologie générale de conception et de réalisation de système de commande de processus discrets dans le cadre d'une production en industrie manufacturière. Cette approche intègre le caractère flexible de la commande et le dimensionnement du modèle, avec la génération d'un système réparti et hiérarchisé. C'est une démarche globale de la spécification à l'implantation.

Le but est de générer, directement à partir d'un cahier des charges proposé par l'utilisateur, le graphe de commande d'un système de production. Un code correspondant au graphe de commande directement implantable sur machine est alors proposé après diverses phases de traitement.

Notre travail s'intègre dans ce projet au niveau de la phase d'implantation, en particulier pour la conception d'un système de pilotage, organisé sur la gestion des modes de marche des différentes machines du process.

### IV . 11 . 1 - La démarche initiale

Nous présentons la chaîne logicielle développée qui permet de générer le graphe de commande à partir des spécifications initiales.

Nous pouvons distinguer 4 grandes phases :

- une phase d'élaboration du cahier des charges et des spécifications,
- une phase de conception et de génération avec la modélisation de la partie commande,
- une phase d'évaluation et de simulation,
- une phase finale d'implantation.

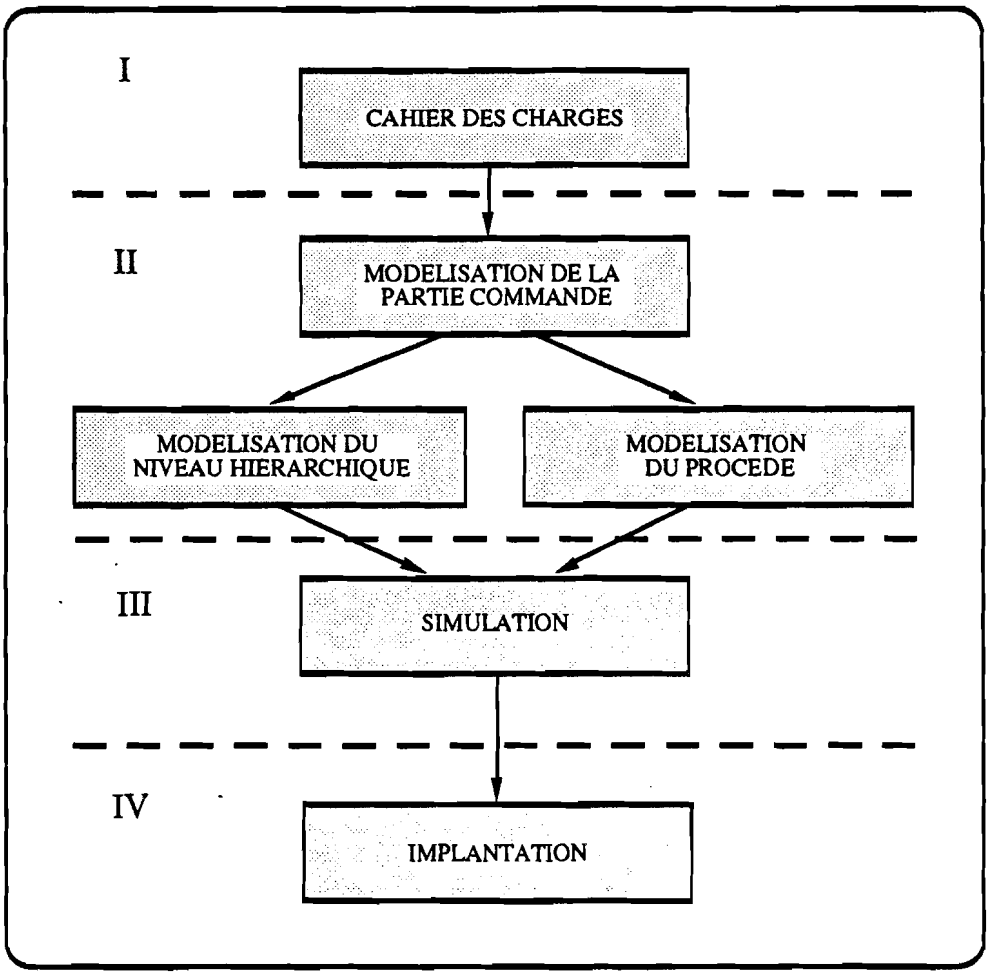



fig 12 : méthodologie Caspaim

Ceci est le squelette général de la chaîne de conception.

Nous allons détailler plus finement les différentes phases. D'une manière générale, à chaque étape, l'utilisateur est assisté dans sa démarche car elle met en oeuvre des modèles complexes (Réseaux de Petri, Grafcet, règles de production).

### IV . 11 . 1 . 1- La spécification [GAS 89]

La première phase de la chaîne logicielle est la phase de spécifications. Un logiciel permet en mode interactif, l'édition et le contrôle des spécifications nécessaires à l'élaboration et la structuration du cahier des charges et des gammes opératoires. L'approche considérée est la prise en considération des gammes opératoires avec la description séquentielle du chemin suivi et des opérations subies par les pièces produites. Une première étude de cohérence et de complétude est réalisée sur la structure des spécifications, exprimées sous forme de régles de production. Une traduction est réalisée, pour être utilisée ensuite dans la phase de pré_étude des gammes.

L'avantage de cette phase est la garantie de travailler sur des données cohérentes tout au long du projet. L'originalité de cette approche est qu'à partir de méthodologies proposées le plus souvent en génie logiciel, de fournir un outil équivalent pour une application plus industrielle.

### IV . 11 . 1 . 2 - La modélisation [KAP 88]

Cette phase se déroule en deux temps :

- tout d'abord une pré_étude des gammes,
- ensuite la génération du prégraphe.

A partir de la base de règles fournie par la phase de spécifications, un logiciel va réaliser une seconde analyse de cohérence et de complétude de la base de connaissances. Il recherche précisément les règles ou les faits manquants à la description car il est primordial pour la suite du projet de s'assurer de la validité et de la conformité des informations. Cela permet une étude qualitative assez fine des spécifications en complément avec l'analyse plutôt quantitative réalisée auparavant, qui assure de plus du bon enchaînement des règles de production et du lien entre un ensemble de faits initiaux et de faits terminaux.

Le logiciel d'aide est un moteur d'inférence fonctionnant en chaînage mixte. Il est interactif avec l'utilisateur, avec son interrogation jusqu'à complète satisfaction dans l'analyse.

#### Génération du prégraphe

Les règles de production sont regroupées par gammes opératoires. La description fonctionnelle est assez lourde, c'est pourquoi une étape d'agrégation s'avère nécessaire pour la simplifier en un prégraphe.

Un prégraphe est un RdP coloré représentant le rôle fonctionnel d'une cellule de production. Il recense l'ensemble des trajets possibles pour les pièces et objets circulant dans le système de production. Il permet de visualiser de plus, les différents transferts et les postes de la cellule.

Ce prégraphe sert de base à l'étape suivante de structuration.

### IV . 11 . 1 . 3 - Génération de la partie commande [BOU 88]

Cette phase a pour but de générer un modèle de la cellule, construit à partir des RdPSAC.

**Structuration du prégraphe**

Cette étape, originale par le fait qu'elle développe un modèle agrégé en modèle développé, fait ressortir les différents niveaux de parallélisme et les différents processus concurrents.

Les différentes entités existantes dans un système de production (convoyeur, machine) sont modélisées de façon à systématiser l'étape de structuration avec la décomposition des séquences opératoires en opérations élémentaires.

**Répartition des processus de commande**

Les contraintes relevant de l'architecture matérielle de l'atelier ne sont pas encore modélisées. Ces contraintes sont la synchronisation des opérations, le partage de robots entre plusieurs tâches de transfert, et les assemblages. Des protocoles d'accés sont ainsi mis en place.

**Modélisation du niveau hiérarchique**

C'est une étape de détection des conflits et interfaçage avec la partie commande.

**Détection des conflits** : c'est la recherche systématique et automatique de tous les conflits structurels spécifiques à la structure du graphe de commande.

Ces conflits sont les accès simultanés à une ressource et les indéterminismes directionnels.

L'intérêt de cette démarche est qu'elle permet de les résoudre par l'écriture des règles du niveau hiérarchique.

**Interfaçage avec la partie commande**

Cette interface est réalisée avec l'association des arcs adaptatifs et de places d'interfaces, qui permettent de bloquer ou non l'évolution de certains sous-ensembles du graphe. Ces places sont modifiées par des règles de production.

**Etape de modélisation du procédé**

Cette phase de modélisation est composée du choix des entrées, du dimensionnement, et de la temporisation.

Cette étape permet de paramètrer le graphe de commande en vue particulièrement de l'étape de simulation.

Le modèle obtenu, élaboré de manière assistée, reproduit de manière très fine la cellule de production.

### IV . 11 . 1 . 4 - La simulation [CAS 87]

L'étape de la simulation est l'étape obligée pour la validation du graphe de commande. Cette simulation permet tout d'abord d'évaluer le comportement d'un système : statistiques de fonctionnement, étude de la propagation des défauts, de leurs détections.

Un simulateur a été développé pour l'étude des RdPSAC, modèle utilisé pour l'étape antérieure de structuration.

La partie décisionnelle est décrite par un ensemble de règles de production.

Les paramètres concernant le graphe de commande, permettent une description plus fine du procédé, telles que la temporisation des places, définition de Time_out et de dimensionnement des files d'attentes.

**Support de la simulation**

Trois modèles sont utilisés :

- les RdPSAC pour la modélisation de la partie commande,
- les règles de production pour modéliser le niveau hiérarchique,
- le procédé est modélisé par des catégories génériques multi-niveaux de type langages objets.

Cette étape est intéressante dans la démarche CASPAIM car elle permet de remettre en cause toute la génération antérieure du graphe. Dans le cas d'une validation partiellement défaillante, un retour arrière s'impose. De plus, cette étape permet de définir le dimensionnement optimum du graphe qui permettra d'éviter les blocages. Tout d'abord la simulation permet de vérifier la bonne écriture du graphe pour chaque type de pièces. En outre, cela vérifie que les règles du niveau hiérarchique sont correctement définies pour résoudre les problèmes d'indéterminismes ou de conflits.

### IV . 11 . 1 . 5 - L'implantation de la commande [CRA 89]

L'implantation concerne tout d'abord l'étude **de la partie commande**, validée antérieurement. Les RdPSAC constituent le modèle de base, cependant de par des nécessités

technologiques, une phase de transcription en Grafcet du modèle s'impose pour élaborer le code. Il est important de systématiser cette démarche car la taille du modèle impose une rigueur de transcription de façon à respecter les spécifications initiales.

Dans un second temps, il faut prendre en compte l'architecture matérielle de la cellule, en tenant compte des possibilités de répartition du graphe de commande. Il faut essayer de respecter le parallélisme mis en avant jusqu'à présent, en tenant compte des différents contrôleurs existants. Deux types de regroupements sont retenus : fonctionnels et économiques. Il s'agit, soit de regrouper les machines fortement connexes entre elles afin de limiter les communications, soit de minimiser le coût d'installation en optimisant le fonctionnement des machines.

L'implantation concerne aussi celle **du niveau hiérarchique**. Il a pour but d'assurer le règlement des indéterminismes et conflits du graphe de commande. La solution retenue est son implantation sur un calculateur externe susceptible d'avoir une vision plus générale du fonctionnement et capable d'une programmation dans un langage évolué.

L'idée de ce niveau hiérarchique est, à partir d'informations reçues du process, de gérer les décisions du graphe de commande. Sa définition dépend de critères et d'impératifs choisis par l'utilisateur.

Le logiciel est en fait un système expert d'ordre O+ fonctionnant en chaînage avant. Ces règles sont de type "règle de production" et sont déduites des règles utilisées lors de l'étape de simulation.

La base de faits modélise l'état de certaines données du procédé : l'état d'une pièce, ou de données internes aux automates.

La base de règles modélise des prises de décision choisies par l'utilisateur dans le cadre du règlement de conflits ou d'indéterminismes.

La difficulté de cette phase réside dans la réalisation effective avec les moyens de communications qui restent spécifiques au matériel. L'idée de base est le fonctionnement asynchrone. Le process est considéré comme figé tout au long de l'inférence. A partir d'une image du procédé, le moteur évalue et résoud les indéterminismes. De plus, à l'aide de méta-règles, l'utilisateur oriente les prises de décisions.

### IV . 11 . 2 - Les modèles retenus

Dans le cadre de l'implantation de la commande, nous nous sommes intéressés aux outils proposés par le laboratoire. La commande de notre exemple a été élaborée en utilisant la méthodologie CASPAIM. C'est pourquoi il nous a semblé nécessaire d'en faire une présentation générale.

Nous en montrerons les avantages et les défauts et son avancement actuel.

Nous allons présenter les modèles utilisés pour modéliser la commande et son implantation.

Lors de la création du graphe de commande nous avons utilisé deux modèles de même type : les réseaux de Petri et le Grafcet. Ce sont deux outils qui permettent de modéliser le comportement d'un système de commande et qui permettent d'en faire une représentation graphique. Les réseaux de Petri sont utilisés à des fins de spécifications de modélisations et d'aide à la conception des systèmes discrets de commande en temps réel. Plusieurs extensions en sont déduites pour enrichir le modèle de base. Le Grafcet est un modèle normalisé assez proche des RdP et est très utilisé dans le monde industriel. Nous allons présenter ces deux modèles.

### IV . 11 . 3 - Les réseaux de Petri [BRA 83] [DAV 89]

L'origine des Réseaux de Petri remonte aux débuts des années 60.

Ils ont l'avantage de visualiser et de modéliser des automatismes logiques dont le comportement peut inclure des aspects communication, concurrence, parallélisme. De plus, leur modélisation est assez précise et détaillée pour pouvoir envisager une étude qualitative et quantitative du comportement du système modélisé par réseaux de Petri.

Nous n'insisterons pas sur la description des Réseaux de Petri, mais plutôt sur leurs utilisations. Nous avons limité le modèle à l'utilisation des réseaux de Petri structurés, adaptatifs et colorés.

#### IV . 11 . 3 . 1 - Présentation des Réseaux de Petri

Les réseaux de Petri sont utilisés pour modéliser le comportement dynamique de systèmes discrets. Ils concernent la conception de systèmes de commande et leurs validations.

Leur formalisme de base permet la génération d'une description dépourvue de toute ambiguïté. Le modèle peut être ainsi traduit facilement en une réalisation matérielle ou logicielle. Cependant les modèles que nous avons utilisés sont plus complexes et entraînent des ambiguïtés d'analyse.

Un réseau comporte deux types d'objets : des places et noeuds.

Les places représentent les états possibles du système modélisé.

Les transitions représentent les conditions d'évolution du graphe. Cette transition s'effectuera si la transition est validée, c'est à dire si toutes les places en amont sont marquées. Cela signifie que le système change d'état ainsi que le marquage associé.

Une réalisation à base d'automatismes est possible à condition d'imposer certaines restrictions [NUS 88] :

- réseau sauf,
- pas de boucles (une place ne doit pas servir d'entrée et de sortie à une même transition),
- l'évolution entre états stables est infiniment rapide,
- il ne peut y avoir de variations simultanées des entrées,
- les entrées ne varient pas pendant un régime transitoire.

La complexité des processus à modéliser ainsi que la nature des relations mises en jeu a conduit à utiliser un certain nombre de primitives de communication et à structurer le modèle.

### IV . 11 . 3 . 2 - Nouvelle classe de Réseaux de Petri

**Les réseaux de Petri structurés** [COR 79] [COR 80]

L'étape normale et intuitive est, lors de la modélisation d'un processus complexe, de le décomposer fonctionnellement en sous-problèmes élémentaires, plus simples à représenter, structuration identique à celles rencontrées dans l'utilisation des langages informatiques.

Cela est permis par la nature des RdP Structurés. Chaque processus indépendant est en relation avec d'autres processus, par l'intermédiaire de primitives de liaisons.

De plus, cela simplifie la description et garantit son fonctionnement par l'étude de propriétés qui découlent de la structuration (caractère sauf et invariant).

### IV . 11 . 3 . 3 - Les graphes de processus

Un processus est un séquencement de tâches élémentaires. Trois structures élémentaires sont dégagées, l'action, la répétitive et l'alternative. En utilisant le formalisme Backus-Naur, nous avons la syntaxe des RdPS.

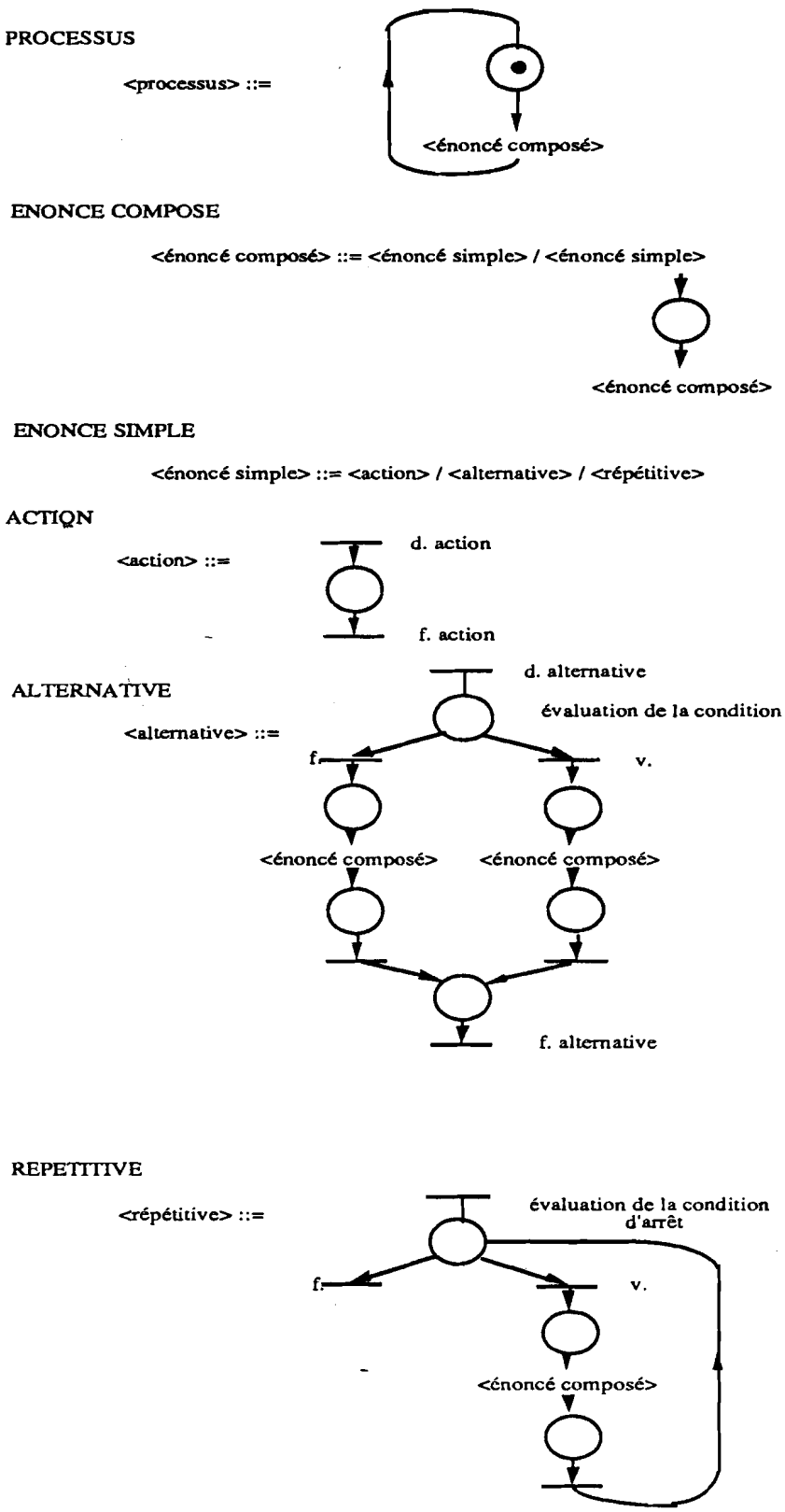



fig 13 : syntaxe des RdP

Des liaisons inter-processus interconnectent les graphes de processus : ce sont les modélisations de l'exclusion mutuelle, d'une synchronisation et du producteur/consommateur.

**Les réseaux de Petri adaptatifs et colorés**

**Les réseaux de Petri adaptatifs** rendent le modèle déterministe et permet l'interfaçage avec d'autres modèles. Ils introduisent d'ailleurs, la notion de flexibilité.

**Les réseaux de Petri colorés** [COR 85] permettent de réduire la taille du modèle en modélisant sur une même place, différentes classes de jetons. Des informations de changement d'état des jetons au niveau des transitions.

## IV . 11 . 4 - LE GRAFCET

### IV . 11 . 4 . 1 - Introduction

Le Grafcet est le fruit de recherches menées par l'ADEPA. L'idée du Grafcet au départ est de rendre l'approche de ce type de langage plus directe et plus simple, mieux adaptée aux automatismes industriels. Cet outil peut être considéré comme une version simplifiée des réseaux de Petri, permettant une utilisation plus étendue des représentations graphiques en particulier au niveau des industriels.

Il est destiné à la représentation du cahier des charges d'un automatisme logique.

Les éléments de base du Grafcet sont les étapes et les transitions.

Les étapes représentent les états du système modélisé et sont associées à des actions.

Les transitions sont des arcs orientés qui permettent de respecter les conditions de franchissement qui feront évoluer le modèle.

### IV . 11 . 4 . 2 - Différences et analogies avec les RdP

**Les analogies**

Le Grafcet peut être considéré comme un Réseau de Petri sauf avec adjonction d'une condition de réceptivité aux transitions et avec une modification des règles de tir des transitions. Par ce point de vue, le Grafcet se rapproche des Réseaux de Petri Interprétés, dans lesquels les évolutions des marquages sont synchronisées sur des événements extérieurs.

Les points communs sont l'utilisation de deux types de noeuds (les étapes et les transitions) et leurs évolutions sont donc synchronisées sur les occurrences extérieures.

**Les différences**

Les différences tiennent essentiellement aux différences d'interprétation dans le tir des transitions. L'évolution du marquage est différente de celle des réseaux de Petri. Elle diffère

par les règles de franchissement. En effet, les règles de franchissement des transitions pour le Grafcet sont :

- toute transition franchissable est immédiatement franchie,
- plusieurs transitions simultanément franchissables sont simultanément franchies,
- lorsqu'une étape doit être simultanément activée et désactivée, elle reste active.

Les règles 1 et 3 sont identiques pour les réseaux de Petri. Cependant, la règle 2 est nuancée par le fait que les transitions ne peuvent être simultanément franchies : il y a l'introduction d'un conflit.

Dans un Grafcet, toutes les transitions sont simultanément franchies, la marque étant divisible à volonté alors que dans un réseau de Petri, il est possible que toutes les conditions franchissables ne soient pas franchies. Dans un RdP, les marques s'accumulent (excepté réseau sauf) et servent à déterminer l'état de la place. Dans un Grafcet, une étape n'a que deux états, actif ou inactif. Dans un RdP, les réceptivités associées aux transitions ne peuvent porter sur des états internes des places, alors que cette notion est possible dans le Grafcet.

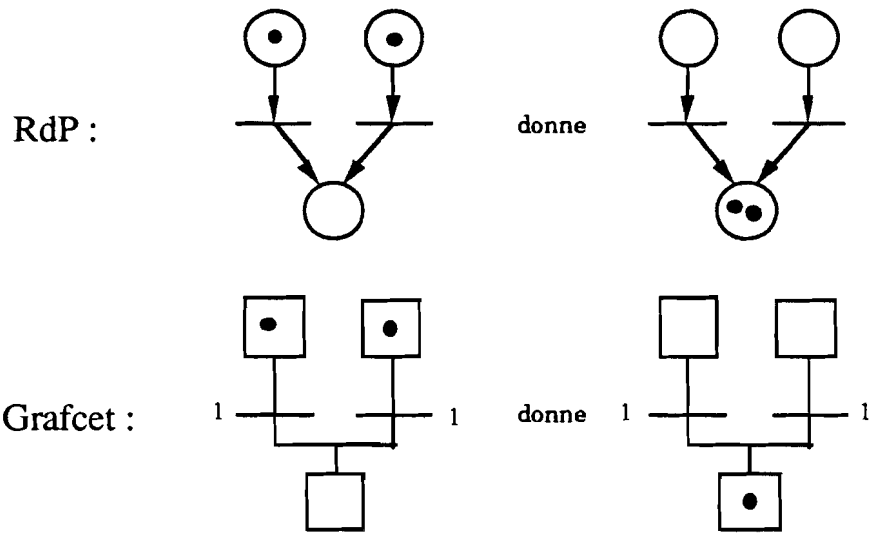



fig 14 : différence de comportement

### IV . 11 . 4 . 3 - Exemple d'utilisation du Grafcet

Nous avons pris l'exemple du transfert d'une pièce sur un convoyeur à chaîne, géré à partir d'un automate programmable.

Chaque poste sur le convoyeur est constitué d'un emplacement destiné aux opérations de chargement/déchargement et de postes d'attente, gérés en FIFO.

Lorsque une palette arrive au niveau de la zone opératoire, sa présence est détectée par le capteur C1. La butée B1 est alors déclenchée pour bloquer la palette. Le capteur C2 vérifie sa présence effective.

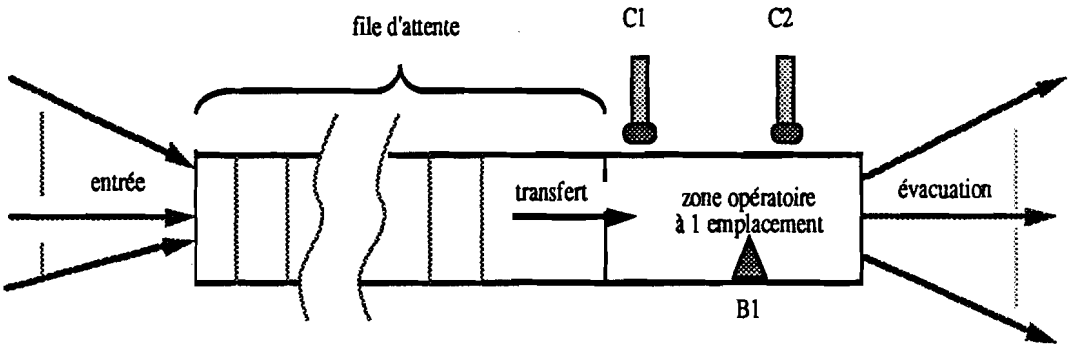



fig 15 : positionnement d'une palette

Voici le Grafcet fonctionnel représentant le cahier des charges.

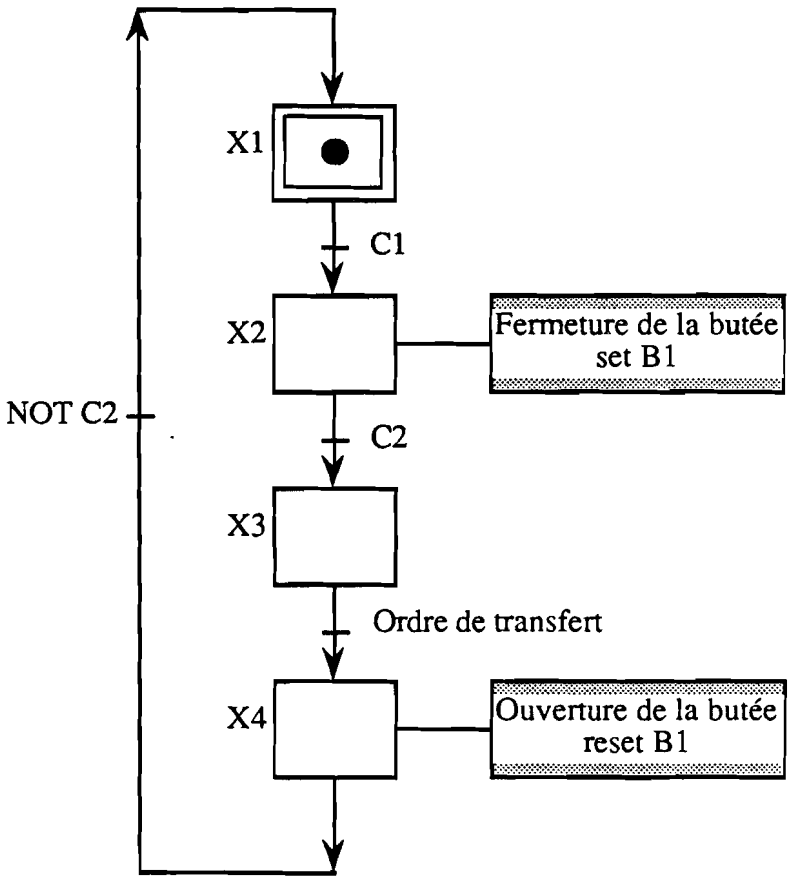



fig 16 : Grafcet équivalent

L'étape d'initialisation X1 indique si le poste est libre ou occupé. Si l'étape est marquée, cela signifie que le processus est disponible à l'arrivée d'une palette.

Dans ce cas, elle sera détectée par le capteur C1. L'étape X1 est désactivée avec l'activation de l'étape X2. L'action associée est la fermeture de la butée avec le blocage de la palette. Le capteur C2 permet de vérifier la validité des opérations avec l'activation de l'étape X3.

Un ordre de transfert permet l'ouverture de la butée B1. L'étape X4 est activée et l'action associée est l'ouverture de la butée.

Dans ce cas, le capteur détectera la disparition de la palette avec la désactivation de l'étape X4 et la réinitialisation du processus.

### IV . 11. 5 - Transposition des RdPSAC [CRA 89]

Le Grafcet est un outil de programmation largement utilisé par les constructeurs d'automatismes sous l'impulsion de l'ADEPA. Les réseaux de Petri sont plus souvent utilisés dans les laboratoires.

Dans la perspective d'une implantation en Grafcet du graphe de commande, la transposition du modèle RdPSAC en Grafcet s'est donc avérée nécessaire pour pouvoir passer à l'étape d'implantation.

#### IV . 11 . 5 . 1 - Transposition du modèle de base

A priori, la transposition ne paraît pas évidente car le modèle RDP est beaucoup plus riche que celui du Grafcet. La transposition du modèle est simplifiée par la nature structurée du Réseau de Petri et son caractère sauf autorise une transposition assez directe.

La transposition Réseau de Petri/Grafcet a été possible car les problèmes dus aux nuances d'interprétation des règles de franchissement ont été escamotées. Le réseau de Petri est sauf car ne circule qu'une pièce au niveau des processus. Le passage a été possible par le coté déterministe et autonome de la traduction par l'introduction d'un niveau hiérarchique.

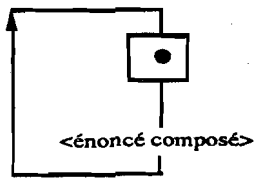
La transposition au départ ne nécessite que la modification graphique avec un respect des fonctionnements des langages, en remplaçant les places par des étapes. La structure initiale est totalement conservée et le caractère structuré des processus est présent.

#### IV . 11 . 5 . 2 - Syntaxe du Grafcet

De manière identique à la représentation de la syntaxe des RDPS, nous pouvons représenter la syntaxe Grafcet :

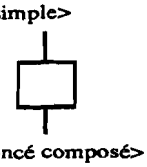
PROCESSUS

<processus> ::=




ENONCE COMPOSE

<énoncé composé> ::= <énoncé simple> / <énoncé simple>

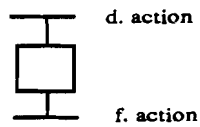



ENONCE SIMPLE

<énoncé simple> ::= <action> / <alternative> / <répétitive>

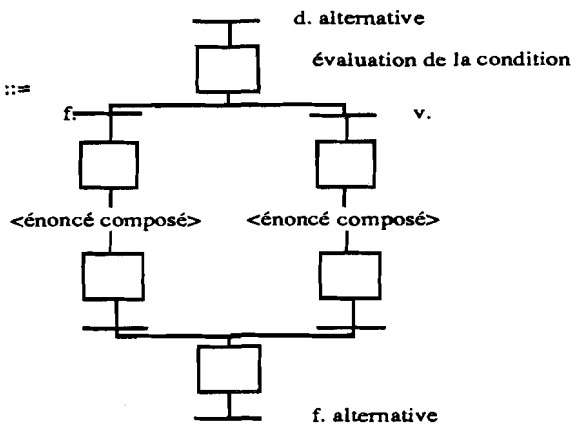
ACTION

<action> ::=




ALTERNATIVE

<alternative> ::=




REPETITIVE

<répétitive> ::=

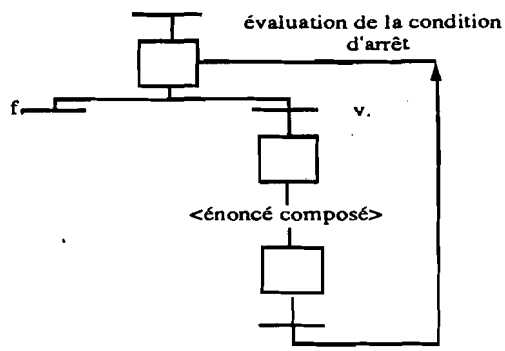



fig 17 : modèle Grafcet

Cependant des problèmes se posent :

- transposition des primitives de communication entre processus,

- transposition de la couleur,
- transposition du caractère adaptatif.

En effet, ces différentes notions ne sont pas prises en compte par le langage Grafcet. La transposition des primitives de communication a été systématisée [CRA 89].

Celles de la couleur et des arcs adaptatifs ont été réalisées en vue d'une implantation donc avec l'opportunité de pouvoir utiliser des variables internes dans la transcription.

### IV . 11 . 5 . 3 - Cas des réseaux adaptatifs

Les réseaux de Petri adaptatifs bloquent l'évolution d'un graphe. En effet, les règles d'évolution obligent que le marquage courant de la place étiquetée soit supérieur ou égal au marquage de la place et quotant cet arc.

Dans notre cas, nous utilisons des réseaux saufs, c'est à dire dont le marquage est égal ou inférieur à 1. Ceci revient, non pas à avoir un étiquetage d'un arc mais à avoir une variable booléenne sur la transition. Si la variable est à faux alors il y a blocage de la transition.

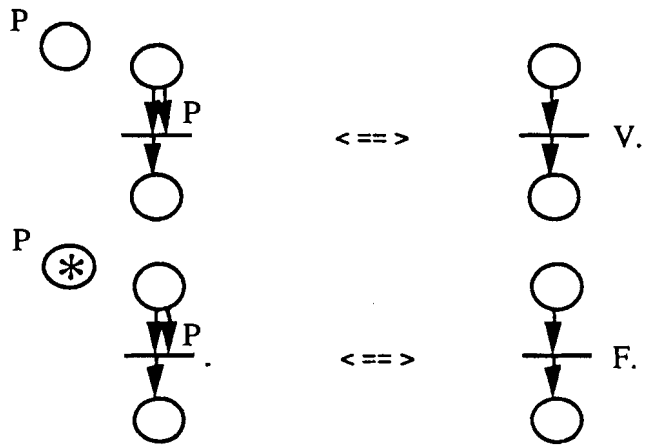



fig 18 : équivalence pour les processus saufs

Pour la transposition, nous utilisons pleinement les facilités offertes par le langage PL7/3. En effet, il est possible de tester l'activité des étapes d'où d'avoir comme test sur les arcs, la valeur de la réceptivité des étapes.

Par exemple, une modélisation possible est l'utilisation de deux étapes I et J, dont nous testons les activités Xi et Xj au niveau des transitions.

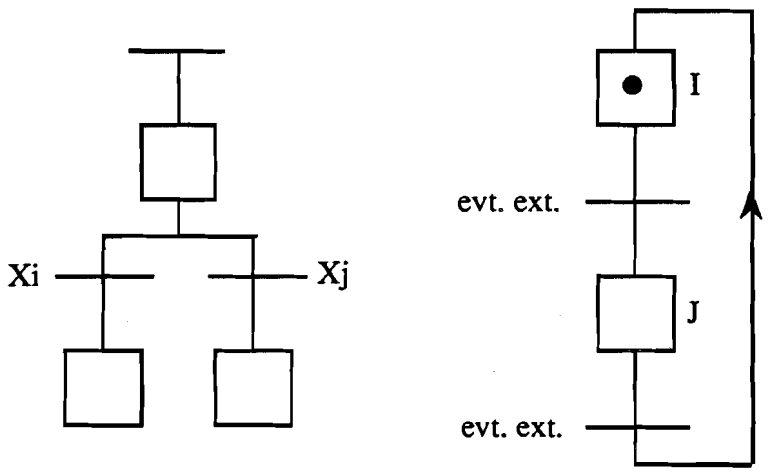



fig 19 : gestion explicite des priorités

La modélisation est possible car nous utilisons des RdPA saufs avec la seule différence que les conditions d'adaptivité sont reportées au niveau des arcs.

### IV . 11 . 5 . 4 - Cas de la couleur

Le Grafcet ne supporte pas la coloration qui permet d'enrichir les informations véhiculées par la marque. L'information véhiculée initialement dans une représentation Grafcet est l'état en cours du processus, c'est à dire la nature des actions engendrées. Les informations susceptibles d'être utilisées sont le type et la nature de la pièce modélisée par un jeton. A partir de ces informations, nous pouvons connaître le degré d'avancement des pièces dans leur gamme d'usinage.

Pour effectuer la modélisation, des mots sont utilisés. Les graphes étant saufs par construction, il suffit d'une seule variable par processus pour réaliser le codage en Grafcet.

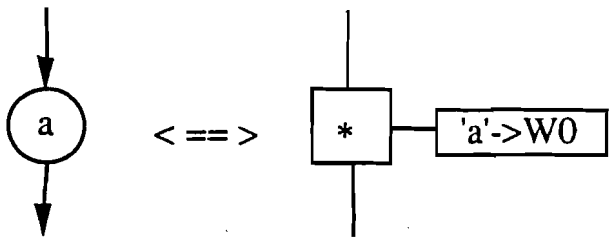



fig 20 : W0 contient le type de la marque qui est a

De plus il faut pouvoir modéliser les tests de couleurs au niveau des arcs, afin de connaître les chemins admissibles. Ces tests sont ramenés au niveau des transitions des Grafcet.

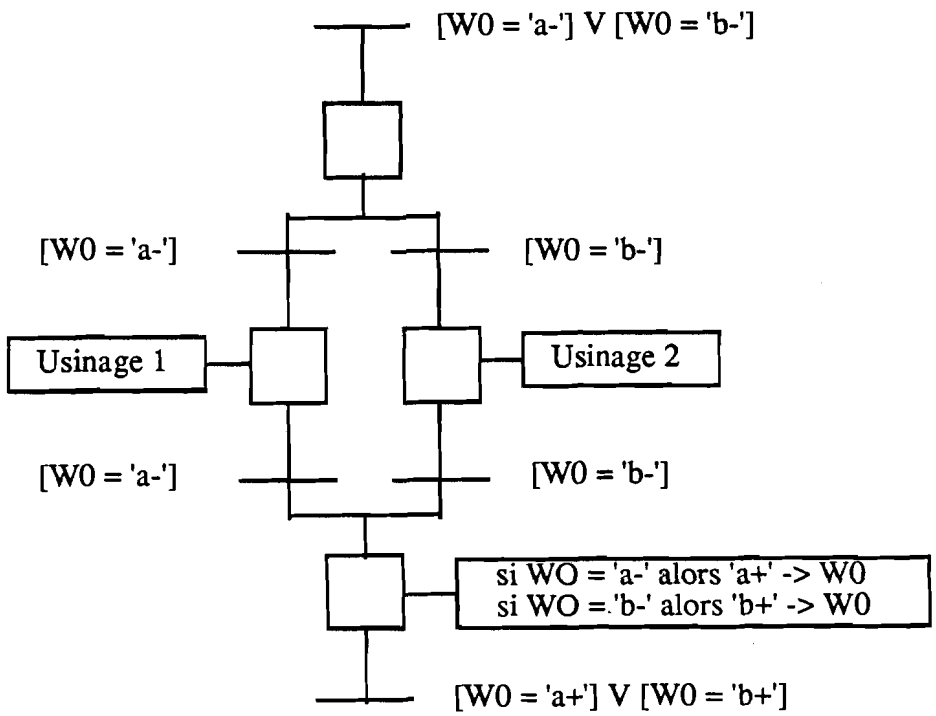



fig 21 : exemple de coloration en Grafcet

On vérifie par des tests d'égalité l'appartenance de la marque à un domaine de couleur ce qui permet de sélectionner les opérations d'usinage en fonction de la pièce.

L'information concernant les pièces est mise à jour dans les étapes où il se passe effectivement des transformations de pièces.

Par exemple, dans le cas d'un usinage, la pièce marquée a- devient a+ après usinage.

### IV . 11 . 5 . 5 - Conclusion sur la transposition

Le caractère structuré et sauf du modèle Réseau de Pétri adaptatif et structuré autorise une transposition automatique du modèle engendré par la conception.

Cette transposition est rigoureuse et systématique. Le parallélisme et les communications sont rendus complètement transparents.

Cependant, la modélisation pourrait être améliorée en utilisant les RdP à prédicats qui enrichisseraient le graphe de commande avec l'association d'informations aux marques. D'ailleurs, cette extension a été retenue dans le cadre de la nouvelle démarche CASPAIM.

Nous avons donc dans le cadre du projet pour l'implantation d'un système de pilotage, utilisé cette traduction pour générer les programmes de la partie commande, implantables sur machines. Le cycle de fonctionnement de l'automate autorise une telle transposition.

### IV . 11 . 6 - La nouvelle démarche Caspaim [CRU 91]

Certains exemples ont mis en évidence des lacunes et des insuffisances dans la méthodologie Caspaim. Il était par exemple difficile de modéliser d'une façon élégante les objets de type "lots". De plus, le caractère agrégé du Prégraphe, rend la sémantique associée imprécise (ne différencie pas, par exemple, un assemblage d'une palettisation). Afin de remédier à ces défauts, la démarche générale a été profondément repensée, toujours en se limitant au domaine d'application relevant de l'industrie manufacturière.

Cette nouvelle méthodologie est orientée "produits". Une première modélisation des gammes logiques relatives à chaque produit est réalisée, elle spécifie le séquencement des états des pièces. Une deuxième modélisation intègre les moyens de production et de manipulation afin de constituer les gammes opératoires. Le support de modélisation est le RdP.

L'avantage de cette démarche est qu'elle spécifie séparément les opérations et les produits.

Le graphe de commande est obtenu alors, en intégrant la spécification des protocoles d'accés aux lieux physiques (gestion d'allocation des ressources et des lieux de stockage).

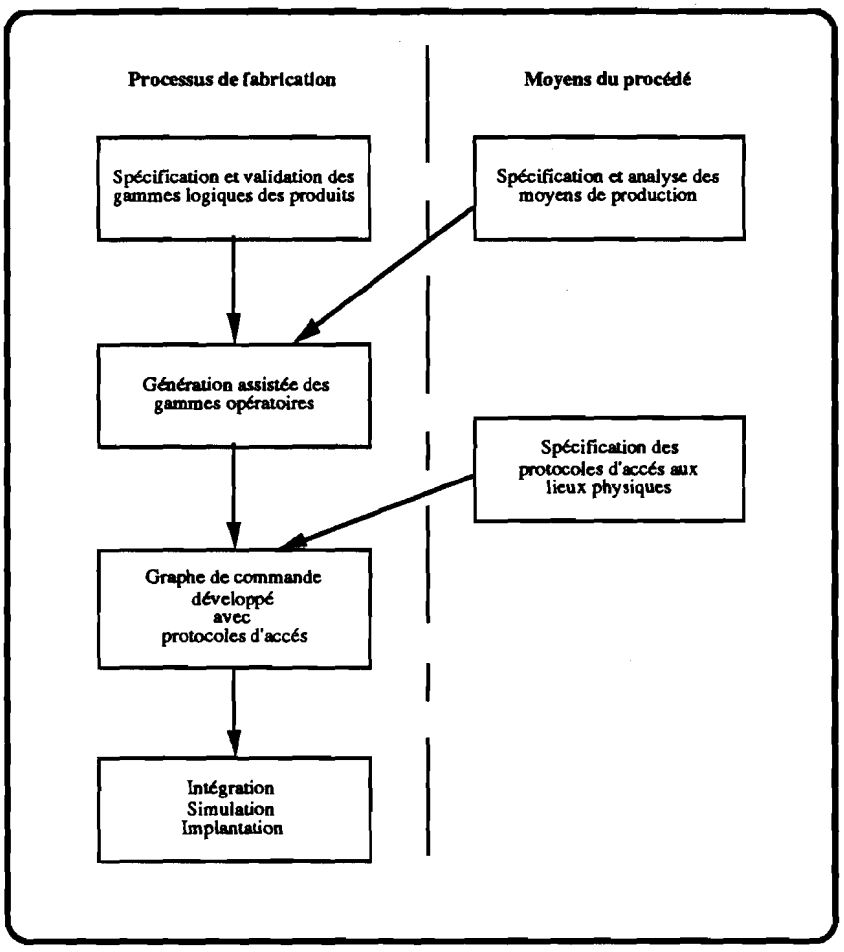



fig 22 : nouvelle démarche Caspaim

### IV . 11 . 7 - Situations de nos travaux

Le travail présenté dans ce mémoire s'oriente plus particulièrement vers la conception d'un système de pilotage du processus de fabrication. Ce système se situe à un niveau hiérarchiquement supérieur à celui de la partie commande présentée auparavant dont nous avons présenté la démarche d'obtention et les modèles retenus.

Dans le cadre du projet CASPAIM, un premier niveau de système de pilotage a été étudié [CRA 89] et conçu avec le souci d'optimiser le fonctionnement de la partie commande. Cependant, il demeure "limité" du fait de son fonctionnement très spécialisé, dédié à la résolution des indéterminismes et des conflits de la commande. C'est pourquoi nous nous sommes attachés à développer et à structurer le niveau hiérarchique de façon à résoudre des problèmes plus généraux tels que la gestion des modes de marche et leurs transitoires.

La démarche CASPAIM est fortement orientée "produits" et nous avons essayé d'intégrer le côté "machines" du procédé. Pour cela, nous nous sommes intéressés aux différents modes de marches des machines, de façon à les intégrer dans le système de commande.

## CONCLUSION

L'informatisation et l'automatisation des systèmes industriels sont des facteurs indéniables dans l'amélioration de productivité.

Cependant, ceci ne peut être fait que de manière globale et générale, sinon voué à être inutile. Il est impératif d'aboutir à des systèmes évolutifs adaptables par opposition aux systèmes homogènes rigides et qui deviennent obsolètes.

Il existe peu de méthodologie de conception globale et directe, car la spécificité des usines entraîne à jouer sur de multiples paramètres. Le savoir-faire demeure important avec de multiples compromis entre les coûts financiers et les impératifs techniques.

Les résultats obtenus avec l'introduction d'un contexte CIM sont significatifs. IBM a constaté un gain de 20 % de la productivité sur les effectifs, une réduction de 50% des cycles de fabrications et une diminution de 30 % des délais.

Une démarche méthodologique cohérente est un facteur-clé de l'automatisation. C'est pourquoi le laboratoire s'attache à développer des méthodes de conception de systèmes de commande. Nous avons utilisé ses résultats lors de l'implantation de notre exemple.

Dans ce chapitre, nous avons voulu présenter les systèmes de production, en nous limitant au contexte CIM, ceci afin de mettre en valeurs différents aspects, qui sont évoqués tout au long du rapport et qui permet de préciser la situation de nos travaux :

- la décomposition hiérarchisée et structurée à laquelle aboutissent les systèmes classique de fabrication constitue le fil directeur de notre démarche de modélisation présentée au chapitre II,

- la réalisation d'un exemple de système de pilotage, présenté au chapitre III, intègre les idées du CIM, c'est à dire le contrôle automatique, complétement informatisé d'une unité de fabrication.

# CHAPITRE II

## Modélisation

## INTRODUCTION

Les processus industriels deviennent de plus en plus performants et cela entraîne un niveau supérieur de complexité pour assurer leurs commandes et leurs suivis. Cette augmentation des performances est nécessaire de par les critères imposés que sont les notions de compétitivité et de productivité. Une façon de procéder est l'intégration des différents modes de marche des machines composant le système de production. Plusieurs approches sont envisageables : GPAO, Intelligence Artificielle [RIC 87], heuristiques mathématiques, langages synchrones [BER 86]...

Une démarche possible est l'intégration d'un système expert au sein du processus de commande. Celui-ci assume plusieurs fonctions [OBS 89] :

- gestion des informations,
- filtrage, vérification et émission des ordres vers le processus,
- pilotage et coordination des machines.

Ces fonctions s'implémentent à différents niveaux suivant leurs portées. Nous nous sommes attachés à résoudre deux fonctions importantes : le pilotage des machines et le filtrage des ordres vers le procédé, fonctions qui sont entièrement complémentaires.

Le problème de l'élaboration d'un système expert est d'être assez performant pour répondre aux exigences d'un expert dans la conduite mais assez simple d'utilisation pour pouvoir être utilisable par un usager non spécialiste. Les autres problèmes majeurs rencontrés lors de l'implantation d'un système expert sont la prise en compte globale des données et son temps de réponse. La nature diverse et le nombre de ces données obligent à en faire une synthèse rigoureuse pour faciliter leur compréhension. Nous présenterons la modélisation adoptée pour formaliser les informations utilisées.

Deux outils sont utilisés : **les graphes d'état** et **les tables de contraintes** [BOI 91].

Nous décrirons tout au long de la démarche un exemple complet afin d'illustrer la méthode.

# I - DEFINITION DE LA GESTION DES MODES DE MARCHE

La gestion des modes de marche consiste en la commande effective des machines en intégrant la modification de leurs états. Ces états peuvent être modifiés par une émission d'ordre ou par l'occurrence d'une panne indépendante des ordres du système. Cette gestion est complexe car il faut, non seulement considérer la machine comme une entité indépendante, mais aussi comme la partie intégrante d'un ensemble de machines. Ces différentes composantes évoluent indépendamment. Cependant, à des moments précis, sur des occurrences d'événements prévus ou imprévus, il s'établit des liens difficiles à négliger dans la conduite de chaque élément. Il s'agir de retenir les liens qui demeurent primordiaux.

Ces modes de marche concernent par exemple :

- l'arrêt ou la marche d'une machine,
- la panne ou la marche dégradée,
- un changement de stratégie avec reconfiguration des machines pour l'intégration d'une nouvelle gamme.

La gestion des modes de marche est un problème crucial. Dans le cas d'une ligne de production, les machines étant liées les unes aux autres, l'arrêt de l'une peut provoquer implicitement l'arrêt des autres. Réciproquement une machine ne peut fonctionner si les autres machines ne fonctionnent pas.

La structure même de l'installation influe sur les modes de marche de la cellule. Un poste de travail indépendant, autonome, est moins contraint par rapport aux autres machines. La nature et la densité du système de transport sont des facteurs importants.

Le but terminal de la gestion est donc d'intégrer les différents modes de marche des composants d'une unité de production en tenant compte des contraintes de comportement existantes. Il doit pouvoir appréhender la détection d'incohérences de comportements lors de différents changements d'états des machines à partir de l'analyse de ces états. Il génère, par ailleurs, des actions pour le pilotage de l'unité de production.

Ce logiciel fonctionne en mode interactif pour aider au maximum l'utilisateur dans sa démarche de pilotage. En effet, l'utilisateur peut proposer une commande au gestionnaire qui, alors, l'analysera de façon à vérifier sa cohérence : il faut éviter en effet d'envoyer des ordres erronés vers le process. Le gestionnaire peut alors proposer, en remplacement, une meilleure solution de commande. Dans le cas où la commande serait valide, elle est alors décomposée en actions plus locales et plus fines.

De plus, le gestionnaire permet de donner à l'utilisateur une image exacte du procédé, à tout moment.

### Analogie avec un système d'exploitation temps réel

Nous pouvons faire une analogie des fonctions assumées par notre système avec celles réalisées par un système d'exploitation : nous retrouvons ainsi la notion de processus indépendant comme les machines, les fonctions de gestion de processus telles que la synchronisation, la coordination et le parallélisme de tâches. Ces notions, utilisées dans un système d'exploitation, sont symbolisées par les différentes connexions et contraintes entre machines. La prise en compte du temps y est tout aussi importante [CNR 88].

## II - MODELISATION DES MACHINES

Le but de la représentation est d'avoir la meilleure description possible du système que nous voulons superviser. La qualité de la représentation est importante par le fait qu'elle offre à l'utilisateur, non seulement un aperçu détaillé du système, mais aussi l'assurance d'effectuer une supervision correcte. Cependant, il ne faut pas tomber dans l'excès inverse qui est d'essayer de modéliser trop finement un système. Ceci rendrait la modélisation difficile et avec un degré de complexité supérieur, lourde à mettre en oeuvre et ne permettant pas un temps de réponse opérationnel. Il est donc nécessaire de faire un choix et un compromis afin de savoir quels critères seront à évaluer et à optimiser. Un modèle complet considère simultanément les produits, les machines et éventuellement les opérateurs. Nous nous sommes restreints à une approche "machines" dans notre étude.

Rares sont actuellement les modèles capables d'intégrer les multiples contraintes de fonctionnement [LHO 85], liées à l'exploitation des SAP (Systèmes de Production Automatisés). Citons, en particulier le Gemma (Guide d'Etude des Modes de Marches et d'Arrêts) [ADE 79] et MESAP (Modèle d'Exploitation des Systèmes Automatisés de Production) [PAR 91].

Lorsque l'usager supervise son système de production, la manière la plus intuitive de procéder est d'analyser l'état des machines pour déterminer les commandes à émettre. Ces machines ont des comportements plus ou moins complexes avec des procédures de commandes à respecter et coopèrent souvent entre elles (interactions). Malheureusement, la multitude et la complexité de ces contraintes ne sont pas facilement analysables par un superviseur humain, celui-ci ne pouvant pas appréhender rapidement l'ensemble des contraintes de manière cohérente et optimale.

Le modèle est obtenu à partir d'une analyse qualitative qui permet de prendre en compte les différentes fonctionnalités et connexions entre chaque sous-système élémentaire.

**L'étape de modélisation** de l'univers est la première phase de réalisation d'un système expert. Cette modélisation utilise une notation symbolique qui sera transformée par la suite en base de connaissances.

La notation que nous serons amenés à utiliser, est basée sur des tables de contraintes, des graphes d'état et des représentations arborescentes. En effet, plusieurs modèles de description sont nécessaires, chacun d'eux prenant en compte des notions différentes et indépendantes des autres.

La modélisation permet d'obtenir, en fin d'analyse, une description structuro-fonctionnelle du système, assurant ainsi une décomposition systémique qui caractérise les systèmes de production [LEM 77].

L'aboutissement de la démarche, sera l'élaboration d'une base de connaissances destinée à un système expert dont un prototype est présenté au chapitre III.

**La démarche générale** est illustrée par le schéma suivant (fig 1).

A partir de spécifications initiales, qui comprennent la liste des machines et la description de leurs fonctionnements, deux représentations sont développées : une représentation par graphes d'état ou de comportement et une représentation par tables de contraintes.

L'analyse et le traitement de ces données permettent d'arriver à une représentation structurée fonctionnelle du système de production qui met en évidence les différents liens et niveaux hiérarchiques existants.

Une traduction en arbres de recherche est effectuée. Elle aboutit à la conception d'une base de règles.

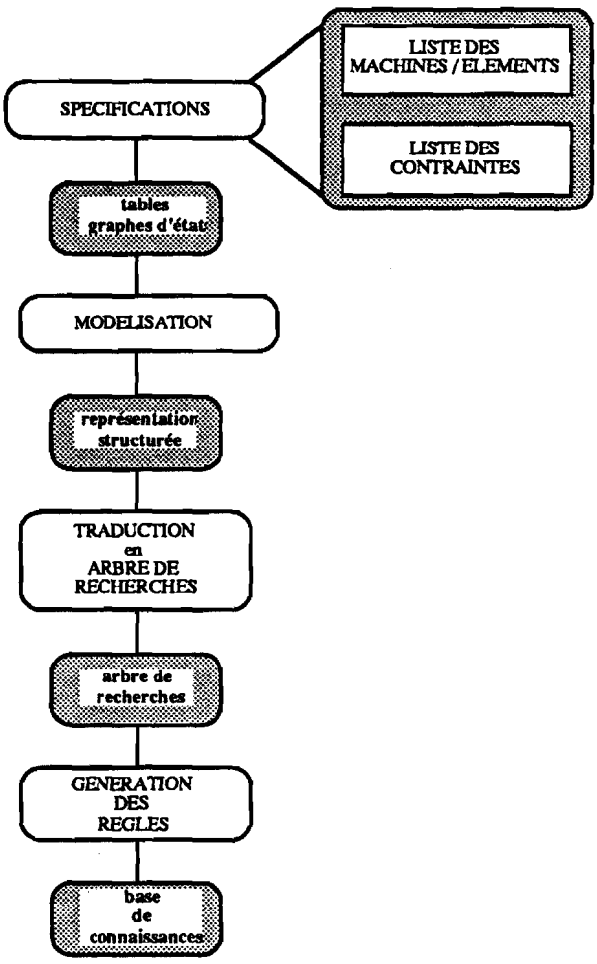



fig 1 : démarche générale

### II . 1 - Description structuro-fonctionnelle

La représentation adoptée est celle qui paraît la plus naturelle, c'est à dire une représentation structurelle de l'unité de production appréhendée par une analyse ascendante du système de production. Nous essayons de retrouver des entités de production telles que les îlots, les cellules et les ateliers, par une démarche de regroupements de type fonctionnel.

Pour cela, nous manipulons deux sortes de machines, **les machines réelles ou effectives, et les machines virtuelles**. L'idée est d'effectuer des regroupements de machines en analysant leurs comportements et les interactions entre ces comportements en nous limitant aux machines commandables.

En outre une décomposition plus fine est possible, en tenant compte des éléments d'une machine : ces éléments ne sont pas commandables mais leur état est susceptible de modifier l'état de la machine qui les supporte, ceci dans le but de faire un minimum de surveillance au niveau du procédé.

## II . 2 - Modélisation du comportement des machines

Nous avons donc restreint la représentation à la modélisation du comportement dynamique de chaque machine. Nous avons, pour cela, utilisé **des graphes d'état.** Cependant, cette modélisation serait incomplète sans la prise en compte des contraintes entre graphes d'état, c'est à dire entre machines. A cet effet, nous avons utilisé **des tables de contraintes.**

### II . 2 . 1 - Les graphes d'état

#### Introduction

Les graphes d'état décrivent le comportement dynamique d'une machine et mettent en évidence les actions susceptibles de modifier son état. Nous avons un graphe d'état par machine, effective ou virtuelle.

Ce graphe d'état est généré à partir du Gemma et est complètement générique. Il regroupe les différents états possibles atteints par une machine. Chaque place du graphe correspond à un état donné de la machine. Les arcs sont associés aux actions logiques ou dynamiques qui permettent de faire évoluer le comportement de la machine.

### II . 2 . 2 - Le Gemma

Le Gemma [ADE 79], Guide d'Etude des Modes de Marches et d'Arrêts, propose une structure initiale d'agencements de modes de marche et d'arrêt. La représentation graphique permet de visualiser le passage d'un mode à l'autre. Ce graphique est adaptable aux besoins des machines et est utilisé dans l'établissement de cahier des charges.

Chaque rectangle correspond à un état possible et les arcs symbolisent les conditions de passages possibles entre états.

La modélisation du système se fait par la représentation de tous les états. Elle permet de décrire précisément le fonctionnement complet sur les causes de passage d'un état à l'autre.

Quatre familles de procédures sont décrites :

- procédures lors de vérifications ou d'essais,
- procédures lors du fonctionnement normal,
- procédures lors de dysfonctionnements,
- procédures lors de mises en arrêt ou de remises en route.

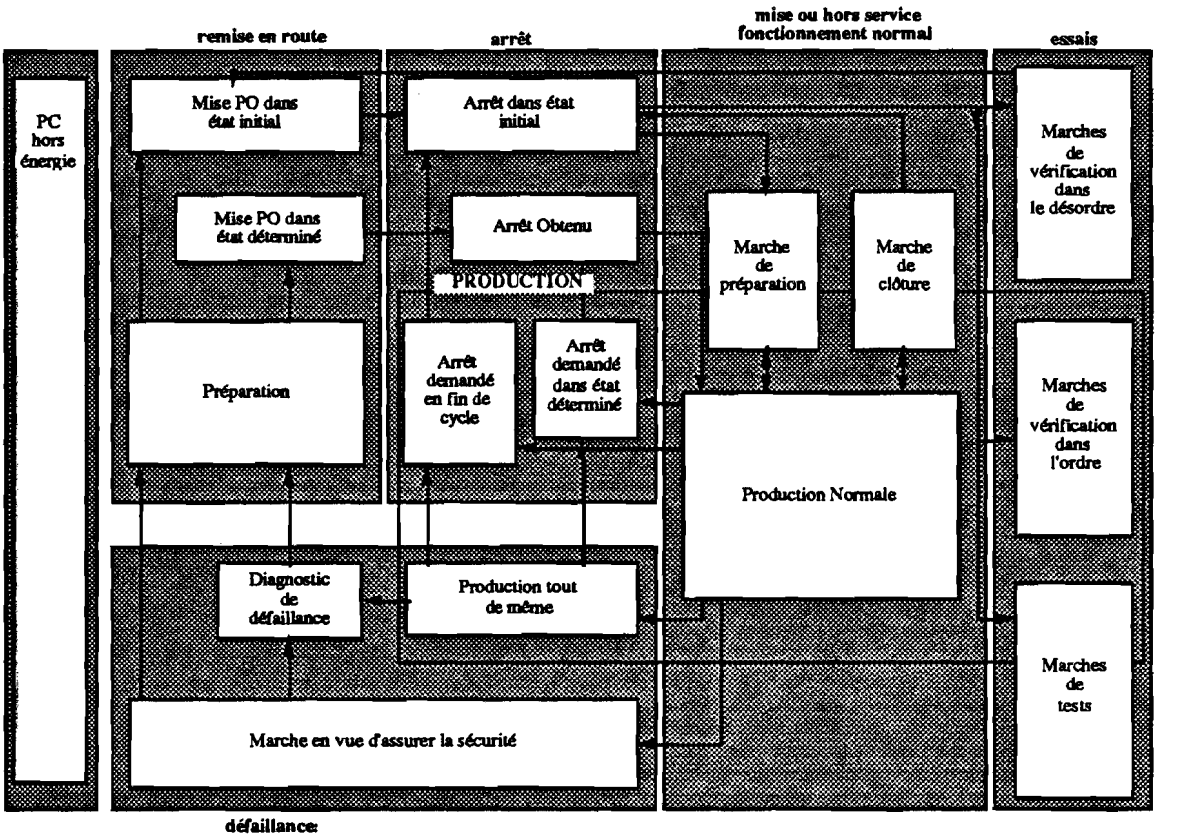



fig 2 : Gemma

Nous décrivons ces différentes procédures [CHA 87]:

### Procédures de vérifications (essais)

Les vérifications dans l'ordre permettent l'analyse de séquences ou de sous-ensembles d'opérations. Les vérifications dans le désordre permettent de contrôler certaines fonctions, sans tenir compte de l'ordre des cycles.

### Procédures de marche (fonctionnement normal)

La marche de production normale correspond au fonctionnement normal. Les marches de préparation et de clôture sont des marches transitoires qui amènent respectivement vers la marche normale et l'arrêt de la production.

### Procédures lors de dysfonctionnements (défaillances)

Elles sont nécessaires lors de l'apparition de défauts. Un diagnostic permet de déterminer la gravité de la panne qui nécessitera éventuellement un arrêt de la machine afin de garantir la sécurité ou une marche forcée, qui correspond à une production forcée.

**Procédures d'arrêt ou remise en route**

L'arrêt normal correspond à l'arrêt sans défaillance, qui peut s'effectuer dans l'état initial, en fin de cycle ou dans un état déterminé. L'arrêt sur défaut est dû à une défaillance et nécessite des procédures de dégagement.

La remise en route peut se faire, soit après une marche de préparation avec une configuration du système, soit directement dans l'état.

Lors de l'implantation, chaque élément peut être représenté par un Grafcet indépendant :

- un Grafcet de marche automatique,
- un Grafcet de marche manuelle,
- un Grafcet de marche cycle/cycle.

### II . 2 . 3 - Représentation adoptée

La représentation retenue est celle de graphes d'états [FAR 87]. Ceux-ci, d'une manière générale, permettent de représenter idéalement le comportement d'un système. Les différents états possibles atteints sont mis en valeur et les évolutions possibles entre états sont représentées par des arcs. De plus, dans notre cas, cette représentation convient bien car le niveau d'abstraction est relativement élevé. Néanmoins, si le système est plus complexe, son graphe d'état sera rapidement très lourd, et inutilisable.

Nous considérons la gestion de ces états comme un problème global. L'ensemble des états du problème est représenté par un graphe orienté. Chaque sommet est un état possible et chaque arc du noeud A au noeud B correspond à une action ou à une transformation qui permet de passer de l'état A à l'état B. En général, un noeud initial est défini, qui sera l'origine de la recherche. L'état final peut être n'importe quel autre état. Une solution au problème est de déterminer un chemin entre l'état initial et l'état final. Le graphe peut comporter des circuits ou des cycles, synonymes de plusieurs solutions. Le processus de recherche est équivalent à la stratégie de contrôle.

A la différence du Gemma qui ne distingue pas, au niveau de sa représentation, les états de la manière d'accéder à ces états, nous considérons en faits certains de ses états, comme des actions transitoires qui permettent de changer d'état. La modélisation par graphes enrichit donc la description.

De plus, nous avons distingué le comportement d'une machine effective de celui d'une machine virtuelle, combinaison de plusieurs machines effectives ou virtuelles.

### II . 2 . 4 - Cas d'une machine effective

Chaque machine effective a son comportement dynamique modélisé par un graphe d'état qui reste spécifique à la machine. Ceci suppose évidement que le comportement de la machine soit complètement déterministe.

Les éléments modélisés sont des objets commandables, c'est à dire qu'il est possible physiquement de modifier leurs états et de pouvoir connaître leurs situations internes.

### II . 2 . 5 - Les différents états possibles

Nous avons défini les états possibles d'après ceux proposés par le Gemma, présenté figure 2.

Les différents états possibles sont :

- l'état **TENSION-OFF** : état 0 qui correspond à l'état hors tension de la machine. La production est considérée comme arrêtée et la machine n'est plus utilisable,

- l'état **TENSION-ON** : état 1 qui correspond à l'état sous tension de la machine. La machine est prête à produire. Cet état peut être considéré comme un arrêt en fin de cycle ou dans un état déterminé ou comme l'état d'attente d'une "reprise à chaud",

- l'état **INITIALISE** : état 2 qui correspond à un état configuré pour une reprise. Cet état peut être considéré comme l'état d'attente d'une "reprise à froid",

- l'état **MARCHE NORMALE** : état 4 qui correspond à la marche normale de la cellule. C'est l'état vers lequel le système doit tendre, c'est à dire la production effective,

- l'état **MARCHE DEGRADEE** : état 5 qui correspond à une marche sous défaut de la machine. Un dysfonctionnement a été détecté mais la production n'est pas forcément arrêtée avec éventuellement un forçage de données pour continuer à fonctionner,

- l'état **ARRET_SOUS_DEFAUT** : état 3 qui correspond à un arrêt de la tension après la détection d'une erreur. La production a été stoppée et est en attente de reconfiguration.

Ces états sont liés par des arcs correspondant à des actions effectives du type ordre de mise en marche ou à une détection d'un événement susceptible de modifier l'état de la machine.

### II . 2 . 6 - Description des passages d'un état à un autre

Ce sont en fait des transitoires entre états, qui correspondent à des actions ou des occurrences d'événements non contrôlés.

**Passage de l'état 0 vers l'état 1 :** ceci correspond à la mise sous tension de la machine.

**Passage de l'état 1 vers l'état 0 :** ceci correspond à l'arrêt de la tension de la machine associée en général à des procédures d'arrêt de tension.

**Passage de l'état 1 vers l'état 1 :** ceci correspond à une opération de vérification de certaines fonctions ou de cycles d'opérations.

**Passage de l'état 1 vers l'état 2 :** ceci correspond à l'initialisation et à la préparation vers une marche.

**Passage de l'état 1 vers l'état 3 :** ceci correspond au passage à l'état "arrêt_sous_défaut". Un défaut a été détecté, en général après une vérification des fonctions, nécessitant un traitement de remise en état.

**Passage de l'état 1 vers l'état 4 :** ceci correspond à la mise en marche de la machine sans configuration initiale. Le plus souvent, c'est une reprise directe après un arrêt de la marche appelée "*reprise à chaud*".

La "reprise à chaud" est une mise sous tension de la machine avec une mémoire correcte de données.

**Passage de l'état 2 vers l'état 0 :** ceci correspond à l'arrêt de la tension à partir d'un état initialisé. Cela peut survenir lors d'une marche normale, lorsque une machine est en fin de cycle et est stoppée normalement.

**Passage de l'état 2 vers l'état 3 :** ceci correspond à la détection d'un défaut qui apparaît lors de l'étape d'initialisation, nécessitant un traitement spécifique.

**Passage de l'état 2 vers l'état 4 :** ceci correspond à la mise en marche après une initialisation et une configuration de la machine en vue d'une marche. En général, c'est une démarche nécessaire lors du démarrage du système appelée "*reprise à froid*".

La "reprise à froid" est le redémarrage de la machine avec la perte de la mémoire de données.

**Passage de l'état 3 vers l'état 0 :** ceci correspond à l'arrêt de la machine dans un état sous défaut. Les procédures de dégagement ne permettent pas de revenir dans une situation satisfaisante et une intervention plus poussée nécessitant l'arrêt de la machine, se révèle nécessaire.

**Passage de l'état 3 vers l'état 1 :** ceci correspond au retour vers une situation normale avec une correction de la machine. Ces opérations peuvent être manuelles avec l'intervention d'un opérateur.

**Passage de l'état 4 vers l'état 1 :** ceci correspond à l'arrêt normal de la machine. Cet arrêt peut être associé à des procédures de terminaison.

**Passage de l'état 4 vers l'état 3 :** ceci correspond à la détection d'un défaut de la machine, qui impose un arrêt immédiat de la machine.

**Passage de l'état 4 vers l'état 5 :** ceci correspond à la détection d'un défaut de la machine, en cours de production. Ce dysfonctionnement ne nuit pas à sa marche mais impose certaines contraintes de marche et restreint ses possibilités d'utilisation.

**Passage de l'état 5 vers l'état 3 :** ceci correspond à l'arrêt sous défaut de la machine. Une situation possible est que le dysfonctionnement s'aggrave, ce qui entraîne l'arrêt de la production. Une autre situation possible est l'arrêt de la machine en vue d'un recouvrement d'erreurs.

Les passages qui ne sont pas présentés sont considérés comme inexistants et impossibles.

Le graphe d'état présenté est générique.

**Remarque 1** : nous n'avons pas distingué dans l'approche, l'arrêt propre de l'arrêt induit [ROD 89].

**L'arrêt propre** subi par une machine est un arrêt dont elle est à la fois la cause et la victime. Ceci peut être dû à un arrêt normal de fin de production.

**L'arrêt induit** signifie qu'une machine a été arrêtée à la suite de l'arrêt d'une autre, sans avoir pour cela fini de produire. cet état correspondant à celui d'un état d'attente en vue d'une reprise lorsque l'élément perturbateur aura été réparé.

Nous aurions donc pu distinguer ces deux états au niveau du graphe de comportement.

**Remarque 2** : il est possible d'avoir des graphes d'état plus ou moins différents. En effet, le graphe reste spécifique à chaque machine et demeure dépendant du comportement effectif de la machine et de ses commandes possibles. En effet, dans le cas d'un comportement plus simple d'une machine, le graphe peut être simplifié et réduit. La représentation de base prend normalement en compte la liste exhaustive des états possibles.

De même, les éléments simples non commandables ont un graphe associé beaucoup plus simple. Leurs états évoluent de manière totalement asynchrone. Ces éléments peuvent être un capteur, un outil, un coupleur de communication...

Le graphe de base se présente comme suit (fig 3):

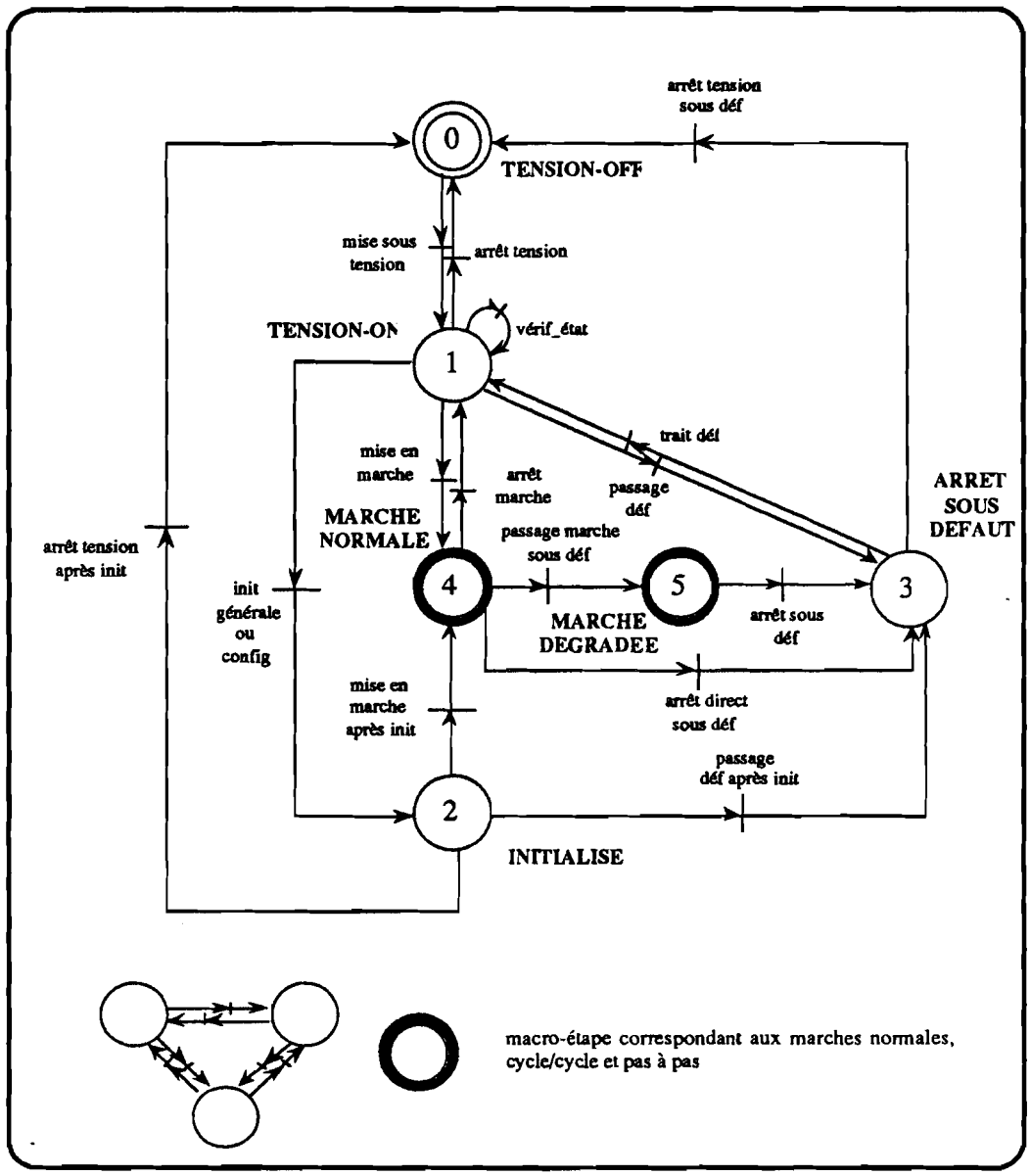



fig 3 : graphe d'état d'une machine effective

Ce graphe d'état reproduit donc le comportement dynamique et logique d'une machine. Nous avons intégré des macro-étapes qui correspondent, soit à une marche normale ou automatique, soit à une marche cycle par cycle, soit pas à pas. La différence principale réside dans le fait que le mode automatique ne permet pas une commande en ligne. Les marches non automatiques peuvent correspondre à la commande en ligne ou manuelle de la machine par un utilisateur. Nous avons regroupé ces trois états car leurs relations vis à vis des autres états possibles du système sont identiques.

**Description d'une Macro-étape**

La description d'une macro-étape est détaillée ci-après (fig 4). Il est possible de passer à partir d'un état à tous les autres, moyennant quelques actions associées.

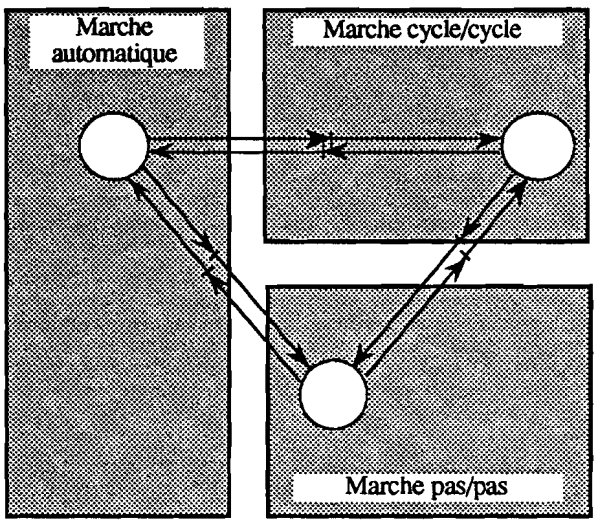



fig 4 : macro-étape

**II . 2 . 7 -** Exemple d'un convoyeur

Sa structure de commande est très simple. Un convoyeur ne peut prendre, en général, que deux états : l'état "tension-off" et l'état "marche normale". Sa mécanique est assez fiable pour ne pas envisager de pannes. Les états "marche dégradée", "arrêt_sous_défaut" et "initialisé" n'existent donc pas.

L'état "marche normale" et l'état "tension-on" sont agrégés. L'état "tension-off" correspond à l'arrêt de la marche du convoyeur.

Son graphe d'état associé est donc très simplifié par rapport au graphe d'état initial.

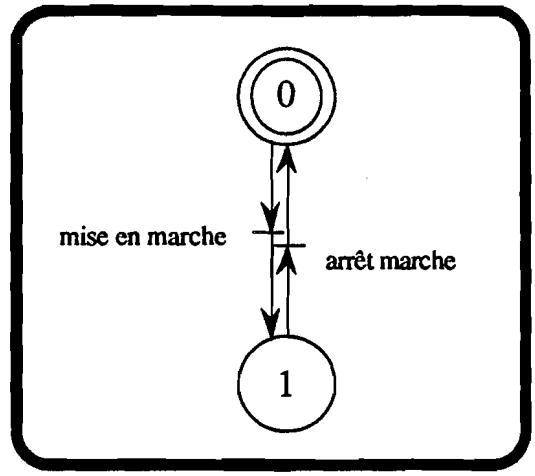



fig 5 : graphe d'état d'un convoyeur

### II . 2 . 8 - Cas des éléments non commandables

Ce sont des composants de machines, sur lesquels il n'est pas possible d'agir, c'est à dire envoyer une commande ou un ordre. Ce sont en général des effecteurs ou outils, ou des capteurs, voire des modules de communication.

Nous nous restreignons aux objets importants dans la marche d'une machine, c'est à dire ceux susceptibles de modifier son état.

Par exemple, il peut s'agir d'outils d'une machine d'usinage. Si un outil casse lors d'un usinage, la machine de fabrication n'est alors plus utilisable et peut être considérée comme en panne. La procédure de recouvrement consiste alors à changer d'outil pour revenir dans un mode de fonctionnement normal.

De même, un robot n'est plus utilisable si ses capteurs ne fonctionnent plus, ceux-ci étant ses seuls moyens de perception (caractère critique de ces éléments).

Cette prise en compte d'éléments non commandables permet d'améliorer et de préciser la représentation.

#### Graphe d'état associé

Il est possible d'associer un graphe d'état aux éléments afin de modéliser leurs comportements.

Trois états sont seulement distingués :

- **un état correct** : l'élément est neuf ou dans un état parfait. Il n'a pas été détérioré lors de ses utilisations antérieures,

- **un état défectueux** : l'élément est défectueux. Ceci peut correspondre à un outil brisé ou qui a dépassé son degré tolérable d'usure (considéré comme "mort"),
- **un état utilisable :** l'élément se trouve dans un état intermédiaire. Il n'est pas parfait, mais il s'agit de tenir compte d'éventuelles anomalies, à la suite de son utilisation. Il peut s'agir d'outils qui ne sont pas suffisamment usés pour nuire à la précision de leurs tâches.

Nous n'avons pas distingué d'état de marche. En effet, ces objets sont "inertes" ou "inanimés"et il n'est pas possible de leurs considérer des états dynamiques.

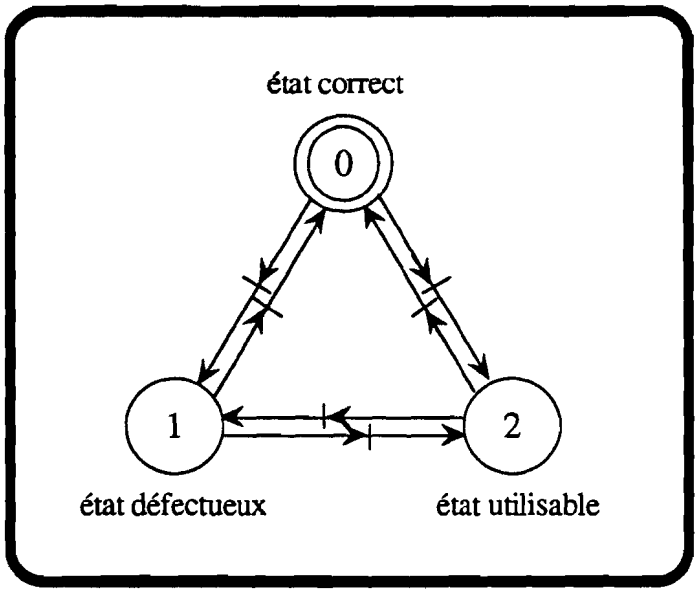



fig 6 : graphe d'état d'un objet non commandable

La détection d'un défaut constitue une information qui peut être envoyée, soit directement par l'intermédiaire d'un capteur, soit indirectement par l'intermédiaire d'une personne chargée de la surveillance qui renseignera la base de connaissances ou par un système spécialisé dans la détection de défauts, associée éventuellement, avec un diagnostic précis.

La correction d'un défaut nécessite, en général, une intervention manuelle. Si la correction est effective alors l'information associée sera transmise afin de mettre à jour la base de données. Une intervention automatique est possible, avec l'utilisation d'un changeur automatique d'outils. Souvent, des positions de repli sont nécessaires.

### II . 2 . 9 - Cas du fonctionnement d'un automate TSX 67

Les automates TSX de la série 7 de la Télémécanique sont des automates multi-tâches et multi-fonctions. De façon à les rendre complètement autonomes et réactifs vis à vis de

l'extérieur et en particulier des dysfonctionnements, des procédures concernant les modes de marche ont été établies.

Des dispositifs de surveillance détectent par exemple les coupures secteurs. Des procédures de recouvrement de coupures secteurs ou de défauts sont prévues.

Un Gemma simplifié est construit à partir de ces procédures.

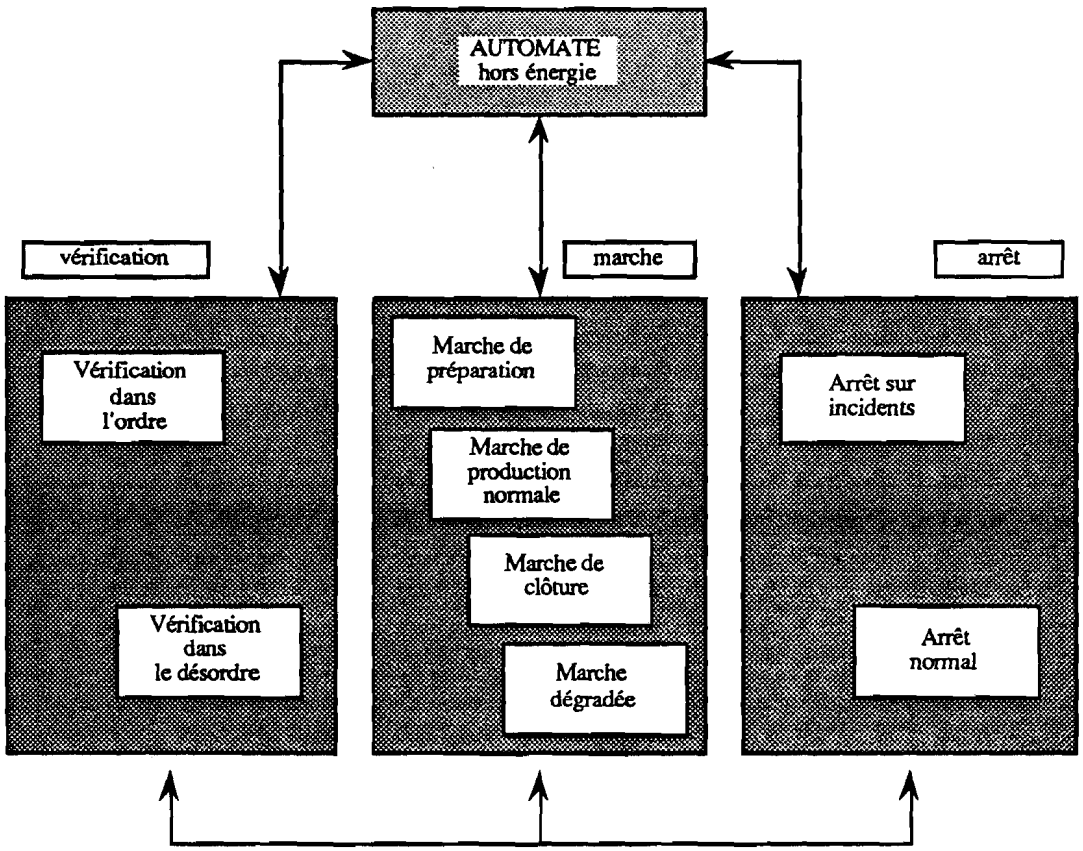



fig 7 : Gemma simplifié [TEL 87]

Le Gemma proposé par Télémécanique est une version simplifiée. Trois groupes de procédures sont distingués :

- procédures de vérifications,
- procédures de marche,
- procédures d'arrêt.

**Graphe d'état associé :**

Certains états proposés par la Télémécanique sont considérés dans notre démarche, en fait, comme des états transitoires : marche de préparation, vérification dans l'ordre et dans le désordre, et marche de clôture. Nous construisons directement le graphe d'état à partir du Gemma.

Nous avons associé aux transitions de notre graphe et dans la mesure du possible, les bits système SYi définis par le constructeur. Ils correspondent typiquement à des capteurs permettant de connaître l'état courant du système et ses phases transitoires ; ils peuvent également faire office "d'actionneurs" pour forcer un changement d'état de l'automate. Notre point de vue est donc ici de considérer l'automate programmable comme une machine de production et de le surveiller de façon identique.

Ce sont en l'occurrence les bits :

- SY0 qui traite la reprise à froid, accessible en écriture par programme,
- SY1 qui traite la reprise à chaud, accessible en test par programme,
- SY2 qui traite la reprise immédiate, accessible par programme,
- SY21 qui initialise le Grafcet, accessible par programme,
- SY22 qui remet à 0 le Grafcet, accessible par programme,
- SY23 qui fige le Grafcet, accessible par programme.

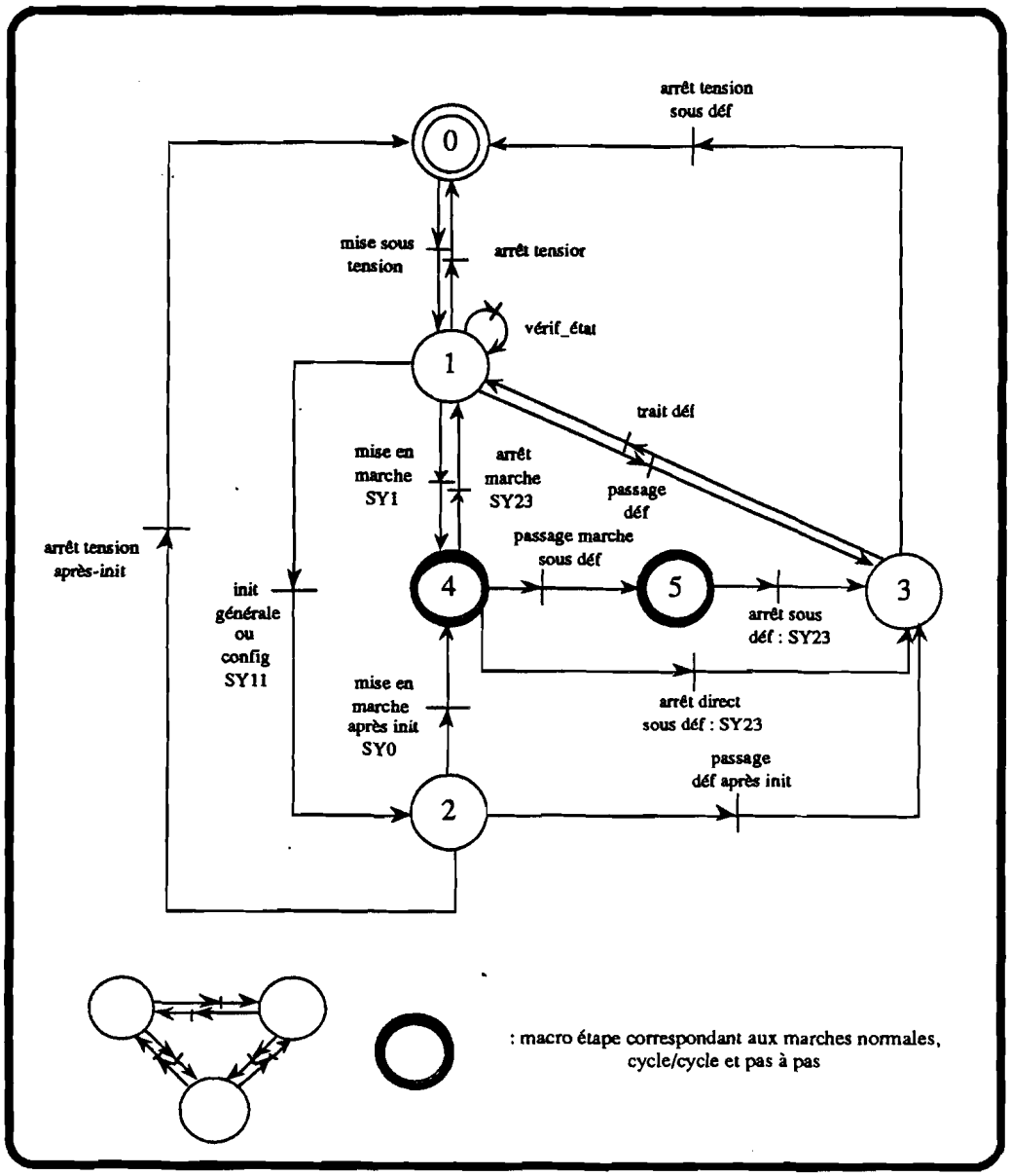



fig 8 : graphe d'état équivalent

## II . 2 . 10 - Modèle d'implantation sur l'automate

Ceci est un modèle d'implantation du graphe d'état sur un automate programmable. Il est décrit en langage Grafcet. La transcription est simple. Nous avons une étape par état.

Ce Grafcet peut être utilisé par exemple, dans la configuration suivante : un groupe d'automates de bas niveau est supervisé par un automate de haut niveau, qui possède une image de l'état de chacune des machines par l'intermédiaire d'un Grafcet. Cela lui permet de connaître et de maîtriser le comportement de chacun des automates qu'il supervise lors de leurs marches.

Ces graphes peuvent évoluer de manière indépendante ou être commandés par une autre machine.

En effet, en considérant que l'automate fasse partie d'une machine virtuelle, son état dépend de celui d'autres sous-machines. En fonction de la situation, chaque transition peut être commandée afin de faire évoluer le Grafcet associé.

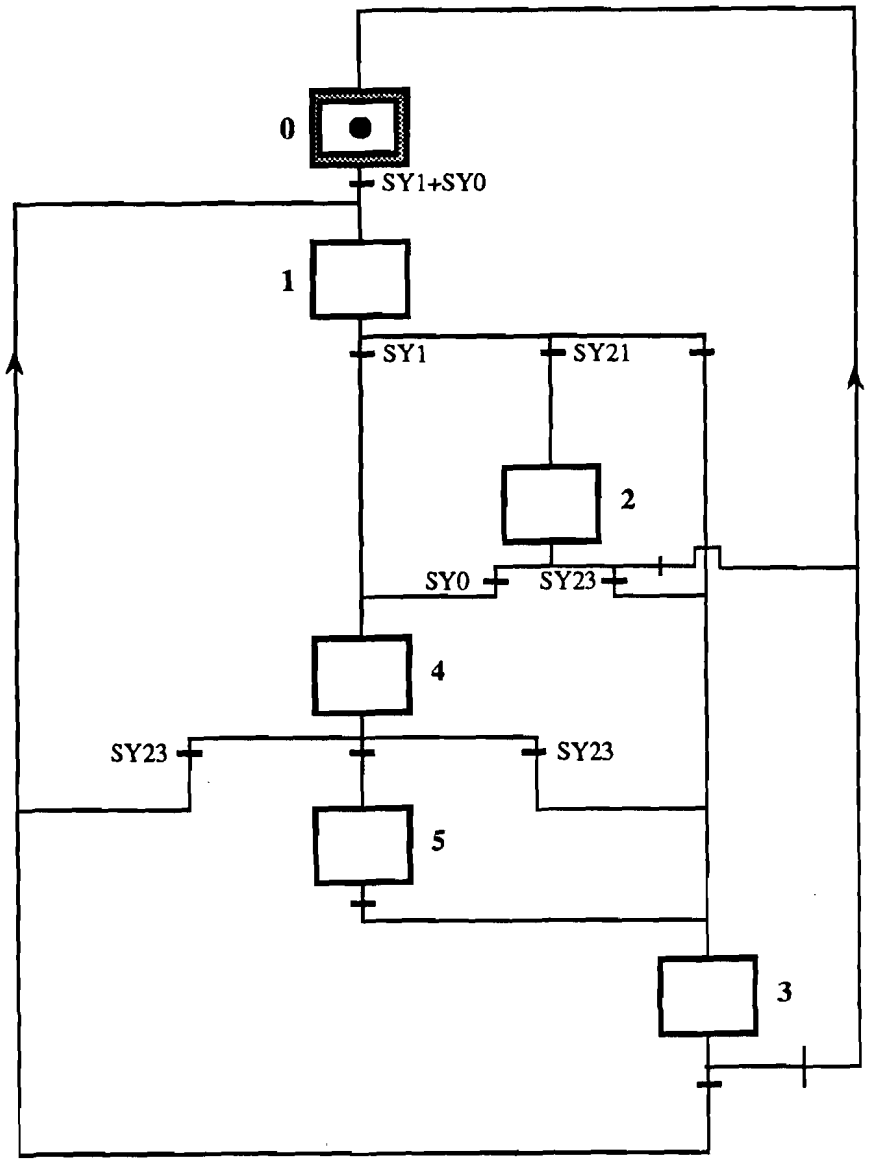



fig 9 : Grafcet associé

L'étape initiale est l'étape 0.

Le passage de l'état 0 à l'état 1 se fait par l'intermédiaire de SY1, qui détecte une reprise secteur, sans perte de données, ou de SY0, qui détecte une reprise secteur avec perte de données.

Pour une mise en marche (passage à l'état 4) qui correspond à une reprise à chaud, nous testons le bit système SY1, qui indique si l'opération est possible. Cette mise en marche peut être demandée, par exemple, par une autre machine via une requête réseau.

De même, les autres changements d'états interagissent avec des bits systèmes qui paramètrent le Grafcet et le font évoluer. Plusieurs bits système peuvent être associés à une même transition.

### II . 2 . 11 - Cas d'une machine virtuelle

#### II . 2 . 11 . 1 - Définition

Une machine virtuelle est un regroupement de sous-machines. La composition d'une machine virtuelle est réalisée lorsque des machines ont des comportements liés par des contraintes de fonctionnement. Celles-ci présentent des caractères impératifs qui rendent les analyses de comportement de chaque sous-machine complètement indissociables les unes des autres. Une sous-machine du regroupement, peut être soit une machine effective, soit une machine virtuelle.

Le comportement d'une machine virtuelle est fonctionnellement identique à celui d'une machine effective et son état dépend de l'état de ses sous-machines. Ces états sont semblables. Nous retrouvons donc la même structure initiale de graphe. Cependant ce graphe est enrichi pour prendre en compte les états particuliers de la machine virtuelle.

En effet, si l'une des sous-machines change d'état, l'état global de la machine virtuelle ne correspond plus à son état effectif. Il n'y a plus de correspondance entre les différents états des sous-machines et nous pouvons considérer que l'état de la machine virtuelle passe par des états instables ou états transitoires.

La sélection de l'état transitoire sera fonction du calcul d'état de la machine virtuelle, effectué en fonction de l'état de l'ensemble des sous-machines.

Une machine virtuelle est dans un état dit "**instable**" si une, au moins, de ses sous-machines se trouve dans un état incompatible vis à vis de celui des autres sous-machines ou lorsque la conformité des états de ses sous-machines n'a pas été vérifiée.

Par opposition, un état est dit "**stable**" lorsque l'état de toutes les sous-machines sont en conformité et ne présentent pas d'incohérence de fonctionnement par rapport aux contraintes de comportement.

Un état instable est en fait un état transitoire entre deux états stables. Le principe de gestion est de revenir dans un état stable le plus vite possible. L'instabilité demeure jusqu'à ce que le système de gestion génère des actions de façon à corriger l'état de la machine virtuelle.

### II . 2 . 11 . 2 - Représentation

Nous avons choisi de ne représenter qu'un seul état transitoire entre deux états stables de façon à ne pas surcharger la représentation. En effet, le nombre d'états instables dépend du nombre de sous-machines et du nombre d'états qu'elles peuvent prendre. Cela peut donc induire une complexité combinatoire importante, difficile à représenter. Le but du système de gestion est de gérer ces transitoires de façon à en limiter le temps d'évolution et leur nombre.

Le graphe d'état d'une machine virtuelle est présenté ci dessous :

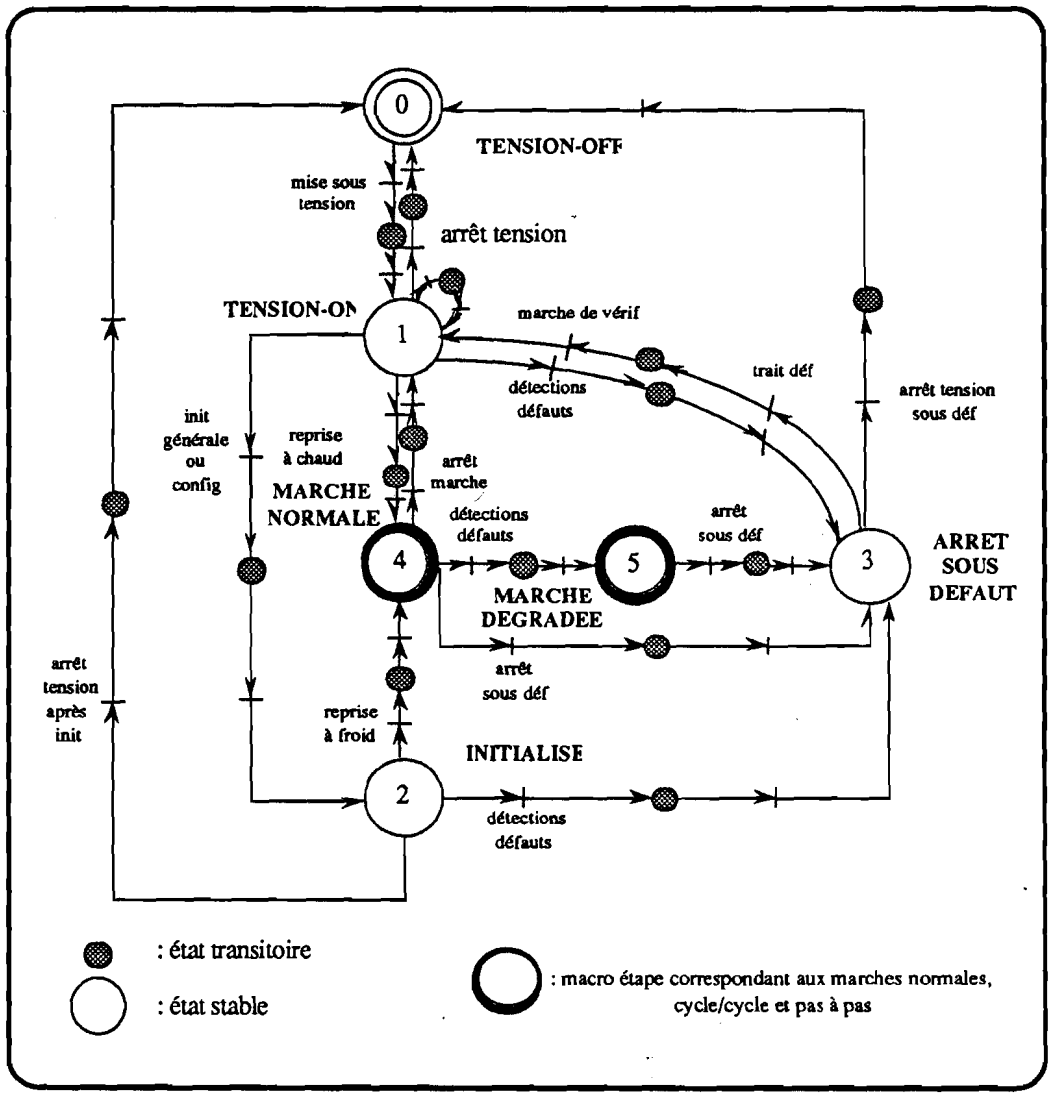



fig 10 : graphe d'état d'une machine virtuelle

### II . 2 . 11 . 3 - Vue éclatée d'un état transitoire

Un état transitoire est en fait une agrégation des différents états transitoires possibles. Cela dépend du nombre n de sous-machines et est égal à $2^n$ -1. Il suffit qu'une des sous-machines change d'état pour passer dans un état transitoire.

Nous avons, par exemple, une machine virtuelle constituée de trois sous-machines. Pour passer de l'état 1 à l'état 2, il existe un état transitoire. Il comporte 7 états différents. Si la machine virtuelle se trouve dans l'état 1, cela signifie que ses sous-machines sont dans le même état ou dans un état compatible.

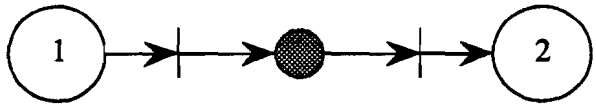



fig 11 : état transitoire agrégé

Prenons le cas où elles sont toutes dans le même état 1. Si l'une d'elles change d'état (passe à 2), alors un des états transitoires est activé.

Nous représentons les 7 états intermédiaires possibles : x/y/z sont respectivement l'état de chaque sous-machine (1 ou 2).

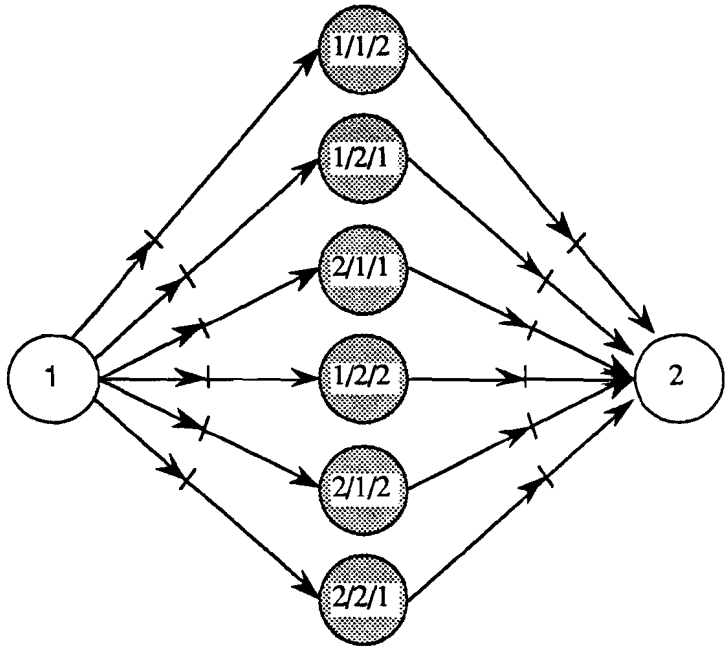



fig 12 : les différents états intermédiaires

Une machine virtuelle est donc composée de sous-machines, virtuelles ou effectives, contraintes entre elles et dont les contraintes avec les autres machines de niveau équivalent dans la structure hiérarchisée sont de même type (sur état ou changement d'état).

### II . 2 . 12 - Les changements d'états

Ce sont les opérations susceptibles de modifier le comportement des machines et en l'occurrence les marquages des graphes d'état. Cela peut être des ordres émis par l'opérateur ou des événements extérieurs, indépendants de l'opérateur.

Les actions possibles, contrôlées par le système de pilotage, sont par exemple :

- des ordres d'arrêt, d'initialisation ou de marche de machines,
- des vérifications de marche.

Les événements extérieurs sont des occurrences non maîtrisées par le système de pilotage et sont totalement asynchrones. Ce sont par exemple :

- une panne de machine,
- un stock vide de pièces.

Les ordres sont associés, par exemple à des actions de reconfiguration d'un graphe de commande servant à coordonner la partie procédé de l'atelier. Ces reconfigurations sont spécifiques aux actions et servent à limiter les indéterminismes de fonctionnement et de routage des pièces, dûs en particulier aux problèmes de transitoires entre modes de marche.

Un logiciel [GEN 90], développé en Le_Lisp [CHA 89] et associé au système de pilotage, permet de modifier dynamiquement et de manière transparente le graphe de commande.

**Exemple** : dans la cas d'un automate TSX de la série 7, chaque changement de mode de marche (mise sous tension, reprise secteur) occasionne en particulier :

- une initialisation du Grafcet,
- un prépositionnement d'étapes ou de données,
- un gel de graphe.

Des bits "système" sont associés à chaque type de changement. Ces données fournissent des renseignements sur l'état de l'automate ou permettent de traiter les changements.

## III - MODELISATION DES CONTRAINTES

### Introduction

Les contraintes sont les liens qui existent entre des machines lors de la production, introduites par les impératifs de l'automatisation avancée de la production. Elles permettent de définir et de spécifier les multiples dépendances qui existent entre les machines et leurs tâches.

Ces contraintes de fonctionnement entre machines introduisent en fait des relations de dépendance entre les graphes d'état.

L'absence de contrainte entre machines traduit implicitement, en fait, une relation d'indépendance entre elles, qui n'est pas modélisée lors de la représentation.

La connaissance de ces contraintes nécessite une approche assez fine des machines avec l'analyse détaillée des relations inter et intra-ressources, travail d'ailleurs considéré comme phases initiales de spécifications lors de l'élaboration d'un système expert.

### III . 1 - Modélisation

Nous proposons de préciser le type et la liste des contraintes pouvant intervenir entre machines.

Ces contraintes peuvent concerner soit des états bien précis des machines, soit le comportement global des machines. Elles sont enregistrées sous la forme d'une table ou d'une matrice de couplage.

Ainsi nous pouvons distinguer deux sortes de contraintes :

- relations sur les états,
- relations sur les changements d'états.

**Les relations sur états** sont des contraintes qui sont vérifiées sans tenir compte des actions et de leurs combinaisons qui mènent à ces états. N'est pris en compte que le caractère stable ou statique des états atteints. Le seul impératif retenu est que les états correspondent aux règles de fonctionnement imposées par les contraintes. Ceci permet d'éviter d'obtenir, par exemple, des machines dans des états mutuellement incompatibles.

**Les relations sur changement d'état** considèrent les actions qui mènent aux états avec des impératifs supplémentaires. Celles-ci peuvent traduire des relations de précédence à caractère dynamique. Une tâche doit être effectuée avant ou après une autre bien spécifiée. Il n'est possible de déclencher l'action qu'en étant sûr de l'état de la machine sur laquelle s'applique l'action. Ceci permet d'éviter l'envoi d'une séquence d'opérations totalement incohérentes.

### III . 2 - Liste des types de relations

Nous présentons les différentes contraintes existantes ainsi que les représentations arborescentes associées. Ce sont ces descriptions qui sont retenues lors de la représentation définitive de système de production.

Ces contraintes sont :

**des contraintes sur états :**

- contrainte de coopération **CC**,
- contrainte de coopération partagée **CCP**,
- contrainte d'exclusion **CE**,
- contrainte structurelle **CS**
  - * contrainte structurelle série **CSS**,
  - * contrainte structurelle parallèle **CSP**.

**des contraintes sur changements d'états :**

- contrainte procédurale **CPRO**,
- contrainte d'observabilité **CO**.

Nous allons les présenter par la suite, plus précisément. Ceci n'est pas une liste exhaustive des contraintes possibles. Ce sont celles que nous avons identifiées, il est vraisemblable que d'autres existent, liées aux modes de fonctionnement des machines.

### III . 3 - Représentations possibles

Deux structures sont possibles et se rapprochent, soit d'un arbre de type OU, soit d'un arbre de type ET.

Il s'agit en fait de variantes d'arbres ET/OU. Nous modifions la relation ET/OU par l'intégration d'une contrainte qui paramètre le type de la relation. Le choix de ET/OU se fait en tenant compte du caractère flexible de la contrainte.

### III . 3 . 1 - L'arbre ET

L'arbre ET est utilisé pour représenter des contraintes inclusives, c'est à dire que l'état d'une machine est conditionné par le fait que l'état de sa machine associée doive être dans un autre état.

Une contrainte de type ET restreint la notion de flexibilité (dépendance conjointe de 2 machines).

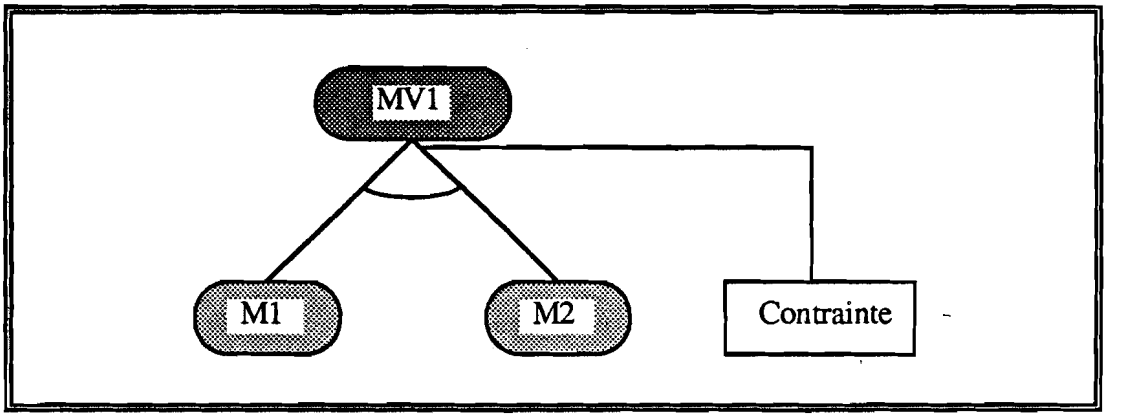



fig 13 : arbre ET

Cette représentation signifie que deux machines sont associées en une machine virtuelle, avec une contrainte qui se rapproche d'un ET avec variante (M1 et M2 sont ici fondamentalement dépendantes).

### III . 3 . 2 - L'arbre OU

L'arbre OU est utilisé pour représenter des contraintes moins fortes que celles de type ET. Ce type de contrainte engendre un degré supplémentaire de flexibilité au niveau du fonctionnement des machines.

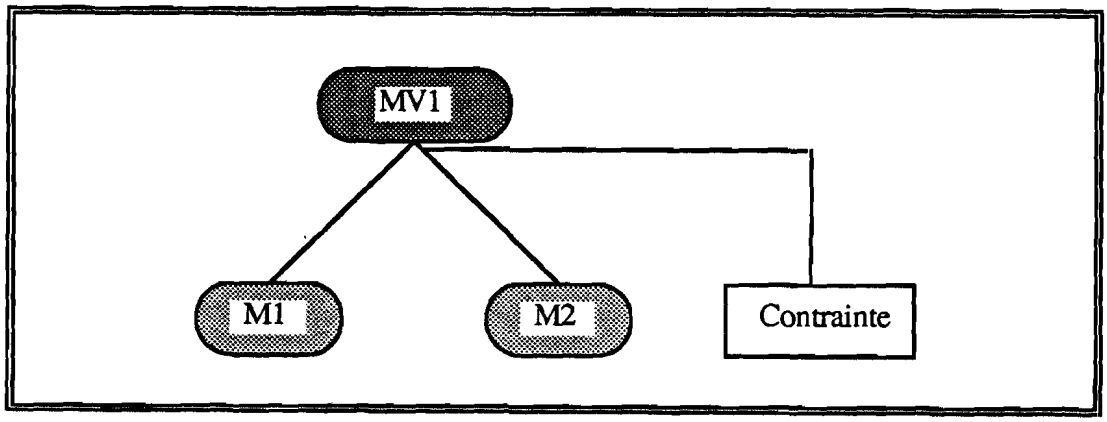



fig 14 : arbre OU

Cette représentation signifie que deux machines sont associées en une machine virtuelle, avec une contrainte qui se rapproche d'un OU avec variante. Dans ce cas, les états des deux machines M1 et M2 sont disjoints : cas de machines qui n'ont strictement aucun lien entre elles.

### III . 3 . 3 - Contrainte de coopération CC

C'est le cas le plus couramment rencontré. Deux ou plusieurs machines coopèrent de manière étroite et sont fortement dépendantes les unes des autres. Par exemple, dans le cas d'une machine d'usinage et de son robot de chargement/déchargement, il suffit qu'une des machines tombe en panne pour modifier le comportement de l'autre machine. Dans les deux cas, il est préférable de mettre les machines qui sont en état de marche, en position de repli. Cette contrainte sur état reste spécifique aux états de marche.

**Représentation arborescente**

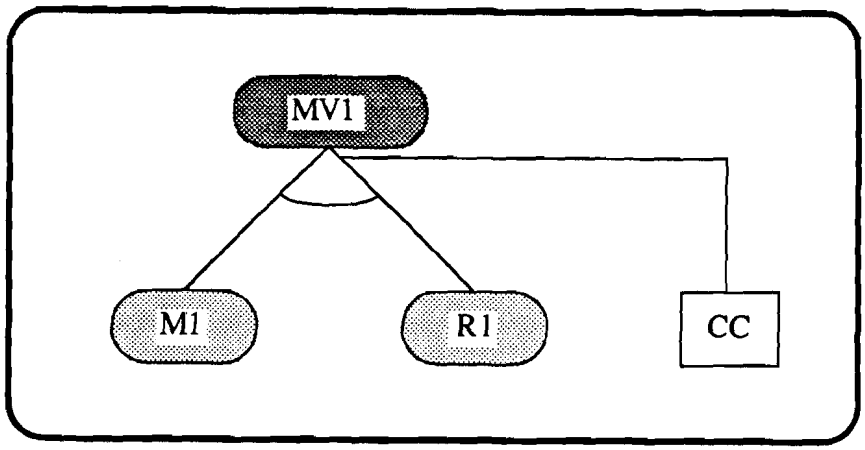



fig 15 : coopération totale

Une machine virtuelle MV1 est composée d'un robot et de la machine à laquelle il est associé. La représentation utilise une structure de ET. Cela montre la relation étroite qui lie les deux sous-machines. L'état de M1 doit correspondre à celui de R1.

$$MV1 = M1 \; CC \; R1 \Leftrightarrow E(M1) = E(R1)$$

Mi $\mathcal{R}$Mj signifie que les machines sont liées par la relation $\mathcal{R}$.
E(M) donne l'état de la machine M.

### III . 3 . 4 - Contrainte de coopération partagée CCP

Deux ou plusieurs machines partagent une ressource. Ces machines peuvent fonctionner de manière parallèle et indépendante et n'avoir aucun lien particulier à l'exclusion de demande d'utilisation à certains instants, d'une même machine comme ressource partagée.

**Représentation arborescente**

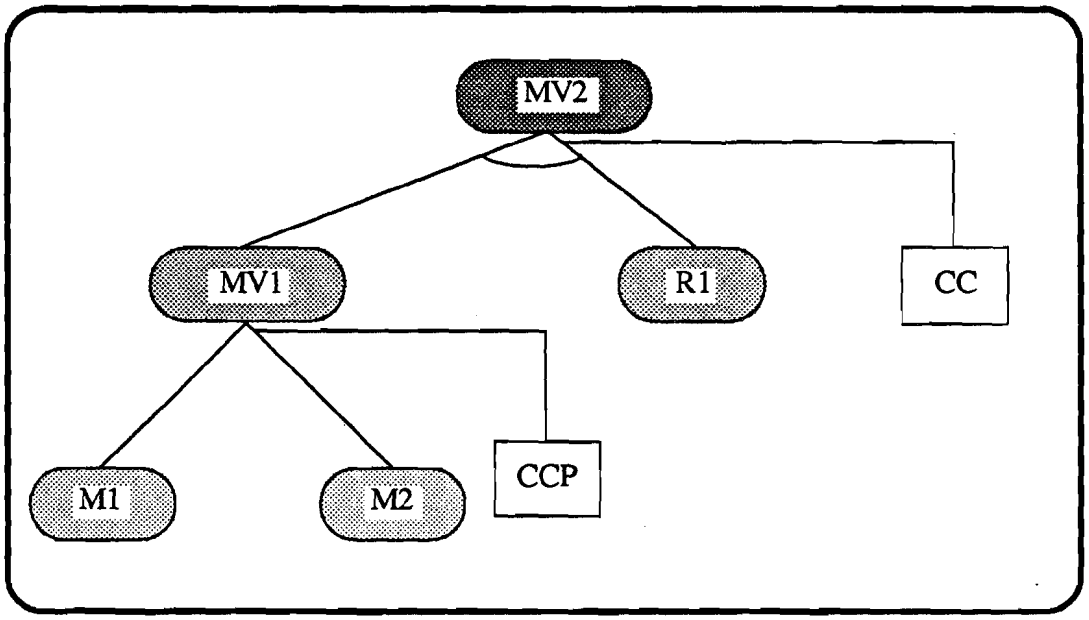



fig 16 : coopération partagée

MV1 est composée des deux sous-machines qui utilisent la ressource R1. Ces deux machines M1 et M2 fonctionnent en coopération partagée. C'est typiquement une relation de type OU : M1 et M2 ne sont liées que par le fait d'utiliser une même ressource et n'ont pas de lien direct.

R1 et MV1 constituent une machine virtuelle MV2 avec une contrainte CC. C'est en fait un cas de coopération entre MV1 et R1. L'état de R1 doit correspondre à celui de MV1, c'est à dire soit à celui de M1, soit à celui de M2.

MV2= MV1 **CCP** R1 et MV1 = M1 **CC** M2 <=> E(MV1) = E(R1)

c'est à dire E(M1) = E(R1) $\oplus$ E(M2) = E(R1)

### III . 3 . 5 - Contrainte d'exclusion CE

C'est le cas inverse de la coopération. Deux ou plusieurs machines se trouvent ensemble dans des états bien déterminés. L'état de ces machines est complètement exclusif, au même moment. La réalisation d'une action de l'une exclut la réalisation d'une action simultanée de l'autre.

Par exemple, ce sont deux machines qui ne peuvent être simultanément en marche. Cette contrainte sur état reste spécifique aux états de marche.

**Représentation arborescente**

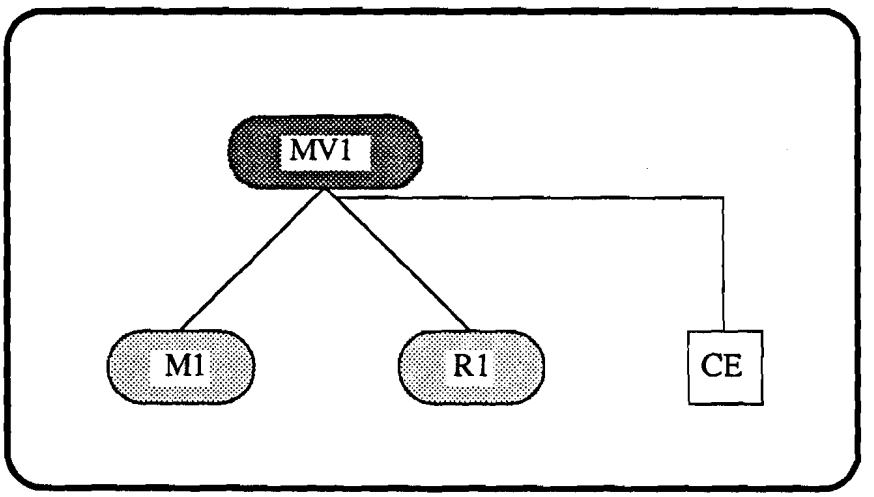



fig 17 : contrainte d'exclusion

MV1= M1 **CE** R1 <=> E(M1) $\neq$ E(R1)

M1 et R1 ne doivent pas être simultanément dans l'état de marche.

### III . 3 . 6 - Contrainte procédurale CPRO

C'est la prise en compte de protocoles de marche à respecter entre machines. La réalisation d'une action doit se faire après la réalisation d'une ou plusieurs autres. Ces contraintes sur changements d'état peuvent concerner n'importe quel état des machines. Cela peut être le cas d'une relation de précédence entre ordres de marche. Cela peut être le cas d'une synchronisation entre deux machines. L'arrêt de l'une est provoqué lorsque son stock de pièces en-cours est vide du fait du non-remplissage par la machine en amont.

**Représentation arborescente**

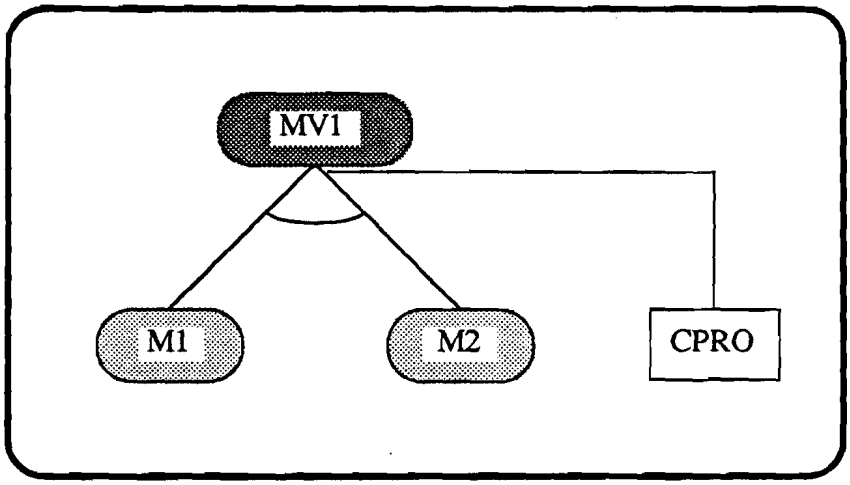



fig 18 : contrainte procédurale

$$MV1 = M1\ \mathbf{CPRO}\ R1 \Leftrightarrow E(M1) = E(M2) \text{ et } \uparrow E(M2) \neq \uparrow E(M1)$$

$\uparrow$ est un caractère signifiant l'occurrence d'un changement d'état. Cette définition signifie que les états atteints sont identiques mais atteints à des moments différents.

### III . 3 . 7 - Contrainte d'observabilité CO

C'est la possibilité ou non d'accéder aux informations nécessaires à la commande de la machine. Il faut pouvoir tenir compte du fait que si l'état d'une machine est modifié alors les états d'autres machines en relation par la contrainte CO ne seront plus accessibles pour la mise à jour d'informations.

**Représentation arborescente**

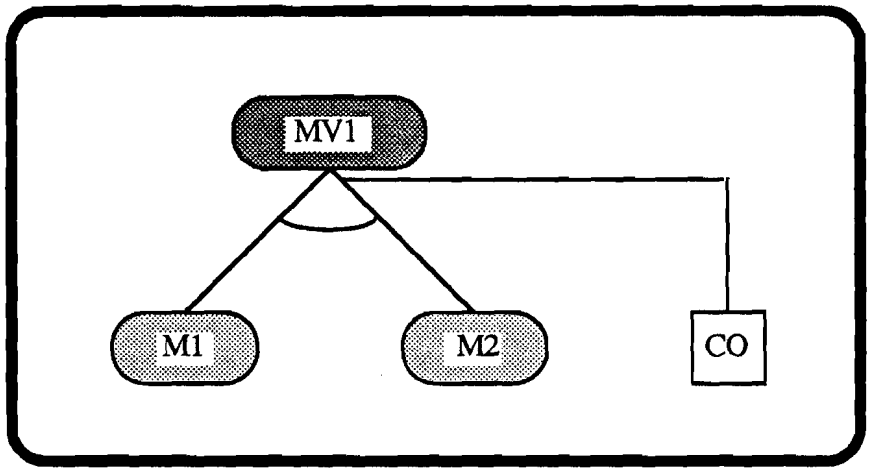



fig 19 : contrainte d'observabilité

Dans le cas de la contrainte d'observabilité, si la machine M1 qui autorise l'observabilité, est en panne, alors la machine associée R1 n'est plus exploitable, vu du gestionnaire. L'ensemble n'est donc plus exploitable.

$$MV1 = M1 \ \mathbf{CO} \ M2 \Leftrightarrow E(M1) = f(E(M2))$$

### III . 3 . 8 - Contrainte structurelle CS

Jusqu'à présent, nous nous sommes attachés à mettre en valeur les contraintes de fonctionnement entre les différentes machines d'un système de production. Cependant, il se peut qu'ils n'en existent pas assez pour obtenir un modèle suffisamment structuré pour être exploitable.

Nous pouvons alors distinguer des contraintes de type structurel, à la différence des contraintes précédentes qui étaient de type fonctionnel. Quand il n'est pas possible de dégager de contraintes fonctionnelles, nous pouvons toujours effectuer des regroupements structurels.

Deux cas sont distingués : un regroupement série ou un regroupement parallèle. Ce sont en fait des regroupements naturels qui ne mettent pas en jeu des contraintes bien explicites mais liées à la structure physique du système de production.

Nous avons distingué ces deux structures par le fait qu'elles induisent un style de production totalement différent. Une structure parallèle privilégie la flexibilité de fonctionnement en multipliant les postes de travail. Chacun d'eux est autonome et l'arrêt de l'un ne constitue pas une gêne pour les autres postes. Une structure série ou en ligne, au contraire, est pénalisée lorsque une de ses machines s'arrête, ce qui risque de bloquer l'ensemble de la chaîne.

### III . 3 . 8 . 1 - Cas d'un fonctionnement en série

Dans **le regroupement série**, le produit d'entrée d'une machine est constitué du produit de sortie d'une autre.

Par analogie, le groupement série est identique à une contrainte de type ET. La contrainte de type ET montre que deux machines liées ainsi doivent fonctionner de la même manière. Nous avons alors une contrainte structurelle série **CSS.**

Cette relation est identique à celle d'une contrainte de coopération mais nous avons distingué la situation où deux machines travaillent fonctionnellement et coopèrent ensemble, de celle où les machines ne sont liées que par des relations de produit.

### III . 3 . 8 . 2 - Cas d'un fonctionnement en parallèle

Dans **le regroupement parallèle**, deux machines effectuent le même type de travail et manipulent le même produit. Elles sont considérées comme interchangeables.

Des machines qui sont regroupées en une machine virtuelle ont des fonctionnements identiques et parallèles. Ceci n'impose pas de contrainte sur leurs modes de marche qui sont autonomes. Elles peuvent se remplacer mutuellement en cas de pannes. Leurs états respectifs sont toutefois complètement indépendants.

Par analogie, le groupement parallèle ressemble à une contrainte de type OU. La relation de type OU montre que deux machines ainsi liées peuvent fonctionner de manière autonome. Nous avons une relation structurelle parallèle **CSP.**

## IV - UTILISATION DES CONTRAINTES

De façon à pouvoir appréhender les relations entre les graphes d'état, il est nécessaire d'introduire un outil capable de les modéliser. Nous utilisons **des tables de contraintes** qui recensent l'ensemble des relations existantes.

Le programmeur, lors d'une phase de spécifications, répertorie l'ensemble des contraintes existantes dans le système de production. La table de contraintes permet de définir les différents liens entre machines en les typant (sur état ou changement d'état).

Dans la table sont répertoriés tous les liens et contraintes, caractérisant les interactions entre les machines citées en horizontal ou vertical.

**Exemple de table**

| | M1 | M2 | M3 | M4 | M5 | | Mi |
|---|---|---|---|---|---|---|---|
| **M1** | | C12 | C13 | C14 | C15 | | C1i |
| **M2** | C21 | | | | | | C2i |
| **M3** | C31 | | | | | | C3i |
| **M4** | C41 | | | | | | C4i |
| **M5** | C51 | | | | | | C5i |
| | | | | | | | |
| **Mi** | Ci1 | Ci2 | Ci2 | Ci4 | Ci5 | | |

fig 20 : exemple de table

Toutes les machines Mi sont répertoriées en horizontal et en vertical.

Les contraintes Cij sont répertoriées dans le tableau. Les machines citées en ligne sont celles qui imposent une contrainte vis à vis des autres machines citées en colonnes, dans le cas où la contrainte ne serait pas symétrique.

Nous supposons que Cij peut être une ou plusieurs contraintes sans en préciser le type.

Cij signifie que la machine i et la machine j sont mutuellement contraintes. Certaines ne sont pas commutatives, c'est à dire qu'une machine exerce une contrainte sur une autre.

### IV . 1 - Modèle structuro-fonctionnelle

L'objectif terminal de la démarche est d'aboutir à une représentation structurée du système de production sous la forme d'une arborescence. Celle-ci permet de visualiser les différents niveaux hiérarchiques et leurs liens.

**Une arborescence** est un graphe particulier pour lequel il existe un sommet appelé racine, pour lequel tout autre sommet du graphe est relié à la racine par un chemin unique.

Cette arborescence est construite à partir de spécifications citées auparavant : la table de contraintes.

## IV . 2 - Exemple d'arborescence

Nous donnons un exemple de représentation à laquelle nous devons aboutir en fin de démarche de modélisation. La représentation est déduite en tenant compte des contraintes citées auparavant.

Nous avons pris le cas d'une entreprise fortement automatisée et équipée d'un réseau local industriel, de type LAC.

Nous avons voulu montrer l'importance des communications et la manière dont nous l'intégrons dans notre représentation. Nous avons modélisé le réseau LAC car ses équipements sont pratiquement équivalents, au point de vue modes de marche, à des machines industrielles.

Le réseau LAC [COM 89] est constitué de 1 à 4 segments, un segment étant constitué d'un câble et de communicateurs. Les segments sont reliés par des modems et/ou des passerelles. Les communicateurs assure la liaison entre un abonné et le câble réseau. C'est pratiquement un micro-ordinateur, avec un système d'exploitation LACE multi-tâches, temps réel et orienté communication.

Des fonctions réseaux sont prévues, qui permettent de contrôler ces communicateurs à distance. La structure est de type maître/maître, ce qui rend les communications directes entre utilisateurs et ne nécessite pas l'utilisation d'un superviseur de réseau.

La sécurité de fonctionnement est assurée par une redondance du câble ainsi que des protocoles de communication.

Les communicateurs peuvent prendre différents états, leurs configurations étant paramétrées par des requêtes réseaux.

Ces états sont :

- un état de marche normal,
- un état d'initialisation simple, qui correspond à une mise sous-tension avec une perte des données internes,
- un état d'initialisation après téléchargement, avec conservation des valeurs.

Ce sont ces communicateurs qui nous permettent d'intégrer le réseau comme une entité commandable. De ce fait, nous pouvons associer à chaque communicateur un graphe d'état modélisant son comportement.

Nous pouvons utiliser ici le graphe d'état présenté précédemment : il peut être implanté par exemple sur l'exécutif temps réel.

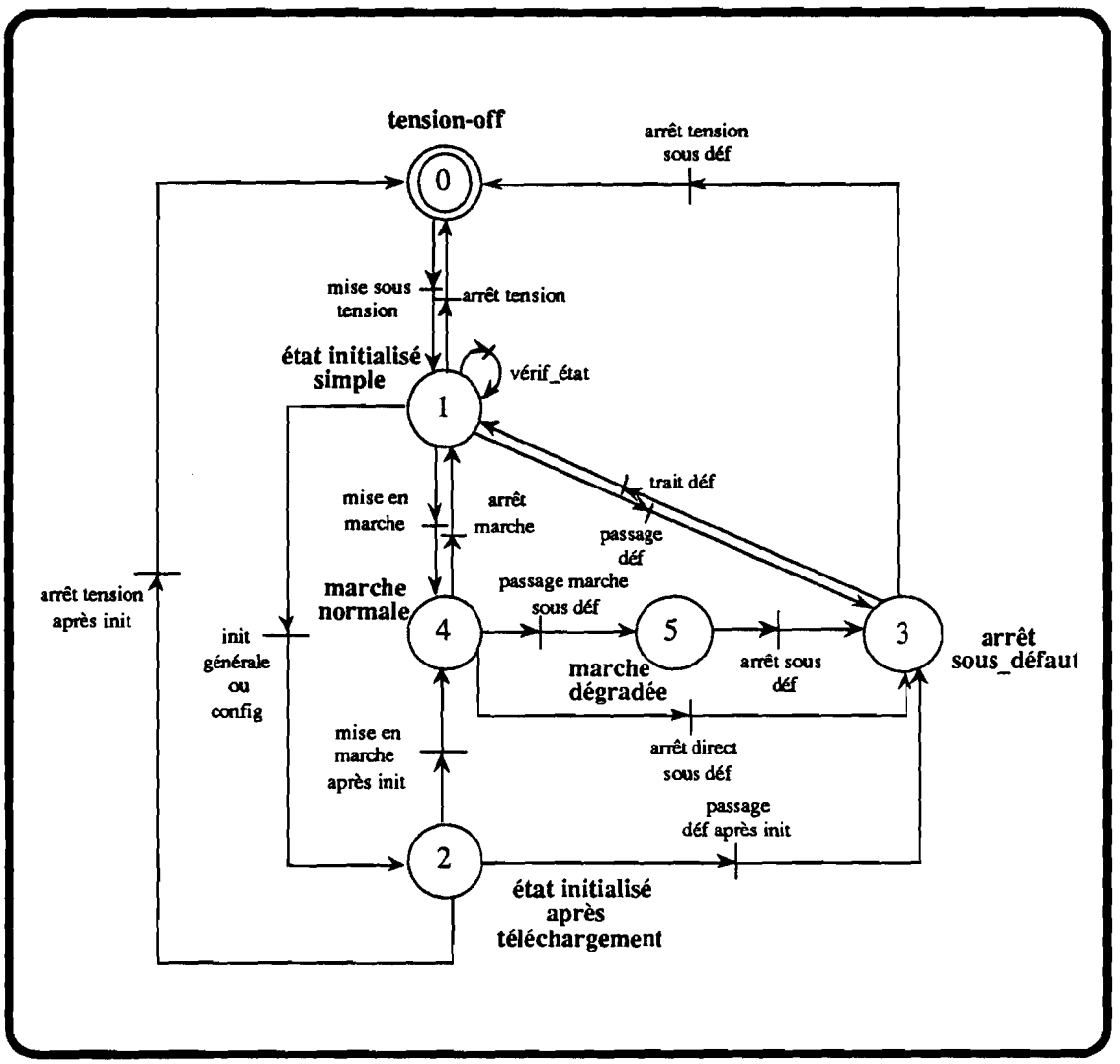



fig 21 : graphe d'état d'un communicateur

Six états sont distingués. L'état "marche normale" correspond à la communication effective. Deux états d'initialisation sont représentés. Les passages de l'état 0 aux autres états se font sur des commandes externes (requêtes réseaux).

L'état "tension-off" correspond en fait à l'absence du communicateur sur le réseau (une fonction demande d'état permet un suivi de l'installation par la détermination des communicateurs présents). L'état "tension-on" correspond en fait à la présence effective du communicateur sur le réseau.

Ce comportement est régi par un protocole spécifique LAC. Un système d'accusé/réception permet de détecter les anomalies de fonctionnement (liaison défectueuse avec l'équipement). Dans ce cas, l'ensemble de l'installation est alors prévenu de la défaillance et de sa nature.

La détection d'un défaut d'un élément, se fait en fait, lorsque il y a réception d'un acquittement négatif provenant de cet élément.

**Représentation structurée**

Nous avons pris l'exemple de l'installation intégrant un réseau LAC dont la description simplifiée est la suivante [LEP 89] :

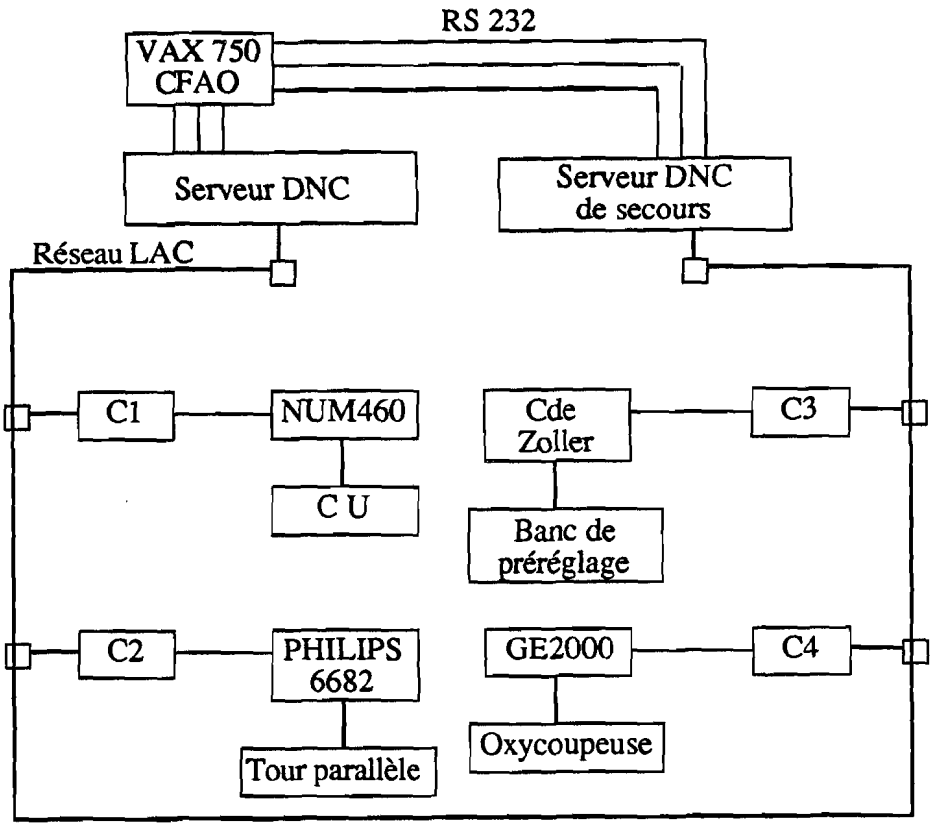



fig 22 : installation simplifiée

Plusieurs machines-outils sont connectées par un réseau LAC via les communicateurs. Un VAX 750 contrôle l'usine par l'intermédiaire d'un serveur DNC principal, remplacé en cas de panne par un autre serveur.

Chaque constituant est associé à un communicateur qui fait partie intégrante de la machine.

L'ensemble des machines-outils est géré par un VAX 750 qui prépare les fichiers de commande.

Nous obtenons la représentation structuro-fonctionnelle suivante :

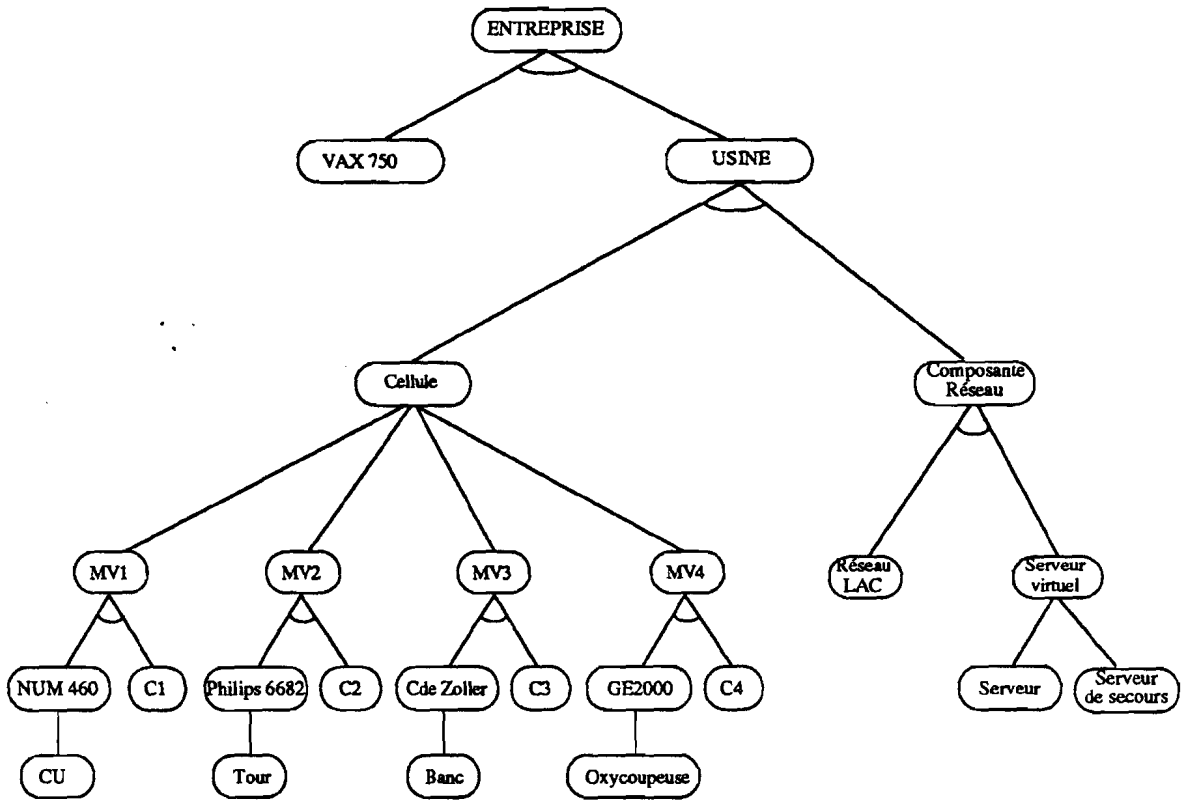



fig 23 : représentation structuro-fonctionnelle

Chaque îlot, équivalent à une machine virtuelle, est mis en évidence. C'est l'association d'une machine-outil et de son communicateur liés par une contrainte de coopération. Ces îlots constituent une machine virtuelle : Cellule. Celle-ci constitue avec la composante réseau une nouvelle machine virtuelle, Usine, qui elle même avec le VAX 750, constitue une machine virtuelle, Entreprise.

L'arborescence de racine Cellule est un arbre de type OU pour montrer le fonctionnement indépendant de chaque îlot. De même, l'arborescence de racine Serveur virtuel est un arbre de type OU pour montrer qu'il suffit qu'un des deux soit en marche.

Par contre les autres arborescences sont de type ET, en particulier celle de racine Composante Réseau. Nous pouvons remarquer l'importance de la communication. Si l'un des communicateurs tombe en panne, la machine associée est inutilisable. De même, si le réseau ou les deux serveurs sont en pannes, la cellule n'est plus observable par le VAX et donc n'est plus commandable.

Ceci est un exemple de représentation à laquelle nous devons aboutir dans notre démarche.

## IV . 3 - Décomposition d'une machine effective

Nous avons cherché à regrouper fonctionnellement, les différents classes d'objets non commandables pour chaque machine effective.

Nous avons trois composantes par machine :

- une composante **actionneurs,**
- une composante **effecteurs/capteurs,**
- une composante **communication.**

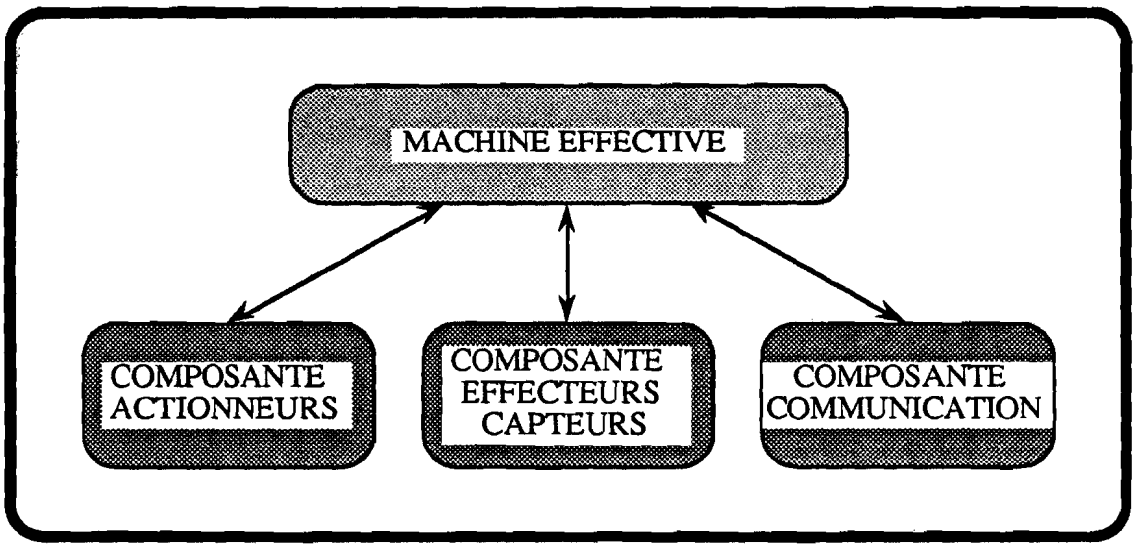



fig 24 : différentes composantes d'une machine effective

**La composante actionneurs** est constituée de l'ensemble des éléments commandables de la machine.

**La composante communication** met en évidence la nécessité posée à une machine de pouvoir communiquer avec l'extérieur. Son utilisation est obligatoire dans le cadre d'un fonctionnement en atelier. Cependant elle n'est pas nécessaire au fonctionnement autonome de la machine, contrairement aux autres composantes de la machine.

**La composante effecteurs/capteurs** est essentielle au fonctionnement de la machine. Nous pouvons particulariser la composante effecteurs de la composante capteurs d'un point de vu fonctionnel.

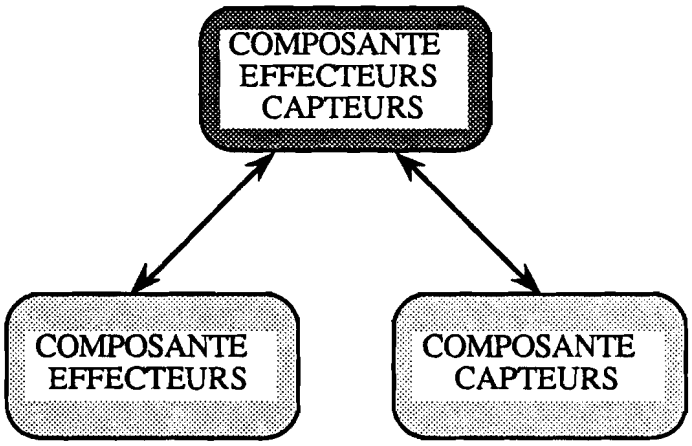



fig 25 : décomposition

Chaque sous-composante est décomposable en éléments simples.

La composante effecteurs regroupe tous les outils possibles, la composante capteurs regroupe tous les capteurs.

Emettons, à titre d'illustrations, des hypothèses de fonctionnement : supposons par exemple, qu'il suffit qu'un capteur soit en panne pour rendre la machine inutilisable. Par contre, si un outil est endommagé, la machine reste utilisable sous certaines conditions (l'outil peut être remplacé ou la machine peut utiliser un autre outil pour un usinage différent).

Au niveau description, les outils interchangeables peuvent être regroupés dans un arbre OU et les capteurs dans un arbre ET, afin de montrer leurs relations.

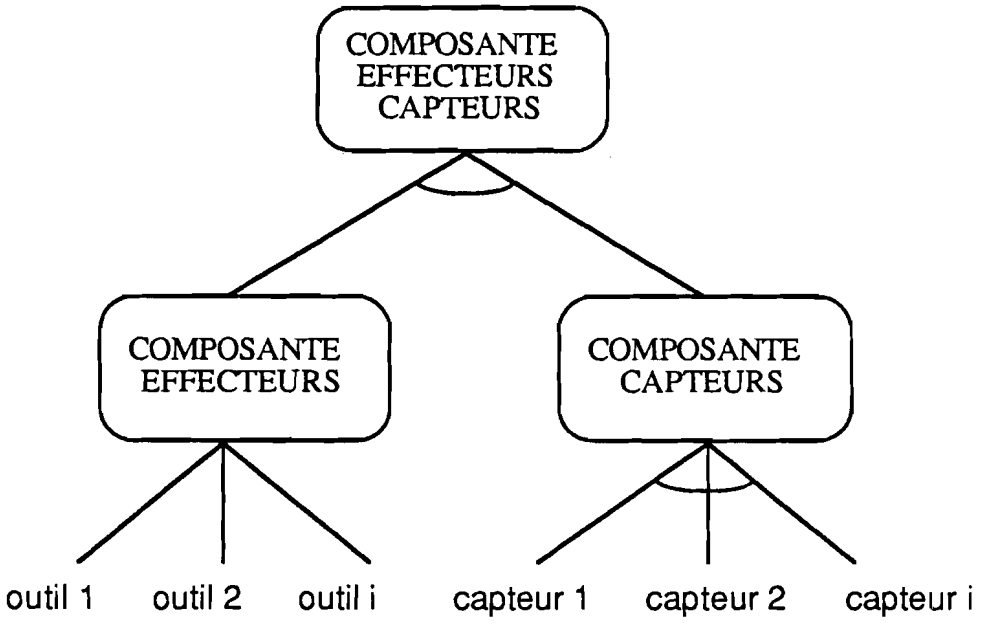



fig 26 : description précise

La relation ET montre bien que tous les capteurs doivent être dans un état de bon fonctionnement alors que la relation OU signifie que l'unique disponibilité d'un outil permet l'utilisation de la machine.

### IV . 3 . 1 - Utilisation dans la surveillance

Cette précision dans la description permet de prendre en compte tous les défauts et les dysfonctionnements possibles qui peuvent survenir et qui sont susceptibles de modifier le fonctionnement général d'une machine ou d'une cellule.

Cela permet d'introduire un premier niveau de surveillance par la propagation d'une information vers un niveau hiérarchique de commande.

Supposons, par exemple, qu'un défaut soit détecté sur un des capteurs d'une machine. Celle-ci est positionnée dans un état de repli avant que le défaut n'ait de conséquences plus importantes (par exemple, la non détection d'une pièce sur un convoyeur). De plus, le défaut peut affecter non seulement la machine supportant le capteur, mais éventuellement son environnement.

## IV . 4 - Regroupement multiple

Jusqu'à présent, nous n'avons effectué que des regroupements binaires. Les regroupements multiples sont possibles à condition que les machines regroupées aient strictement les mêmes types de relations entre elles.

Soit, par exemple, trois sous-machines M1, M2 et M3, liées entre elles par une même contrainte Ci.

| | M1 | M2 | M3 |
|---|---|---|---|
| M1 | | Ci | Ci |
| M2 | Ci | | Ci |
| M3 | Ci | Ci | |

fig 27 : table des contraintes

Nous regroupons ces trois machines en une machine virtuelle MV et nous obtenons la représentation structurée suivante.

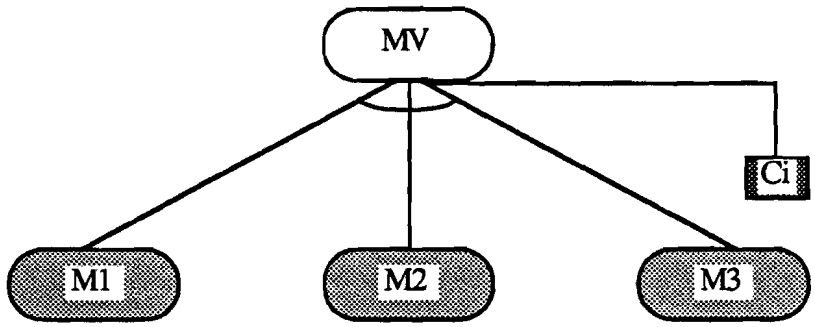



fig 28 : représentation ternaire

### IV . 5 - Phase de regroupements : règles

A partir des informations regroupées dans la table de contraintes, l'usager peut effectuer les regroupements de machines en machines virtuelles par une phase d'analyse ascendante.

Il s'agit pour ce faire, de repérer des groupements de machines partageant les mêmes types de contraintes avec d'autres machines. Le but des regroupements est de proposer des associations de machines assurant la mise en évidence des sous-systèmes indépendants.

La phase d'analyse ascendante prend en compte progressivement des contraintes entre machines pour mettre en évidence les machines virtuelles ; les machines élémentaires qui sont fortement contraintes sont, quand à elles, regroupées.

Il ne faut prendre en compte que les relations qui sont strictement nécessaires, c'est à dire celles qui n'imposent pas de choix ou d'indéterminismes afin d'assurer une résolution complètement déterministe.

Comme il est possible d'avoir des contraintes de différents types entre machines de même niveau, il est donc nécessaire d'effectuer un classement des contraintes selon une relation d'ordre afin d'effectuer des regroupements prioritaires. De plus, comme nous effectuons une démarche ascendante, nous regroupons, à relation d'ordre équivalente, en fonction du nombre de machines impliquées. Moins il y a de machines impliquées, plus la relation est prioritaire.

Lorsque les regroupements de machines effectives en machines virtuelles sont réalisés, il peut apparaître, à ce moment là, de nouvelles relations entre machines virtuelles. De nouveau, des regroupements sont effectués pour obtenir des machines virtuelles de niveau supérieur. Ces opérations sont réalisées pour aboutir à la description d'une seule machine virtuelle, ce qui marque la phase terminale.

La démarche est systématique et ne comporte pas d'indéterminisme aux niveaux des regroupements. L'introduction d'une relation d'ordre entre les contraintes permet ce caractère déterministe.

La structure obtenue n'a pas forcément tenu compte de toutes les relations existantes, lorsqu'elle en privilégie certaines au détriment d'autres. Les applications traitées ont montré que ces situations étaient rares et éliminées par le caractère transitif des contraintes.

## IV . 6 - Relation d'ordre entre contraintes

La relation retenue est déterminée par l'importance du couplage entre deux machines.

La contrainte d'observabilité est selon nous, la relation la plus importante à privilégier car il est crucial de pouvoir toujours maîtriser le comportement d'une machine.

Les relations de coopération et d'exclusion sont également très fortes, ce qui justifie leur classement, présenté respectivement à la 2ème et à la 3ème position.

La relation structurelle est la moins contraignante et n'influe pas directement sur les modes de marche mais sur la façon d'effectuer le contrôle.

Nous obtenons la relation d'ordre suivante :

- contrainte d'observabilité,
- contrainte de coopération,
- contrainte d'exclusion,
- contrainte de coopération partagée,
- contrainte procédurale,
- contrainte structurelle série,
- contrainte structurelle parallèle.

## IV . 7 - Organigramme de regroupements

Cette démarche de regroupement est facile à systématiser. Nous en présentons un algorithme de réalisation.

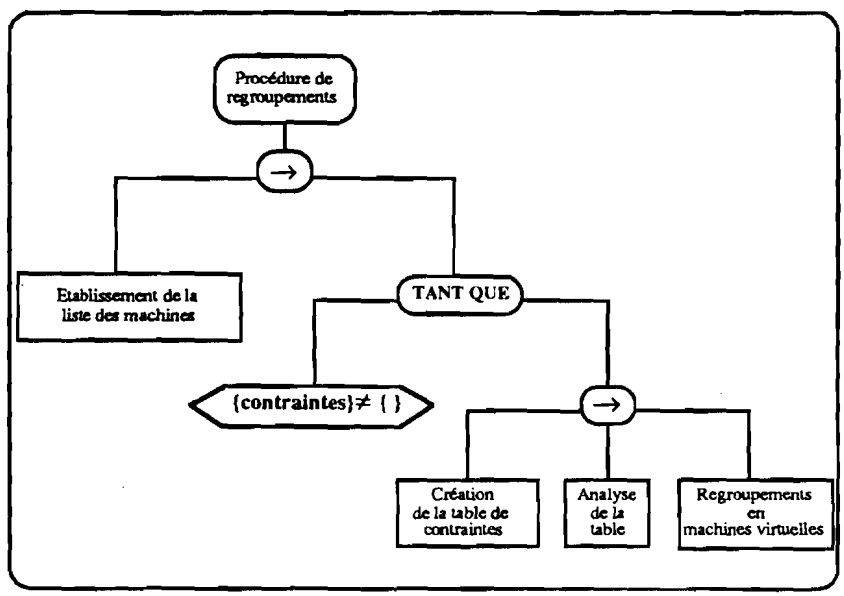



fig 29 : organigramme de regroupements

# V - CALCUL D'ÉTAT

La description obtenue nous autorise à présent, à connaître du point de vue du gestionnaire et de l'utilisateur, l'état du système global. Par une technique de calculs d'états, nous évaluons l'état des différentes machines virtuelles, jusqu'à obtenir celui du système complet.

La situation simple de calcul est le cas où l'ensemble des sous-machines d'une machine virtuelle sont toutes dans le même état, qui correspondrait à l'état de la machine virtuelle. Cependant, ce n'est pas toujours effectif et souvent les sous-machines ont des états différents, nous conduisant à établir des règles précises de calculs.

Pour pouvoir calculer l'état global d'une machine virtuelle à partir de l'état de ses sous-machines, nous avons introduit des règles de calcul basées sur des relations d'ordres entre états. La notion d'ordre et de priorité d'un état est en fait liée à l'importance des conséquences engendrées sur son environnement par le passage d'un élément dans cet état. C'est à dire que nous en considérons les états induits par ce changement

Tout d'abord, nous supposons que pour qu'une machine virtuelle se trouve dans un état, il faut que toutes ses sous-machines soient dans cet état ou tout au moins dans un état compatible moins prioritaire.

Ces règles d'évaluation sont différentes suivant les liens et les relations entre machines.

Nous nous plaçons toujours sous un angle particulier lors de ces calculs, c'est à dire au niveau du gestionnaire de modes de marche. Ceci nous amène a prendre en compte toutes les conditions possibles.

Nous ne tiendrons pas compte dans ce chapitre des relations utilisées pour la gestion effective des modes de marche, cet aspect sera développé dans le chapitre III.

## V . 1 - Cas de la coopération de deux machines

### Relation d'ordre entre états

Dans le cas d'une coopération entre machines, l'état "tension-off" est celui qui est, de toute évidence, le plus prioritaire.

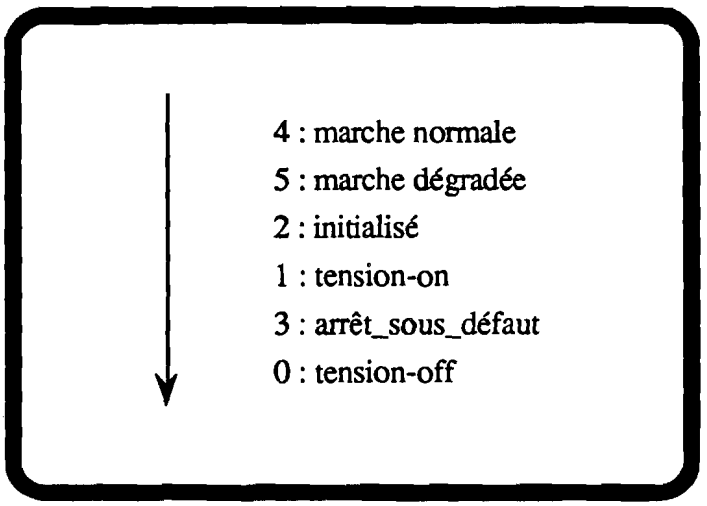



fig 30 : relation d'ordre pour une contrainte de type coopération

La seule possibilité pour qu'une machine virtuelle se trouve dans un état de marche est que l'ensemble de ses sous-machines soient toutes dans l'état marche.

L'état "tension-off" est l'état englobant, sur lequel rien ne peut agir.

Une machine en marche dégradée est un cas particulier de la marche normale avec adaptation des automatismes associés.

Une machine virtuelle sera considérée sous tension, état "tension-on", si et seulement si ses sous-machines sont dans l'état "tension-on", ou tout au moins, dans un état compatible, appartenant à la liste "marche normale", "marche dégradée" ou "initialisé".

Une machine virtuelle sera dans l'état initialisé si ses sous machines sont initialisées, donc au moins sous tension et configurées, prêtes à la marche.

L'état "arrêt_sous_défaut" est, de même, un cas particulier de l'état "tension-on" qui correspond à l'arrêt de la marche.

L'état de la machine virtuelle sera après évaluation, égal à l'état de sa sous-machine dont l'ordre sera le plus élevé dans la relation. A partir de cette relation d'ordre, nous déduisons la table d'évaluation des états.

## Table d'évaluation pour une relation de coopération

Nous avons en vertical et en horizontal, les différents états possibles des machines M1 et M2. Les différents états de la machine virtuelle constituée par M1 et M2 sont répertoriés.

| M1 \ M2 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 3 | 1 | 1 |
| 2 | 0 | 1 | 2 | 3 | 2 | 2 |
| 3 | 0 | 3 | 3 | 3 | 3 | 3 |
| 4 | 0 | 1 | 2 | 3 | 4 | 5 |
| 5 | 0 | 1 | 2 | 3 | 5 | 5 |

fig 31 : table de calculs

Nous considérons qu'il suffit qu'une machine se trouve dans l'état tension-off pour estimer l'ensemble du système dans l'état tension-off dans le cas de la coopération. Les autres sous-machines peuvent prendre soit un état de repli, soit conserver leurs états initiaux, évidemment compatibles avec l'état général du système.

Nous pouvons voir, à titre d'exemple que cette démarche est cohérente. En effet, pour qu'une machine soit considérée en marche, il faut et il suffit que l'ensemble de ses sous-machines soient dans un état "marche" dans le cas d'une contrainte de coopération.

**Exemple d'évaluation d'état :**

M1 est composée de deux sous-machines M2 et M3, liées par une contrainte de coopération. M1 $\supset$ M2 et M1 $\supset$ M3 sont les relations d'inclusion entre machine.

Si marche (M2) $\wedge$ marche (M3) alors M1 est considérée en état "marche".

Si marche (M1) $\wedge$ marche (M2) $\wedge$ tension-off (M3) alors M1 passe en tension-off du fait de la relation d'ordre

## V . 2 - Cas de l'exclusion

**Relation d'ordre entre états**

Elle est inverse à celle concernant la coopération.

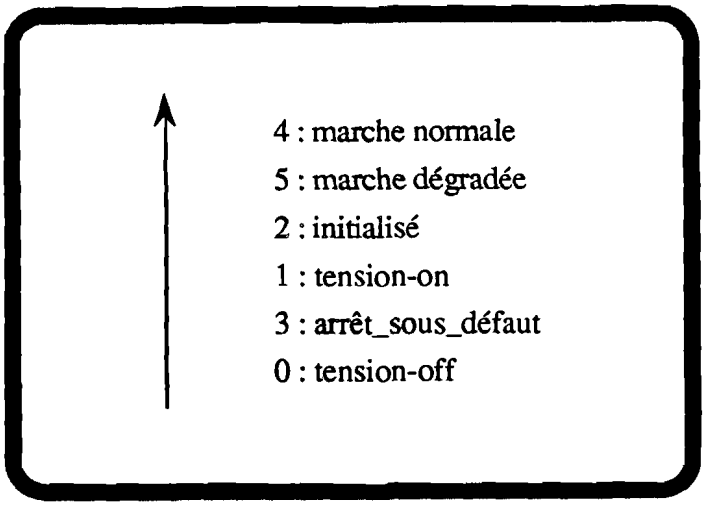



fig 32 : relation d'ordre pour une contrainte d'exclusion

L'état de la machine virtuelle sera, après calcul, égal à l'état de sa sous-machine dont l'ordre sera le plus bas dans la relation. A partir de cette relation d'ordre, nous en déduisons une table d'évaluation.

## Table d'évaluation pour une relation d'exclusion

Nous avons en vertical et en horizontal, les différents états possibles des machines M1 et M2.

| M1 \ M2 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 1 | 2 | 1 | 4 | 5 |
| 2 | 2 | 2 | 2 | 2 | 4 | 5 |
| 3 | 3 | 1 | 2 | 3 | 4 | 5 |
| 4 | 4 | 4 | 4 | 4 | | |
| 5 | 5 | 5 | 5 | 5 | | |

fig 33 : table de calculs

4 situations présentent des incompatibilités si les deux sous-machines sont simultanément en marche ou en marche dégradée. Cela est totalement en contradiction avec la définition de la contrainte associée. C'est pourquoi nous excluons ces cas.

**Exemple de calcul :**

M1 est composée de deux sous-machines M2 et M3, liées par une contrainte d'exclusion. M1 ⊃ M2 et M1 ⊃ M3 sont les relations d'inclusion entre machines

Si marche (M1) ∧ marche (M2) ∧ tension-off (M3) alors M1 conserve son état "marche".

Si marche (M1) ∧ tension-off (M2) ∧ tension-off (M3) alors M1 prend l'état tension-off.

### V . 3 - Cas d'un regroupement de machines en série

La relation d'ordre est identique à celle concernant le cas de la contrainte de coopération. En effet, les fonctionnements sont fortement liés. Si l'une des machines s'arrête, cela contraint les autres machines en série, à modifier leurs comportements.

### V . 4 - Cas d'un regroupement de machines parallèles

La relation d'ordre est identique à celle concernant le cas de la contrainte d'exclusion. L'état de la machine virtuelle sera, après calcul, égal à l'état de sa sous-machine dont l'ordre sera le plus bas dans la relation. A partir de cette relation d'ordre, nous en déduisons une table de calcul. Cependant, il n'y a pas ici de problème d'incompatibilité.

| M1 \ M2 | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 1 | 1 | 2 | 1 | 4 | 5 |
| 2 | 2 | 2 | 2 | 2 | 4 | 5 |
| 3 | 3 | 1 | 2 | 3 | 4 | 5 |
| 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 | 5 | 4 | 5 |

fig 34 : table de calcul pour deux machines en parallèle

### V . 5 - Cas de la coopération partagée

Dans le cas de machines liées par une contrainte de coopération partagée, nous reprenons les résultats précédents en considérant que cette contrainte induit des calculs d'état identiques à ceux d'une contrainte structurelle de type parallèle. En effet, bien que ces relations soient

fonctionnellement différentes, elles privilégient le même type de fonctionnement (les machines sont totalement autonomes les unes des autres).

### V . 6 - Cas des contraintes sur états

Nous avons considéré que les règles de calculs d'état pour une machine virtuelle constituée de machines liées par une contrainte sur état, contrainte d'observabilité ou procédurale, sont identiques à celles spécifiques à une relation de coopération.

En effet, les calculs d'état se font sur les états atteints, ce qui signifie que nous ne tenons pas compte de la manière d'accés à ces états.

### V . 7 - Exemple de calcul d'état

Une cellule flexible est composée de 4 machines : M1, M2, M3 et M4, associées deux par deux, étant respectivement liées par une contrainte de coopération, CC1 et CC2. Deux machines virtuelles MV1 et MV2 sont donc constituées. Celles-ci sont liées par une contrainte de coopération CC3 et sont agrégées en une machine virtuelle MV3.

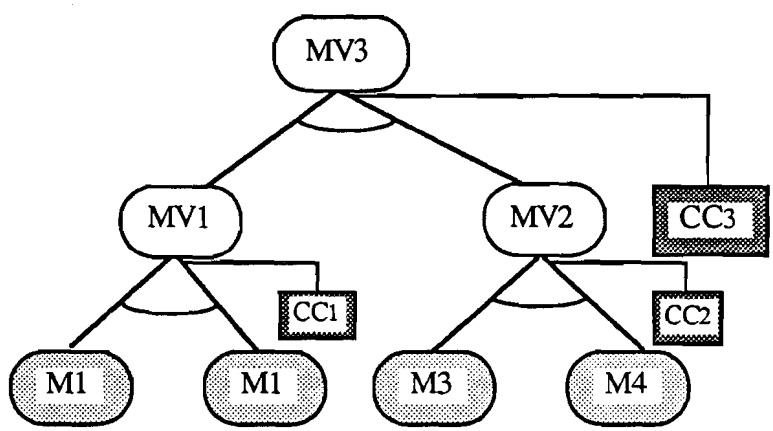



fig 35 : représentation structurée de l'exemple

Sur la 1ère figure, nous avons représenté le graphe d'état de chaque machine. Nous n'avons pas considéré les états transitoires.

Toutes les machines se trouvent au départ, dans l'état "marche".

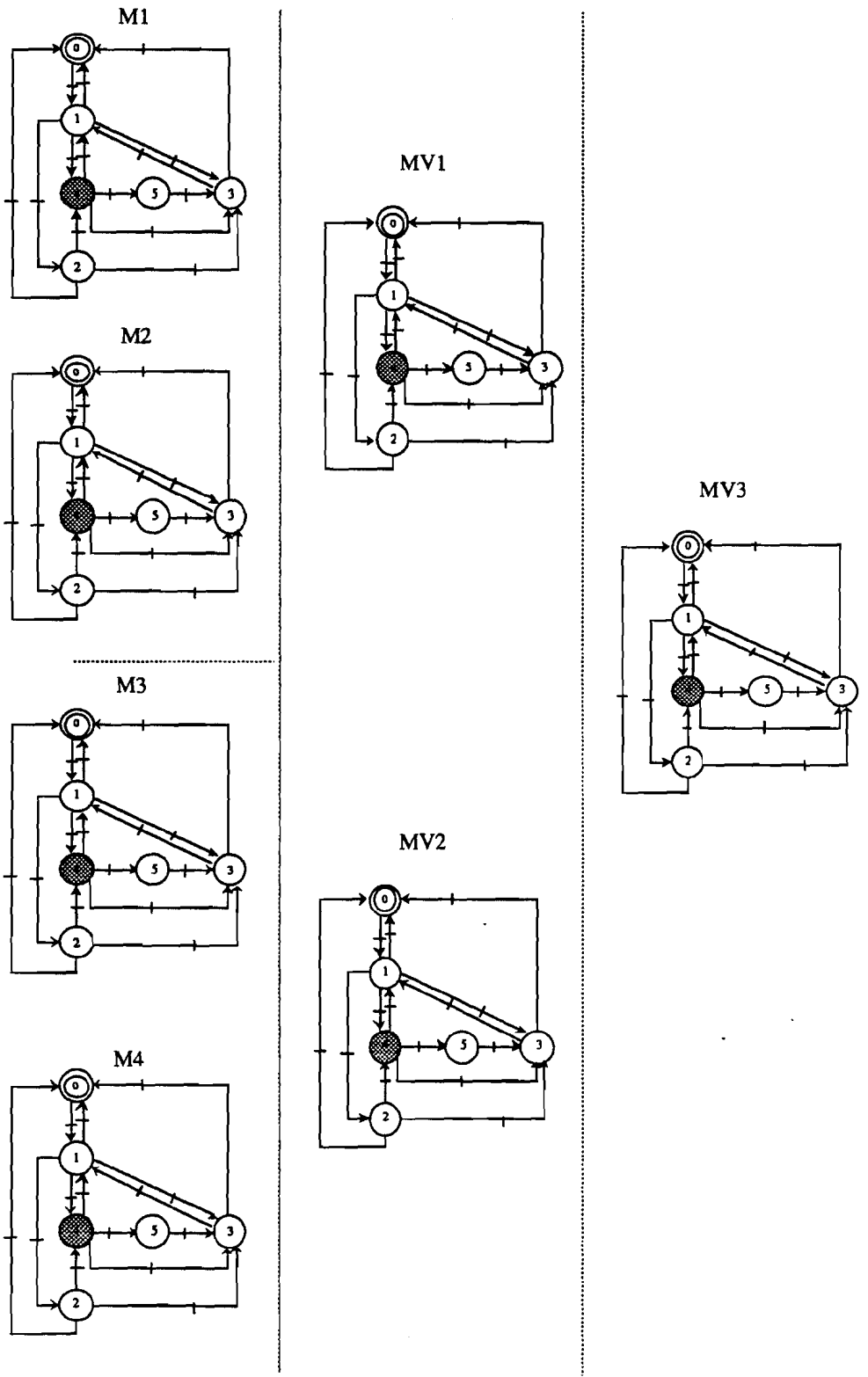



fig 36 : marche normale

Sur la 2ème figure, une machine, M1 change d'état et passe dans l'état "marche_dégradée". Par conséquent, l'état de MV1 change d'état et passe dans le même état " marche_dégradée". De même, M3 change d'état et passe dans l'état "tension-on" d'où le passage de MV2 dans le même état. Enfin, le calcul d'état de MV3 la considère dans l'état

"tension-on". Les autres machines qui ne changent pas directement d'état par calcul, demeurent dans le même état.

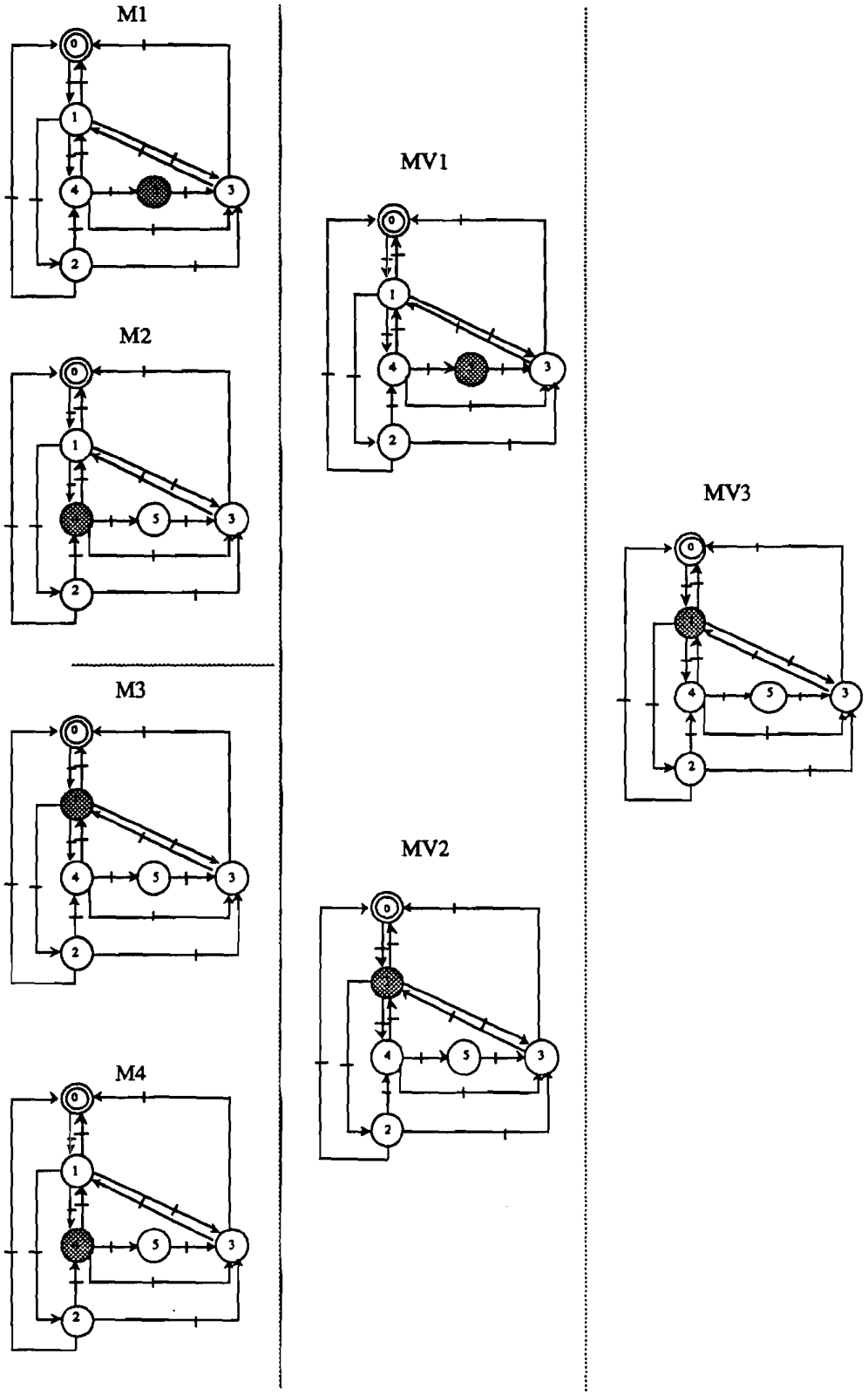



fig 37 : après changement de modes de marche

Cet exemple permet de montrer les imbrications et les répercussions qu'occasionne un changement d'état sur le reste du système de production.

## VI - EXEMPLE D'UNE DECOMPOSITION

Nous étudions l'atelier flexible de l'EC Lille qui sera décrit plus précisément au chapitre III et qui est constituée :

- d'un tour et de son robot de chargement/déchargement : TOUR et R1,
- d'un centre d'usinage et de son robot de chargement/déchargement : FRAISEUSE et R2,
- d'une station d'assemblage et de son robot d'assemblage : ASSE et R3,
- d'un stockeur et de son robot portique : STO et R4,
- d'un convoyeur A qui alimente le tour et le centre d'usinage : CONV1,
- d'un convoyeur B chargé d'alimenter la station d'assemblage et le stockeur : CONV2,
- de deux automates programmables chargés de la coordination et de la synchronisation des machines de l'atelier : LPC1 et LPC2,
- de deux réseaux locaux industriels, Telway et Unitelway. Le réseau Telway NET1 fait communiquer les deux automates TSX et le réseau Unitelway NET2 permet au système de pilotage de communiquer avec tous les automates et les NUM.

Pour simplifier la description, nous n'avons pas représenté l'atelier de façon détaillée à ce niveau. Nous supposons donc que chaque machine (Tour, Fraiseuse) est en fait l'association d'une machine-outil et de son automate de commande. De plus, nous n'avons pas intégré d'éléments non commandables existants tels que des outils ou des coupleurs de communication.

Enfin, nous n'avons pas intégré l'organe de pilotage, en l'occurrence une station de travail. En effet, nous ne pouvons pas le considérer comme une machine commandable, étant donné que c'est elle qui supporte le gestionnaire de tâches.

## VI . 1 - Table initiale des contraintes

Nous représentons les contraintes existantes dans la table suivante :

| | R1 | TOUR | R2 | FRAISE | CONV1 | LPC1 | R3 | ASSE | STO | R4 | CONV2 | LPC2 | NET1 | NET2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | | CC1 | | | CC5 | CO1 | | | | | | | | CO3 |
| TOUR | CC1 | | | | CC5 | CO1 | | | | | | | | CO3 |
| R2 | | | | CC2 | CC5 | CO1 | | | | | | | | CO3 |
| FRAISEUSE | | | CC2 | | CC5 | CO1 | | | | | | | | CO3 |
| CONV1 | CC5 | CC5 | CC5 | CC5 | | CO1 | | | | | | | | CO3 |
| LPC1 | | | | | | | | | | | | CCP3 | CC7 | CO3 |
| R3 | | | | | | | | CC3 | | | CC6 | CO2 | | CO3 |
| ASSE | | | | | | | CC3 | | | | CC6 | CO2 | | CO3 |
| STO | | | | | | | | | | CC4 | CC6 | CO2 | | CO3 |
| R4 | | | | | | | | | CC4 | | CC6 | CO2 | | CO3 |
| CONV2 | | | | | | | CC6 | CC6 | CC6 | CC6 | | CO2 | | CO3 |
| LPC2 | | | | | | CCP3 | | | | | | | CC7 | CO3 |
| NET1 | | | | | | CC7 | | | | | | CC7 | | CO3 |
| NET2 | | | | | | | | | | | | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération
CCP : contrainte de coopération partagée

fig 38 : table initiale des contraintes

**Lecture de la table et explications** : le tour et le robot R1, la fraiseuse et le robot R2, la station d'assemblage et le robot R3, le stockeur et le robot R4, sont des machines fortement contraintes deux à deux et sont donc liées respectivement par une contrainte de coopération CC1, CC2, CC3 et CC4. En effet, ces opérateurs sont directement associés. Par exemple, R1 ne peut charger ou décharger que le tour. Cette contrainte étant réciproque, la relation est indiquée deux fois dans le tableau.

L'automate LPC1 exerce une contrainte d'observabilité CO1 vis à vis des machines CONV1, R1, R2, TOUR et FRAISEUSE. De même, l'automate LPC2 exerce une contrainte d'observabilité CO2 vis à vis des machines R3, R4, STO et ASSE. En effet, les automates sont les organes de coordination de ces machines et servent de relais de commande du niveau

supérieur. Ces contraintes n'étant pas réciproques, elles ne sont donc pas commutatives. La matrice est alors non symétrique (les colonnes sont alors prépondérantes aux lignes).

Le convoyeur CONV1 est lié aux machines qu'il alimente par une contrainte de coopération CC5. De même, le convoyeur CONV2 est lié aux machines qu'il alimente par une contrainte de coopération CC6.

Nous considérons que le réseau NET1 est une ressource de communication utilisée en coopération partagée par les deux automates LPC1 et LPC2.

NET2 a une relation d'observabilité vis vis de toutes les autres machines car s'il est arrêté, alors les machines ne sont plus accessibles par le superviseur de niveau haut.

Nous effectuons alors les regroupements : différentes étapes sont présentées jusqu'à l'obtention d'une seule machine virtuelle, ATELIER.

## VI . 2 - Agrégation du 1er niveau

| | R1 | TOUR | R2 | FRAISE | CONV1 | LPC1 | R3 | ASSE | STO | R4 | CONV2 | LPC2 | NET1 | NET2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | | CC1 | | | CC5 | CO1 | | | | | | | | CO3 |
| TOUR | CC1 | | | | CC5 | CO1 | | | | | | | | CO3 |
| R2 | | | | CC2 | CC5 | CO1 | | | | | | | | CO3 |
| fraiseuse | | | CC2 | | CC5 | CO1 | | | | | | | | CO3 |
| CONV1 | CC5 | CC5 | CC5 | CC5 | | CO1 | | | | | | | | CO3 |
| LPC1 | | | | | | | | | | | | CCP3 | CC7 | CO3 |
| R3 | | | | | | | | CC3 | | | CC6 | CO2 | | CO3 |
| ASSE | | | | | | | CC3 | | | | CC6 | CO2 | | CO3 |
| STO | | | | | | | | | | CC4 | CC6 | CO2 | | CO3 |
| R4 | | | | | | | | | CC4 | | CC6 | CO2 | | CO3 |
| CONV2 | | | | | | | CC6 | CC6 | CC6 | CC6 | | CO2 | | CO3 |
| LPC2 | | | | | | CCP3 | | | | | | | CC7 | CO3 |
| NET1 | | | | | | CC7 | | | | | | CC7 | | CO3 |
| NET2 | | | | | | | | | | | | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération
CCP : contrainte de coopération partagée

fig 39 : niveau 1 d'agrégation

En tenant compte des relations d'ordre entre contraintes, nous pouvons distinguer 4 machines virtuelles M1, M2, M3 et M4 composées respectivement de R1 et du TOUR, R2 et de la FRAISEUSE, de R3 et de ASSE, de R4 et de STO. Ces couples de machines sont formés car mutuellement liées entre elles, et offrent les mêmes relations vis à vis des autres machines.

Un premier niveau d'agrégation est alors effectué et une nouvelle représentation est alors proposée. De nouvelles contraintes sont alors mises en évidence : CCP1 et CCP2. M1 et M2, M3 et M4 sont liées par une contrainte de coopération partagée, respectivement CCP1 et CCP2 vis à vis de l'utilisation des ressources CONV1 et CONV2.

Nous montrons par ailleurs, comment se construit progressivement l'arborescence au fur et à mesure des agrégations et comment apparaissent les différents niveaux de décomposition.

Le niveau bas ou niveau 0 est constitué des machines effectives. Le niveau 1 est constitué des îlots de fabrication.

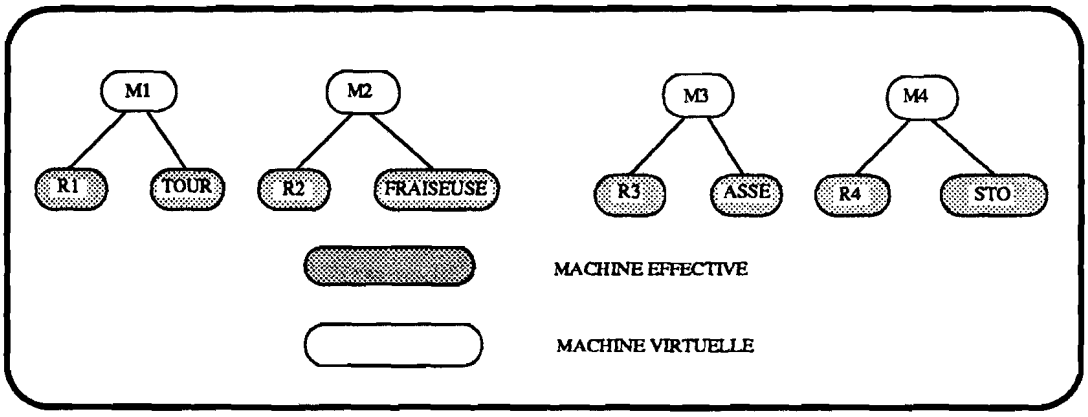



fig 40 : représentation structurée

## VI . 3 - Agrégation du 2ème niveau

| | M1 | M2 | CONV1 | LPC1 | M3 | M4 | CONV2 | LPC2 | NET1 | NET2 |
|---|---|---|---|---|---|---|---|---|---|---|
| M1 | | CCP1 | CC5 | CO1 | | | | | | CO3 |
| M2 | CCP1 | | CC5 | CO1 | | | | | | CO3 |
| CONV1 | CC5 | CC5 | | CO1 | | | | | | CO3 |
| LPC1 | | | | | | | | CCP3 | CC7 | CO3 |
| M3 | | | | | | CCP2 | CC6 | CO2 | | CO3 |
| M4 | | | | | CCP2 | | CC6 | CO2 | | CO3 |
| CONV2 | | | | | CC6 | CC6 | | CO2 | | CO3 |
| LPC2 | | | | CCP3 | | | | | CC7 | CO3 |
| NET1 | | | | CC7 | | | | CC7 | | CO3 |
| NET2 | | | | | | | | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération
CCP : contrainte de coopération partagée

fig 41 : niveau 2 d'agrégation

Deux machines virtuelles MV1 et MV2 sont créées, par l'intégration des contraintes CCP1 et CCP2. L'arborescence équivalente devient donc :

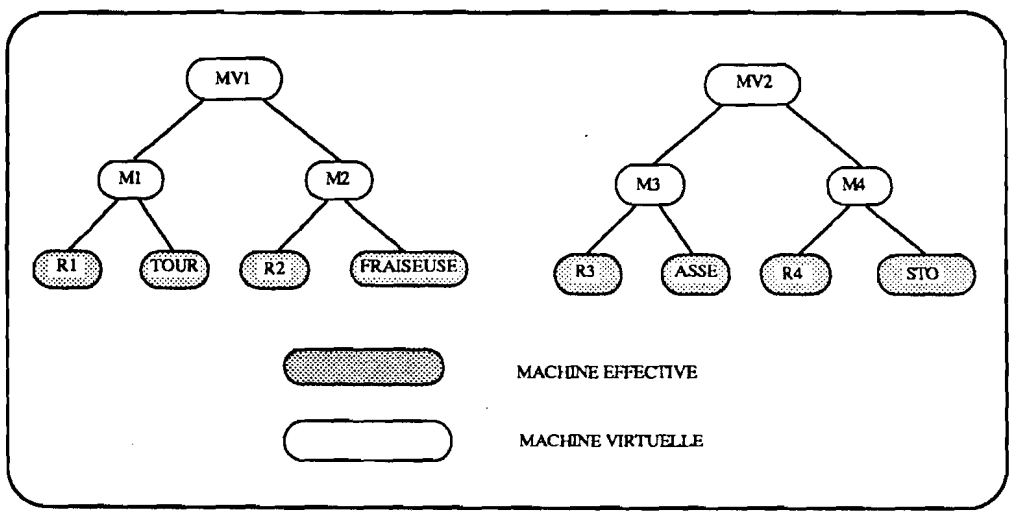



fig 42 : représentation structurée

## VI . 4 - Agrégation du 3ème niveau

| | MV1 | CONV1 | LPC1 | MV2 | CONV2 | LPC2 | NET1 | NET2 |
|---|---|---|---|---|---|---|---|---|
| MV1 | | CC5 | CO1 | | | | | CO3 |
| CONV1 | CC5 | | CO1 | | | | | CO3 |
| LPC1 | | | | | | CCP3 | CC7 | CO3 |
| MV2 | | | | | CC6 | CO2 | | CO3 |
| CONV2 | | | | CC6 | | CO2 | | CO3 |
| LPC2 | | | CCP3 | | | | CC7 | CO3 |
| NET1 | | | CC7 | | | CC7 | | CO3 |
| NET2 | | | | | | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération
CCP : contrainte de coopération partagée

fig 43 : niveau 3 d'agrégation

MV1 et CONV1, MV2 et CONV2 sont regroupées car liées par une contrainte de coopération. Elles ont de plus les mêmes relations vis à vis des autres machines. Deux machines virtuelles sont créées, ILOT1 et ILOT2. L'arborescence équivalente devient donc :

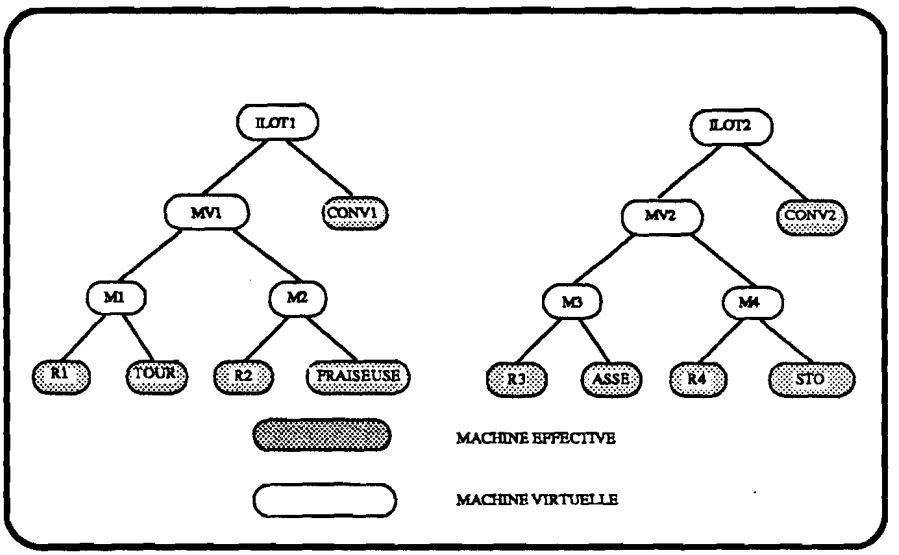



fig 44 : représentation structurée

## VI . 5 - Agrégation du 4ème niveau

| | ILOT1 | LPC1 | ILOT2 | LPC2 | NET1 | NET2 |
|---|---|---|---|---|---|---|
| ILOT1 | | CO1 | | | | CO3 |
| LPC1 | | | | CCP3 | CC7 | CO3 |
| ILOT2 | | | | CO2 | | CO3 |
| LPC2 | | CCP3 | | | CC7 | CO3 |
| NET1 | | CC7 | | CC7 | | CO3 |
| NET2 | | | | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération
CCP : contrainte de coopération partagée

fig 45 : niveau 4 d'agrégation

ILOT1 et LPC1, ILOT2 et LPC2 sont liées par une contrainte d'observabilité CO1 et CO2, et sont donc regroupées.

Deux machines virtuelles CELL1 et CELL2 sont créées. L'arborescence équivalente devient donc :

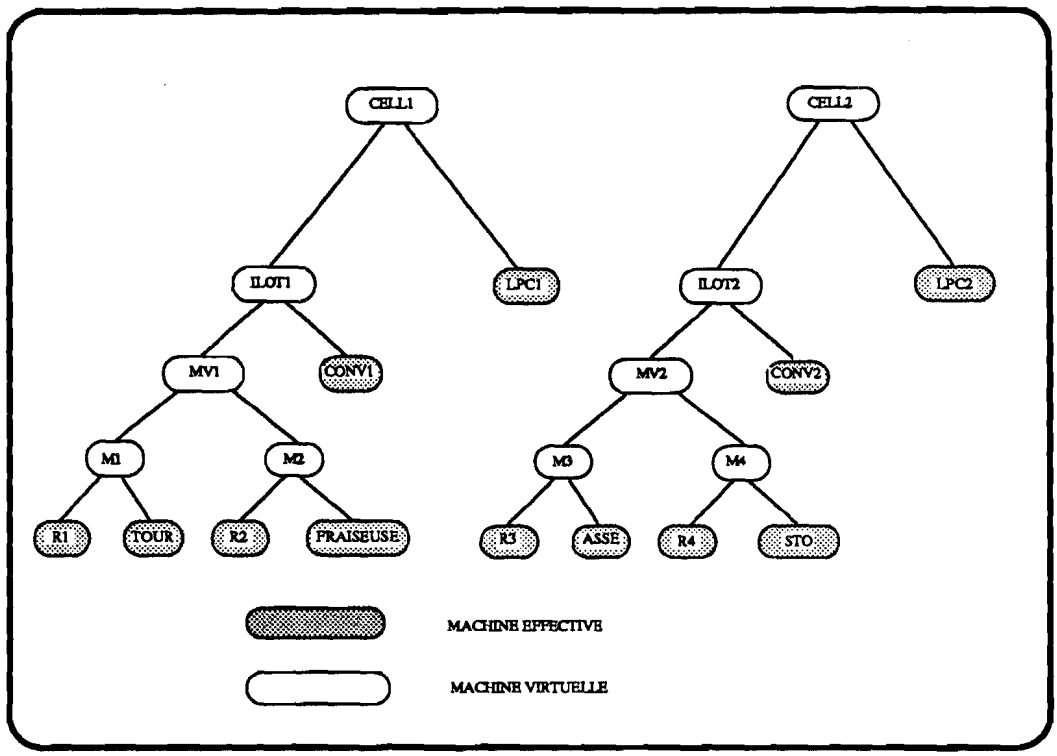



fig 46 : représentation structurée

## VI . 6 - Agrégation de 5ème niveau

| | CELL1 | CELL2 | NET1 | NET2 |
|---|---|---|---|---|
| CELL1 | | CCP3 | CC7 | CO3 |
| CELL2 | CCP3 | | CC7 | CO3 |
| NET1 | CC7 | CC7 | | CO3 |
| NET2 | | | | |

CC : contrainte de coopération
CO : contrainte d'observabilité
CCP : contrainte de coopération partagée

fig 47 : niveau 5 d'agrégation

CELL1 et CELL2 sont liées par une contrainte de coopération partagée CCP3. Elles sont regroupées en une machine virtuelle CELLULE. L'arborescence équivalente devient donc :

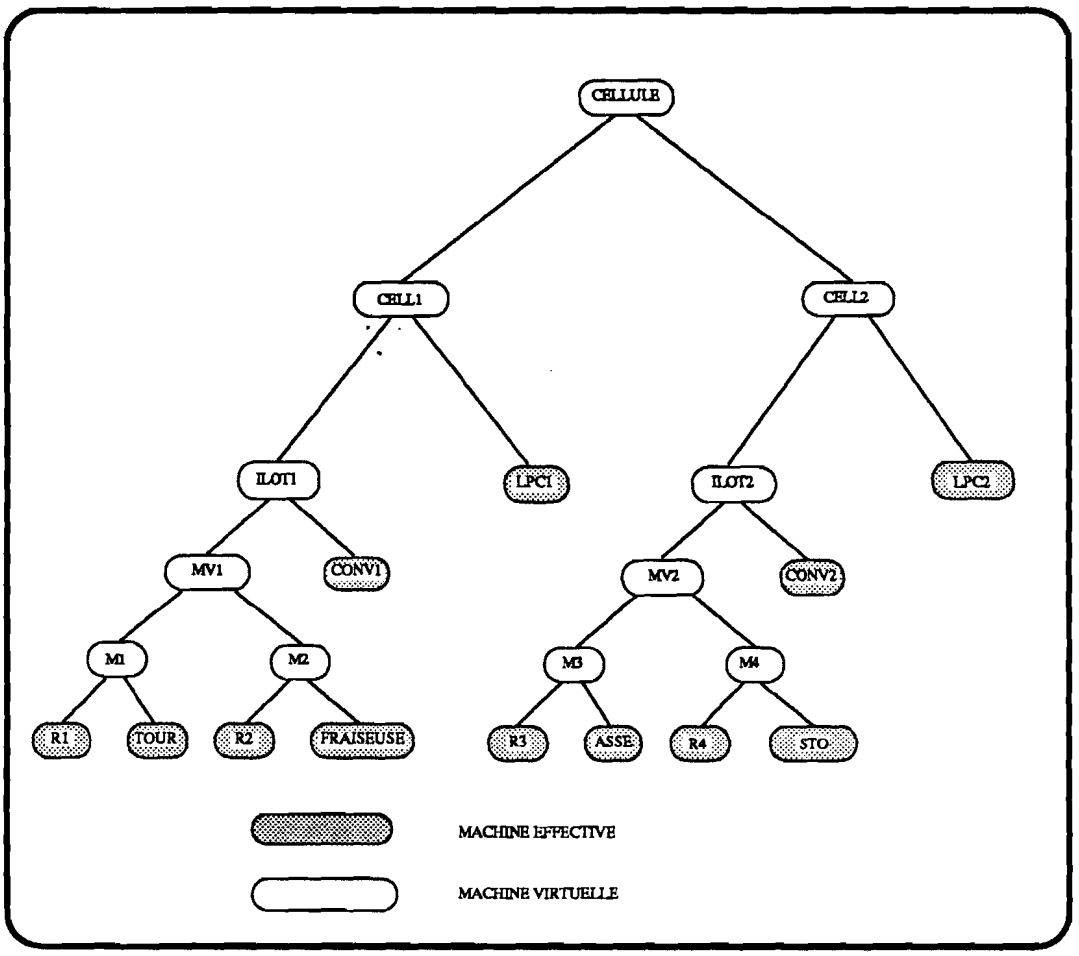



fig 48 : représentation structurée

## VI . 7 - Agrégation de 6ème niveau

| | CELLULE | NET1 | NET2 |
|---|---|---|---|
| CELLULE | | CC7 | CO3 |
| NET1 | CC7 | | CO3 |
| NET2 | | | |

CO : contrainte d'observabilité
CC : contrainte de coopération

fig 49 : niveau 6 d'agrégation

CELLULE a une contrainte de coopération CC7 avec NET1 et sont donc regroupées en la machine virtuelle CELL. L'arborescence équivalente devient donc :

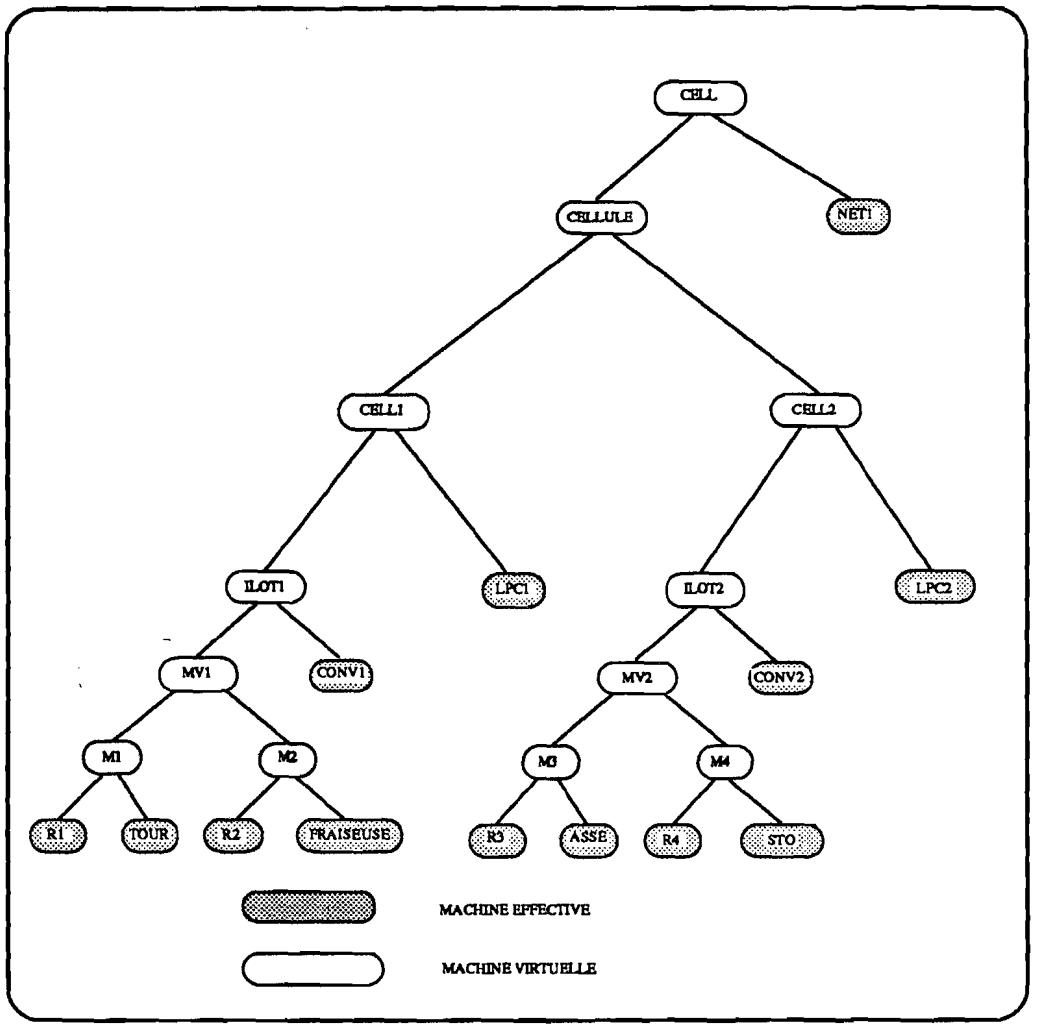



fig 50 : représentation structurée

## VI . 8 - Agrégation de 7ème niveau

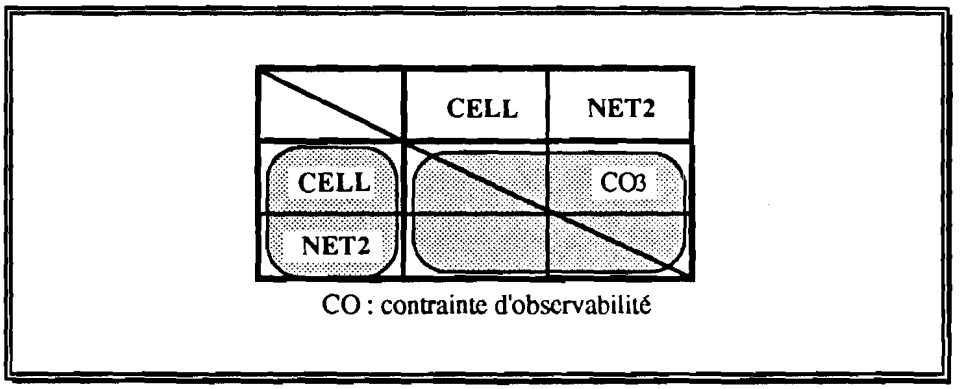



fig 51 : niveau 7 d'agrégation

CELL a une contrainte d'observabilité CO3 vis à vis de NET2 et sont donc regroupées en une machine virtuelle ATELIER.

L'atelier est complètement modélisé. Nous ne voyons de lui qu'une seule machine virtuelle, ATELIER.

La démarche ascendante est donc terminée.

### VI .9 - Représentation structurée

A partir des regroupements, nous obtenons selon le point de vue abordé ici, la structure de l'atelier représentable sous forme arborescente.

Nous pouvons voir aisément les différents niveaux structurels de l'atelier.

Nous n'intégrons pas, pour l'instant, les différentes contraintes.

Cette représentation (figure 52) permet de dégager les différents niveaux de commande de l'atelier, en particulier des sous-systèmes de type îlot.

Les composantes réseaux ont un niveau haut dans la représentation, ce qui met en évidence le caractère critique de la communication dans un système de production automatisé.

Les machines virtuelles CELL1 et CELL2 peuvent être considérées comme des cellules de fabrication supervisées par un système de pilotage situé au niveau ATELIER.

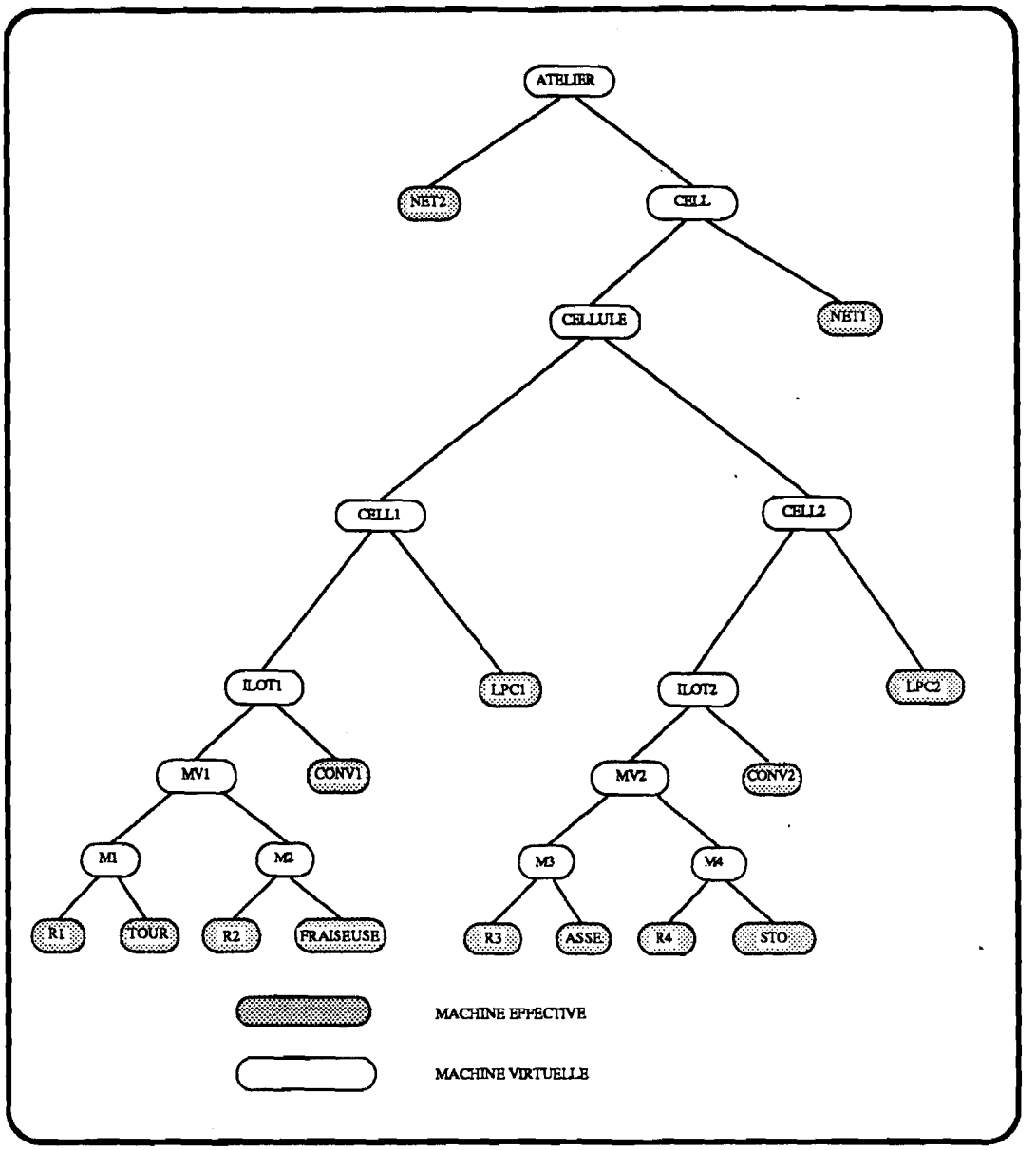



fig 52 : représentation structurelle de l'atelier

## VI . 10 - Représentation complète

Nous intégrons à ce niveau les différentes contraintes entre machines.

A partir de l'arborescence précédente, nous obtenons directement l'arbre complet qui intègre toutes les relations présentées auparavant : il suffit d'associer à chaque noeud la contrainte qui a permis l'association et de spécifier le type de l'association (noeud OU/ET). Cet arbre est de type ET/OU, les noeuds OU traduisant la flexibilité potentielle.

Nous pouvons remarquer que les feuilles correspondent en fait aux machines effectives présentes dans l'atelier, les machines virtuelles apparaissent sur les noeuds.

Cet arbre se comprend de la manière suivante : Cell1 et Cell2 sont deux machines virtuelles, de même niveau, liées par une contrainte de coopération partagée, relatives à l'utilisation du réseau.

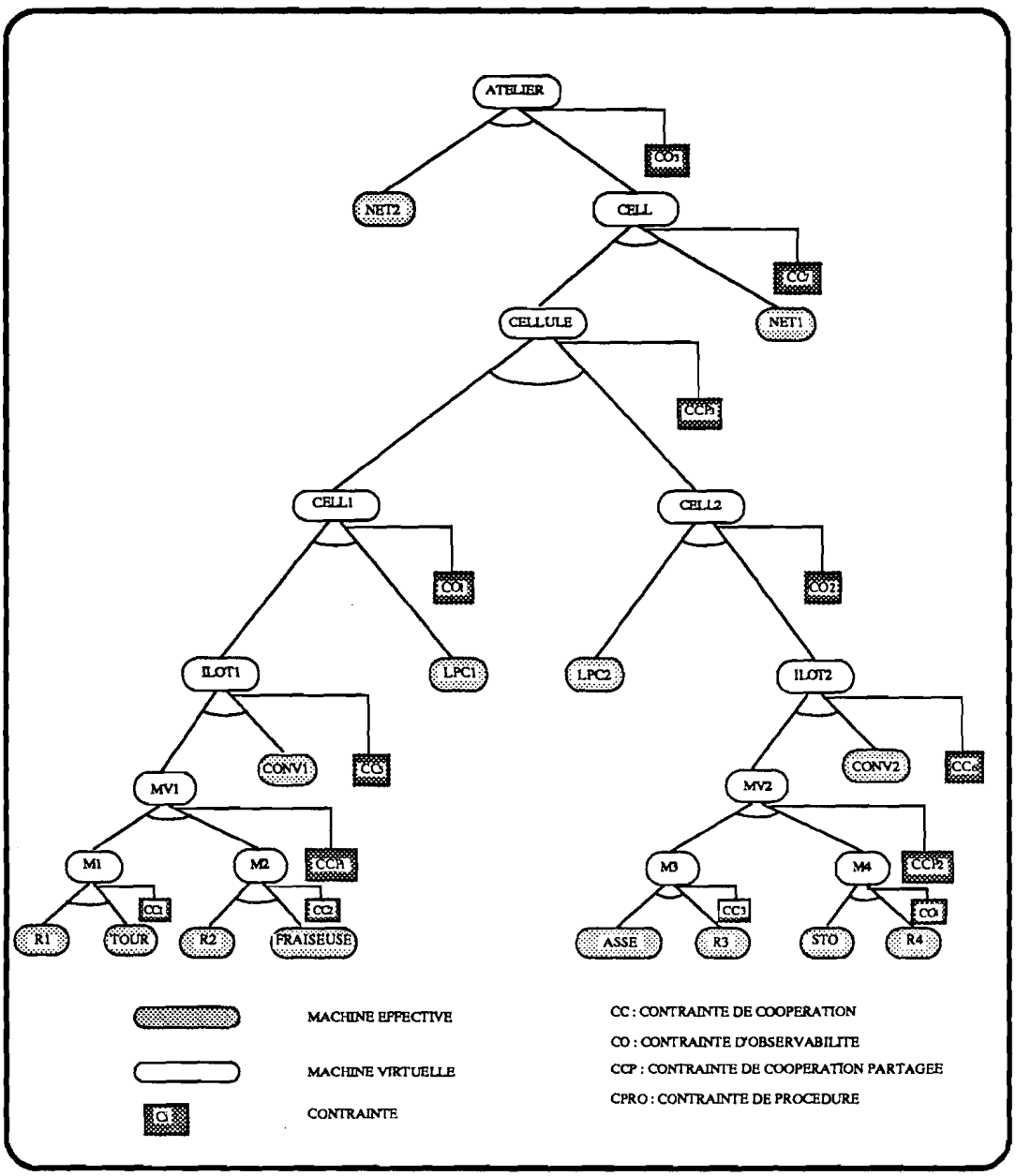



fig 53 : arbre structuré complet de l'atelier

Cet arbre va servir de modèle de référence pour la base de connaissance. Il est obtenu à partir de la représentation arborescente et des tables de contraintes. Il est alors directement exploitable pour la génération automatique de règles.

## CONCLUSION

Au cours de ce chapitre, nous avons proposé une démarche rationnelle permettant par une structuration appliquée aux machines effectives et virtuelles d'un système de production d'aboutir à une méthodologie efficace de raisonnement sur la logique de dépendance ou d'enchaînement des modes de marche. L'apport essentiel de ce travail résulte de l'identité du schéma structurel de décomposition du process et du schéma arborescent de recherche déductive des modes de marche des différentes entités composant le procédé.

Par une démarche ascendante, nous obtenons ainsi une décomposition fonctionnelle et structurelle de l'atelier, qui met en évidence les différents niveaux de regroupements, permettant d'avoir ainsi une vision structurée et hiérarchisée de l'atelier. Cette décomposition en sous-machines permettra de plus, de structurer la base de connaissance. Les actions de contrôle seront ainsi regroupées selon leur portée, locale ou globale.

Nous pouvons intégrer tous les composants possibles d'une unité de production, jusqu'à l'intégration de réseaux informatiques, généralement très peu pris en compte dans les systèmes de pilotage.

Cette technique de modélisation permet, de manière systématique, de créer un modèle arborescent du système de production. La représentation structurelle fournie, met en évidence les différents niveaux de commande du système. Cet arbre de recherche sert de base pour générer les règles destinées au système expert chargé du pilotage de l'atelier.

# CHAPITRE III

## Elaboration du gestionnaire

## INTRODUCTION

La première étape de spécification et de modélisation est terminée. L'unité de production est décrite et modélisée à partir des graphes d'états associés aux comportements des machines et à leurs interactions avec une prise en compte des actions possibles. Les différents niveaux de commande ont été mis en évidence avec la construction d'une arborescence structurée et hiérarchisée.

La prochaine étape est l'élaboration de son système de pilotage. Sa conception est basée sur les spécifications retenues : elle aboutit à l'intégration de plusieurs systèmes d'inférence.

L'atelier flexible de l'EC Lille sert de base et de support de développement pour l'implantation effective de ce système de pilotage, exemple que nous avons intégré tout au long de ce mémoire.

Nous avons cherché à intégrer et à automatiser par l'intermédiaire de l'informatique, l'ensemble des fonctions de commande, ceci dans le but de nous rapprocher d'un contexte CIM. Cela a conduit à l'élaboration de plusieurs logiciels et nous montrerons comment nous connectons le prototype de gestionnaire de modes de marche aux autres tâches de supervision, telles que la surveillance et la coordination.

L'outil retenu pour concevoir ce système de pilotage est de type système expert [BON 86] dont nous allons présenter une réalisation. Il a servi de prototype pour l'exploitation de la modélisation présentée au second chapitre.

La principale difficulté réside dans la génération de sa base de connaissances. Pour cela, nous exploitons les différentes modélisations (graphes d'état et arbres), présentées au chapitre II, à partir desquelles nous générons automatiquement les règles.

Nous présentons donc la base de faits et la base de règles ainsi que les techniques algorithmiques retenues. Cet ensemble de données va constituer la base de travail du gestionnaire. Il doit pouvoir, à l'aide de sa base de connaissances, décider si la marche ou l'arrêt d'un élément de l'unité de fabrication perturbe le fonctionnement normal. Il doit de plus, en cas de non-cohérence de l'ensemble des états du système assurer une remise en conformité dans un mode de marche licite de l'ensemble de la cellule de production.

Nous préciserons la structure générale et l'imbrication de deux moteurs d'inférence : l'un destiné à un raisonnement déductif et l'autre destiné à un raisonnement planificateur.

Ce chapitre est composé de cinq parties :

- la présentation de la structure mécanique et informatique de l'atelier de l'EC Lille, puis les applications logicielles, exemples des travaux présentés dans ce mémoire,
- la présentation du niveau 1 de la commande,
- la présentation du niveau 2 de la commande,
- la présentation du gestionnaire avec la description de son fonctionnement, la composition de sa base de connaissances et la présentation précise de ses moteurs d'inférence,
- la mise en évidence de la génération automatique des règles destinées aux systèmes d'inférence.

# I - PRESENTATION DE L'ATELIER DE L'EC LILLE

L'EC Lille, dans un but pédagogique et de recherche, s'est progressivement équipée d'une unité de fabrication flexible. Cette unité sert de support et de modèle aux prototypes développés dans le cadre de notre travail.

Deux cellules se distinguent fonctionnellement : une cellule réservée au stockage et au montage des pièces, et une cellule réservée à la fabrication effective des pièces.

## I . 1 - Equipement en machines et robots du procédé [MAY 87]

Deux machines à commandes numériques sont installées : un tour et un centre d'usinage.

**Le tour** est un modèle HES 400 de chez HERNAUT SOMUA. Les principales caractéristiques résident dans le fait que le directeur de commande, une armoire à commande numérique NUM 760T, est intégré à la machine. La NUM comprend deux unités de traitement et un automate programmable associé.

**Le centre d'usinage** est un modèle CU 60LRM, de la société GRAFFENSTADEN et comprend un changeur automatique d'outils, avec un directeur de commande NUM 760F.

Ces deux machines sont équipées d'une carte de communication UNI-TELWAY, ce qui autorise leurs intégrations par le système de pilotage.

**Un robot de type AFMA** est chargé du chargement/déchargement du tour.

**Un robot de type CINCINNATI** est chargé des opérations de chargement/déchargement du centre d'usinage.

**Un robot portique** [DAN 91] CROCUS de la société GT-productique est chargé des opérations de chargement/déchargement vers un stockeur ou la station d'assemblage.

Tous les robots sont associés à des ROBONUM 800. Le choix des robots a été orienté par le fait, qu'ils répondent tout d'abord aux tâches qui leurs sont assignées, mais que par ailleurs, ils sont facilement intégrables dans l'unité flexible au point de vue de la commande. Leurs langages de programmation doivent être complets et offrir des primitives de communications.

**Un stockeur** permet le stockage des pièces brutes et finies.

**Un convoyeur à chaînes** permet le transport des pièces entre les différentes machines de la cellule. Deux chaînes indépendantes, une par îlot, transportent les objets déposés sur des palettes.

**Une station d'assemblage** [GIA 89] réalise l'assemblage des pièces usinées. Celles-ci sont transférées par le robot-portique, soit en provenance du convoyeur, soit en provenance du stockeur. Un AT industriel 7532 contrôle le robot d'assemblage. Un PS/2 surveille et gère les ordres de transferts provenant des automates. De plus, il stocke les fichiers de commandes nécessaires.

### I . 2 - Equipement en machines de commande

Deux automates TSX 47 sont retenus. Ils sont chargés de la commande des différentes machines et robots de l'atelier.

Leur architecture est de type multi-fonctions : multi-langages (en particulier le langage Grafcet), multi-tâches et multi-processeurs. Ils possèdent des équipements permettant l'utilisation de modules d'interfaces diverses et de coupleurs de communications, en particulier d'un coupleur réseau Telway 7. Ceci autorise le dialogue entre les deux automates.

Les deux TSX sont connectés aux robots AFMA et CINCINNATI via une liaison série.

Un micro-ordinateur de type PS/2 assure le pilotage du robot portique et est connecté, via une liaison série, aux automates.

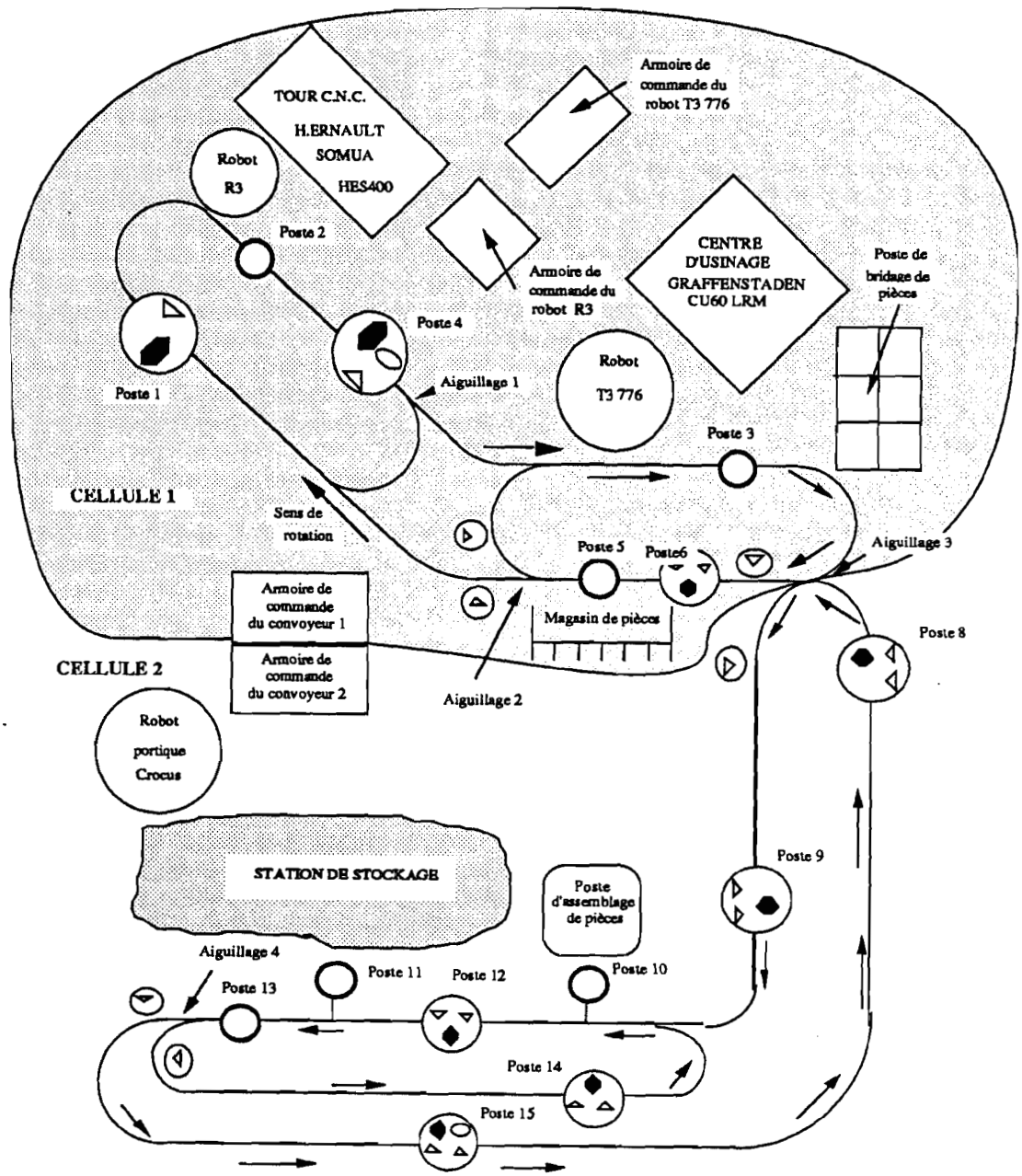
### I . 3 - Equipement en machines de supervision

Une station de travail VAX-GPX de chez DEC (Digital Equipment Corporation) assure les différentes fonctions de pilotage et de surveillance. Elle est connectée, via un réseau ETHERNET à un VAX 3600. Celui-ci est chargé de la partie Conception et Fabrication Assistée, avec des applications disponibles pour le développement tel que le logiciel EUCLID.

### I . 4 - Description physique de l'atelier

Tout au long du convoyeur, sont disposés des capteurs et des butées de blocage de palettes. Leurs placements sont organisés de façon à constituer :

- des postes de chargement/déchargement : postes P2, P3, P5, P10, P11,
- des files d'attentes : postes P1, P4, P8, P9, P13, P14, P15.




**Eléments pouvant entrer dans la composition d'un poste**

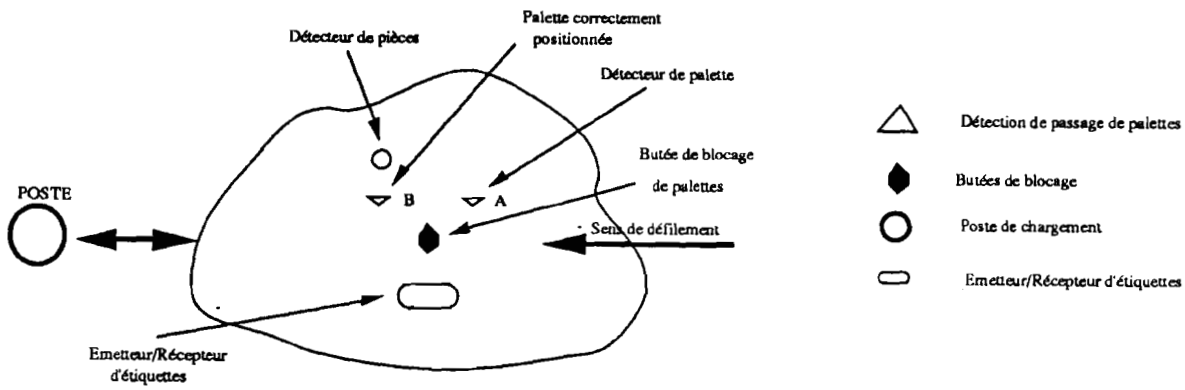



fig 1: architecture matérielle de l'atelier de l'EC Lille

Un dispositif d'étiquettes BALOGH permet de référencer les palettes du système. Ces étiquettes sont accessibles en lecture et en écriture à chaque poste où il se passe effectivement une opération susceptible de modifier l'état de la palette et de sa pièce. Un micro-ordinateur de type PS/2 est utilisé pour la gestion des étiquettes.

### I . 5 - Gammes opératoires

Plusieurs opérations d'usinage sont donc possibles : tournage, fraisage et assemblage, cela autorise diverses gammes opératoires : une pièce peut être tournée, éventuellement tournée une seconde fois, fraisée, tournée puis fraisée, ou fraisée puis tournée. Un assemblage est prévu entre deux pièces préalablement usinées (une étant tournée puis fraisée, l'autre étant fraisée).

Nous identifions chaque type de pièces par un jeu de notations [BOU 89]. Nous appelerons :

"t" les pièces à tourner,
"f" les pièces à fraiser,
"a" les pièces à assembler,
"-" les pièces bruts,
"+" les pièces usinées.

Ces notations permettent de connaître précisément l'état d'avancement de chaque pièce dans leur gamme opératoire.

Les palettes sont notées "p". Trois types de palettes, appelées $p_1$, $p_2$ et $p_3$, sont prévus et associés chacun à un type de pièce différent. Le nom d'une pièce palettisée sera celui de son identificateur d'état courant associé à celui de la palette (par exemple, $p_2$f-). Cependant, pour des raisons de simplicité, nous n'indiquerons pas le numéro de palette, qui se déduit par le type de la pièce.

D'après ces notations, $p_2$f+t- est une pièce palettisée sur une palette de type 2, qui a été fraisée et qui doit être tournée.

Les pièces brutes prévues sont :

**t-** dont la palette associée est $\mathbf{p_1}$ et le support de bridage/débridage est $\mathbf{pl_1}$,
**f-** dont la palette associée est $\mathbf{p_2}$ et le support de bridage/débridage est $\mathbf{pl_2}$,
**t-t-** dont la palette associée est $\mathbf{p_3}$ et le support de bridage/débridage est $\mathbf{pl_3}$,
**t-f-** dont la palette associée est $\mathbf{p_3}$ et le support de bridage/débridage est $\mathbf{pl_3}$,
**f-t-** dont la palette associée est $\mathbf{p_2}$ et le support de bridage/débridage est $\mathbf{pl_2}$,

**a-** dont la palette associée est $\mathbf{p_2}$ et le support de bridage/débridage est $\mathbf{pl_2}$. a- est le résultat du positionnement de la pièce t+f+ sur la pièce f+. L'assemblage effectif consiste à souder ces deux pièces.

Plusieurs gammes opératoires sont possibles et résultent de la combinaison des opérations possibles. Cependant, l'opération de stockage est toujours considérée implicitement dans le cycle de fabrication, comme la dernière opération à effectuer et n'est pas précisée dans la définition des gammes opératoires.

Les pièces suivent les gammes opératoires suivantes :

| | | | | |
|---|---|---|---|---|
| t- | → | t+ | | |
| t-t- | → | t+t- | → | t+t+ |
| f- | → | f+ | | |
| t-f- | → | t+f- | → | t+f+ |
| f-t- | → | f+t- | → | f+t+ |

## I . 6 - Cycle de fabrication

Le cycle de fabrication met en oeuvre des opérations de fraisage, de tournage, et de stockage. Les actions sont déterminées au lancement du processus de fabrication de la pièce et caractérisent le type de la pièce.

Les cycles de fabrications débutent en général, par l'introduction d'une pièce en provenance du magasin de pièces (transfert TAMPON vers P5). La pièce est palettisée par le robot CINCINNATI. La pièce est alors, soit dirigée vers le poste de fraisage, soit vers le poste de tournage.

Dans le cas du tournage, le robot AFMA dépalettise et charge la pièce sur le tour, à partir du poste P2. Par ailleurs, il palettise la pièce finie sur le convoyeur.

De même, dans le cas du fraisage, le robot CINCINNATI charge et décharge la machine de fraisage à partir du poste P3. Avant d'être fraisées, les pièces sont bridées sur un support. Le bridage reste une opération manuelle.

Une pièce fraisée peut être redirigée vers le tour.

Les opérations de fabrications étant terminées, les pièces palettisées sont dirigées vers le poste d'assemblage ou le poste de stockage via les postes P3 et P9.

Un système de transfert avec guides, au niveau des postes P11 et P10, permet de diriger une palette vers les postes de stockage et d'assemblage. Ceci évite de bloquer le cheminement des pièces sur le convoyeur.

L'introduction d'une pièce brute peut se faire à partir du stockeur par le robot-portique (transfert STO vers P11) et dirigée vers l'îlot de fabrication (transfert P13 vers P15, puis vers P8, P6, P5).

L'ensemble du cycle de fabrication est automatique, à l'exception des opérations de bridage/débridage. Les fichiers de commande sont, soit stockés en mémoire des machines, soit téléchargés à partir de la GPX.

Prenons comme exemple, la réalisation de la gamme opératoire tournage puis fraisage.

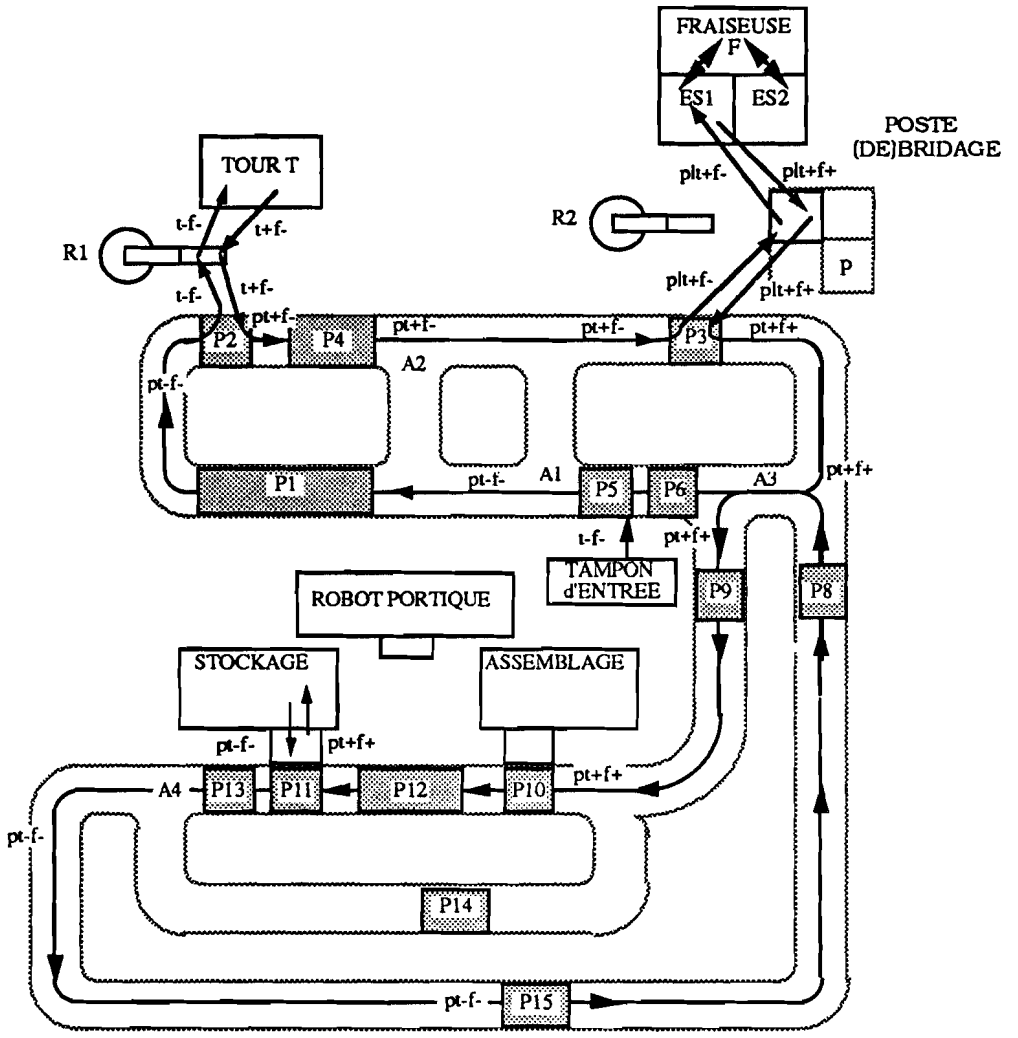



fig 2 : cycle de fabrication t-f-

## I . 7 - Structure du système de pilotage

La structure logicielle implantée à l'EC Lille tend vers une structure de type C.I.M. dont la structure de commande se rapproche de la décomposition classique en trois couches hiérarchisées, présentée au chapitre I : le procédé, la partie commande et le niveau hiérarchique ou décisionnel.

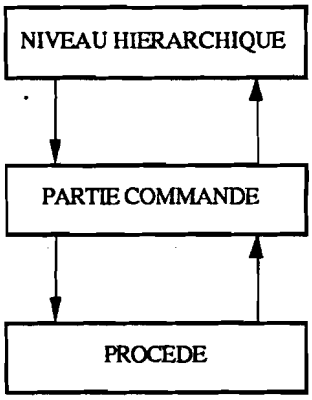



fig 3 : les différents niveaux de commande

Nous allons décrire l'architecture de commande retenue.

**Le Procédé** est composé des divers équipements et machines utilisés dans la production, présentés auparavant.

**La Partie Commande** assure la synchronisation et la coordination des actions du procédé. Elle traite les informations envoyées par les capteurs et renvoie ensuite des ordres vers les actionneurs.

**Le Niveau Hiérarchique** permet, dans un premier temps de résoudre les indéterminismes et conflits qui peuvent survenir lors de la coordination du graphe de commande. Dans un second temps, il traite des problèmes du type ordonnancement, planification de tâches et détections d'erreurs.

L'architecture informatique adoptée est présentée sur la figure N°4. Nous avons distingué les différents niveaux de commande, en regroupant les machines de même niveau.

Nous pouvons remarquer que des machines techniquement identiques sont situées à des niveaux différents, de par les fonctions qu'elles assurent. Ainsi, un micro-ordinateur PS/2 est situé au niveau I et deux autres au niveau 0. Cette distinction résulte du fait que le PS/2 du niveau I est chargé de tâches fonctionnellement associées à la gestion de l'ensemble de l'atelier et en particulier à l'allocation des emplacements de stockage tandis que les deux autres PS sont simplement équivalents à des armoires de commande affectées exclusivement à des ressources de production.

La décomposition multi-niveaux n'est évidement pas seulement matérielle mais également et surtout fonctionnelle. D'après la figure 4, le niveau 3 relève de la conception et non de la conduite de l'atelier de production ; il est normal qu'il ne soit pas repris sur la figure 3 qui décompose uniquement les niveaux d'élaboration de la commande d'une unité de fabrication.

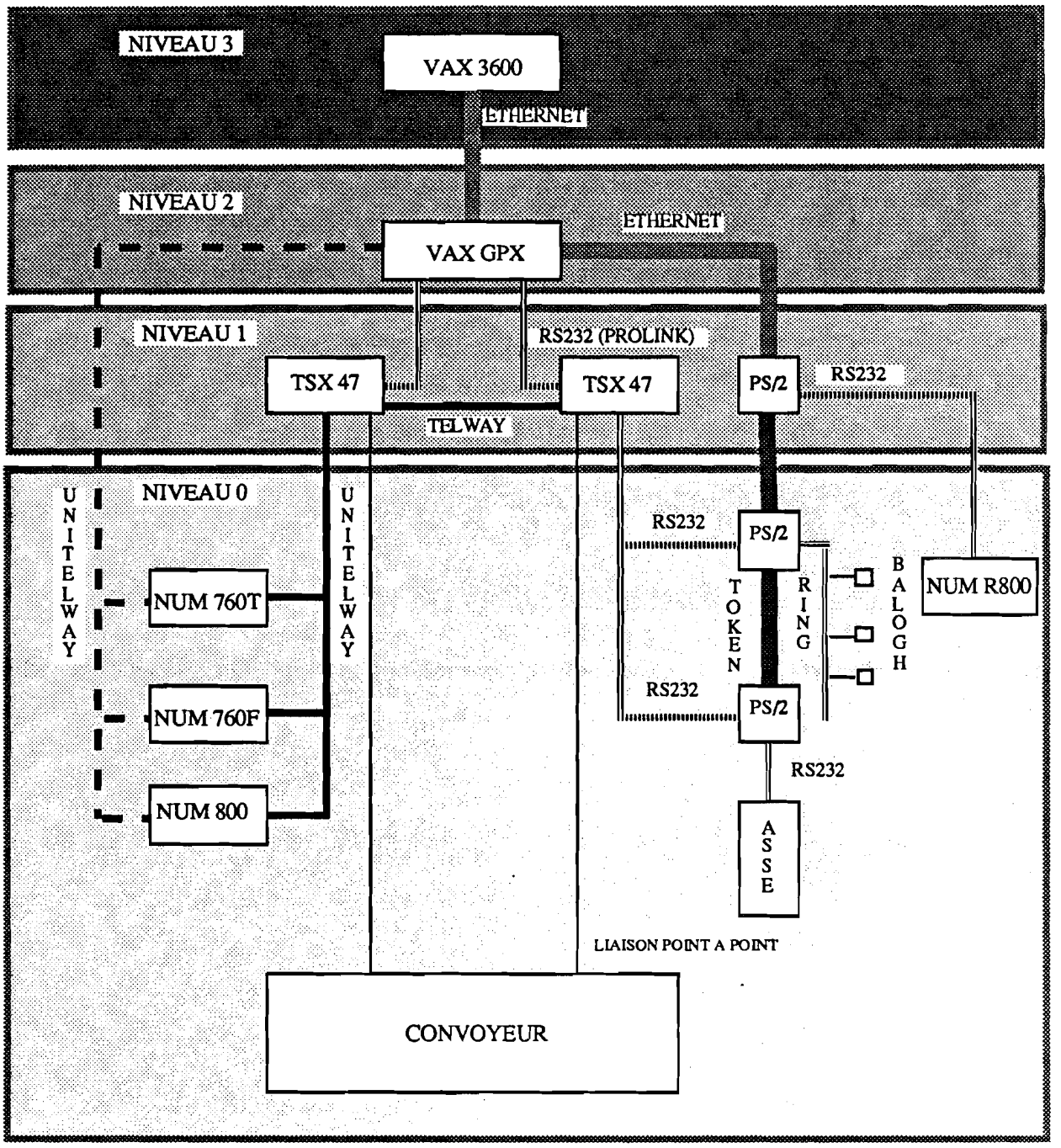



fig 4 : architecture informatique

Nous allons donc détailler successivement la partie commande et le niveau hiérarchique implantés pour la supervision de l'atelier.

## II - NIVEAU 1 DE LA COMMANDE

La partie commande principale consiste en un graphe de coordination, chargé de synchroniser le mouvement des pièces sur le convoyeur et les commandes de fabrication du système. Ce graphe est implanté sur les automates TSX 47, en langage Grafcet.

Un logiciel de contrôle [DAN 91] de la partie commande du robot portique et du stockeur a été réalisé et développé en langage ADA. L'appel à un langage de programmation structurée a été rendu nécessaire par la complexité de la commande de ce robot portique associée à celle de la gestion des emplacements du stockeur.

La démarche de conception de la partie commande reprend celle du projet CASPAIM, présentée au chapitre I.

Le principal critère retenu lors de la génération du graphe, est de rendre le système de coordination le plus flexible possible. C'est à dire qu'il s'agit d'intégrer tous les chemins possibles entre les différentes machines en fonction des gammes opératoires envisagées et de prévoir les modes de marches dégradées qui découleraient de pannes de machines. Les domaines de couleurs contiennent donc toutes les marques possibles.

Cette approche orientée vers la flexibilité maximale, permet en outre, d'intégrer relativement facilement de nouvelles gammes opératoires non prévues initialement à la conception du système. En effet, une reconfiguration adéquate des domaines de couleurs caractérisant le cheminement licite des pièces autorise l'introduction d'une nouvelle gamme.

La démarche débute tout d'abord par l'élaboration d'un prégraphe à partir des gammes opératoires en tenant compte de la stratégie retenue. Celle-ci repose sur l'intégration des différents modes de marche, qui doit se faire dès la phase de conception. En effet, un changement de mode de marche modifie l'organisation du cheminement des pièces et des produits à travers les différentes machines. Le routage est enrichi par la création de nouveaux chemins, qui se traduisent en fait par la création d'arcs reliant des lieux physiques initialement non reliés lors d'un fonctionnement normal. Cela signifie donc un enrichissement des domaines de couleurs, de façon à inclure les couleurs supplémentaires. Nous reviendrons sur ce problème, lors de la présentation d'un changement de modes de marche.

Le prégraphe obtenu de l'atelier, est présenté ci-après :

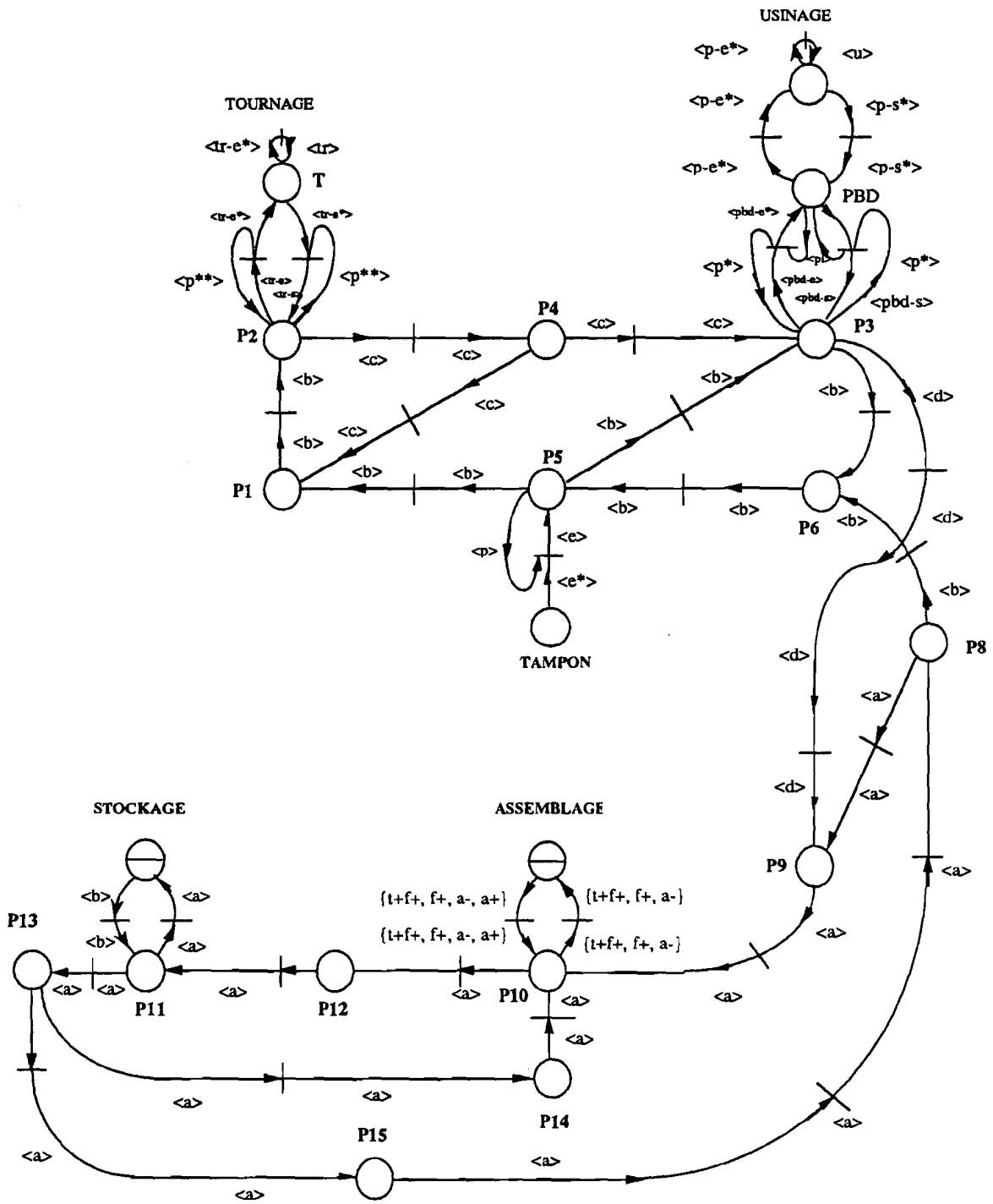



fig 5 : élaboration du prégraphe

Les domaine de couleurs sont ceux prévus pour tous les cas : normaux ou exceptionnels. Cependant, nous imposons aux pièces dont les opérations d'usinage sont terminées d'être systématiquement dirigées vers les lieux de stockage.

Les domaines de couleurs sont :

**Tampon d'entrée**

<e*>={t-, f-, f-t-, t-f-, t-t-}

<e>={pt-, pf-, pf-t-, pt-f-, pt-t-}

**Convoyeur**

<p>={$p_1$, $p_2$, $p_3$}

<p*>={$p_2$, $p_3$}

<p**>={$p_1$, $p_3$}

<a>={$p_1$, $p_2$, $p_3$, pt+, pt+, pt-t-, pt+t-, pt+t+, pt-f-, pt+f-, pf+t-, pf+t+, pf-, pf-t-, pt+f+, pf+, pa-}

<b>={$p_1$, $p_2$, $p_3$, pt-, pt-t-, pt+t-, pt-f-, pt+f-, pf+t-, pf-, pf-t-}

<c>={$p_1$, $p_2$, $p_3$, pt-, pt+, pt-t-, pt+t-, pt+t+, pt-f-, pt+f-, pf+t-, pf+t+, pf-, pf-t-}

<d>={$p_1$, $p_2$, $p_3$, pt+, pt+, pt-t-, pt+t-, pt+t+, pt-f-, pt+f-, pf+t-, pf+t+, pf-, pf-t-, pt+f+, pf+}

**Tour**

<te>={pt-, pt-t-, pt+t-, pt-f-}

<te*>={t-, t-t-, t+t-, t-f-}

<tr*>={t+, t+t-, t+t+, t+f-}

<ts*>={t-, t+, t-t-, t+t-,t+t+, t-f-, t+f-}

<ts>={pt-, pt+, pt-t-, pt+t-, pt+t+, pt-f-, pt+f-}

**Centre d'usinage**

<pbd-e>={pf-, pf-t-, pt+f-}

<pbd-e*>={f-, t+f-,f-t-}

<u-e*>={plf-, plt+f-, plf-t-}

<u>={plf+, plt+f+, plf+t-}

<u-s*>={plf-, plt+f-, plf-t-, plf+, plt+f+, plf+t-}

<pbd-s*>={f-, f+, t+f+, t+f-, f+t-, f-t-}

<pbd-s>={pf-, pf+, pt+f+, pt+f-, pf+t-, pf-t-}

La seconde phase est l'élaboration du graphe de commande en RDPSAC (Réseaux de Petri Structurés Adaptatifs et Colorés). La phase de structuration a généré un graphe dont la taille est assez importante, avec 67 processus indépendants, 430 places et 244 transitions.

Nous montrons, comme exemple, le graphe modélisant la gestion du convoyeur chargé des transferts des pièces au niveau de l'extension de l'atelier, c'est à dire entre les postes de stockage et d'assemblage.

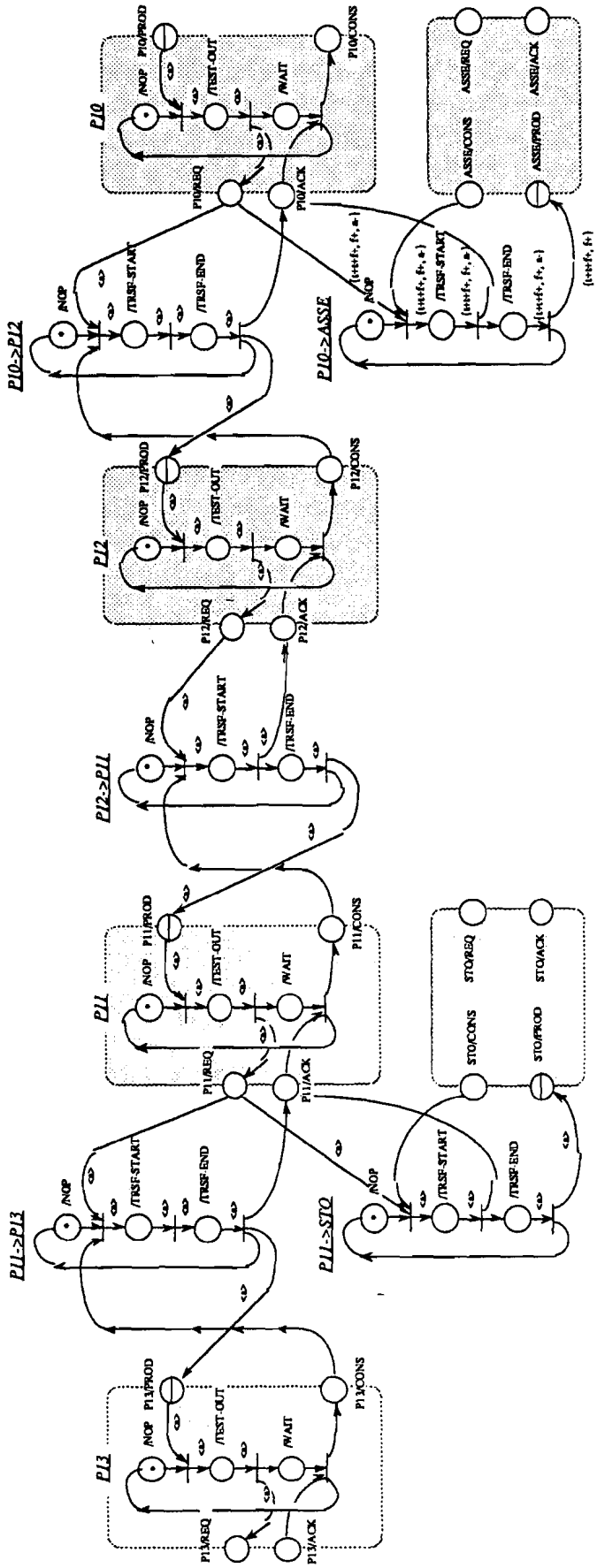



fig 6 : graphe de commande

## II . 1 - Simulation

Deux objectifs sont à réaliser :

- la validation des spécifications du modèle,
- le dimensionnement du modèle (zones tampons, temporisations).

La simulation nous a permis de détecter des situations de blocage, résolues par le choix de séquences d'entrées, et de faire une première étude qualitative et quantitative du comportement général, avec la mise en place des stratégies d'allocation.

## II . 2 - Implantation

La phase d'implantation consiste à la réalisation effective de la partie commande. Les problèmes liés à l'implantation sont de plusieurs ordres : **respect du modèle initial théorique, taille du modèle ainsi que sa répartition sur sites réels**. En effet, l'étude théorique permet d'esquiver certains problèmes, qui apparaissent lors de l'implantation et il s'agit d'adapter la modélisation théorique à la situation effective. Dans notre cas, l'étude d'un modèle équivalent en langage Grafcet autorise une implantation rigoureuse et surtout fidèle au modèle théorique. La phase d'implantation du graphe de commande sur les automates TSX, est systématique [CRA 89].

La structure générale du modèle initial est donc traduite en langage GRAFCET, d'après la méthode présentée au chapitre I, et implantée sur TSX.

### II . 2 . 1 - Répartition du graphe de commande

L'implantation du graphe de commande est réalisée et répartie sur deux automates TSX 47. L'utilisation de deux automates est pleinement justifiée par plusieurs critères. En premier lieu, le nombre d'entrées/sorties à assurer, s'avère dépasser la capacité d'un seul automate ; deuxièmement, la taille de l'application modélisée en Grafcet est supérieure à la capacité mémoire automate. Enfin et surtout, la description physique de l'installation montre la présence de deux cellules relativement autonomes et dont les commandes sont fortement découplées.

La répartition s'est faîte naturellement. En effet, les deux cellules qui sont fonctionnellement distinguées, sont de taille équivalente et ont le même degré de complexité au niveau des opérations de transferts. Leurs graphes de commande associés sont donc de dimensions sensiblement égales. L'idée étant d'avoir une charge uniformément distribuée, la solution de répartition a donc été de réserver un automate pour chacune des cellules. La communication demeure restreinte et se limite à deux connexions entre processus : (liaisons P8

→ P6 et P3 → P9), qui sont facilement appréhendées par le support de communication TELWAY 7.

L'avantage de cette architecture logicielle est la possibilité de pouvoir utiliser chaque cellule de manière indépendante, grâce au très fort découplage réalisé tant au niveau de la commande qu'au niveau fonctionnel.

### II . 2 . 2 - Implantation de la coloration

Nous allons insister sur le problème de la coloration et de son implantation. En effet, cette notion caractérise l'originalité de notre commande. Le plus souvent, les logiciels de commande décrivent séquentiellement les opérations à effectuer et par exemple, dans le cas du manufacturier, présentent un code spécifique à chaque gamme opératoire. Cela aboutit à une commande figée et difficile donc à reprogrammer. La démarche CASPAIM décrit, pour sa part, les opérations en tenant compte du processus de fabrication, les machines et les usinages. Le code n'est donc pas spécifique à une pièce. Nous obtenons une commande plus condensée et facilement paramètrable.

Il est à remarquer que les étiquettes Balogh ne sont pas utilisées de manière optimale, par le fait qu'elles véhiculent des informations complètement redondantes par rapport à celles contenues au niveau du graphe de commande. Ceci montre d'ailleurs la richesse et la puissance du modèle utilisé, puissance qui pourrait être d'ailleurs accrue par l'utilisation d'un Réseau de Petri Objets. Les étiquettes servent en particulier, à récupérer des informations lors d'un démarrage complet, lors de pannes, ou dans le cadre de la surveillance. Une lecture exhaustive de toutes les étiquettes permet une mise à jour du graphe, via un PS/2 chargé de l'opération.

Le caractère structuré du modèle RdP permet de distinguer des processus indépendants et saufs. Une seule variable par processus est alors nécessaire pour la gestion de la coloration. Une variable de type mot est définie par processus.

L'introduction de la flexibilité dans les gammes opératoires entraîne des indéterminismes directionnels. Un indéterminisme se pose lorsque la commande a plusieurs possibilités d'évolution, c'est à dire lorsqu'une pièce, au niveau d'un aiguillage, a le choix de se diriger dans plusieurs directions.

Dans le cas de l'implantation des domaines de couleurs, nous avons simplifié le passage du Réseau de Petri au Grafcet, en ne conservant que ceux situés à des endroits susceptibles d'engendrer des indéterminismes directionnels. En effet, si le chemin est explicite et sans alternative, il n'y a pas besoin de test. Lors de la phase de conception, nous avons pris en compte toutes les conséquences que pouvaient occasionner des changements de modes de marche sur la configuration du graphe, par l'intégration de tous les chemins possibles au niveau des domaines de couleurs. Nous réduisons la taille des domaines de couleurs, considérant

uniquement un mode de marche normale ; ceci permet de lever un nombre important d'indéterminismes. Par exemple, au niveau du poste P5, une pièce à tourner sera obligatoirement dirigée vers le tour tandis que les spécifications du prégraphe intègrent la possibilité de chemins de remplacement en cas de panne du tour. La pièce à tourner a alors la possibilité de se diriger directement vers P3, sans passer par le tour.

Nous simplifions ainsi le graphe de commande, ce qui permet de limiter l'occupation de la mémoire automate, lors de son implantation.

Nous présentons dans le dessin suivant, les places susceptibles d'occasionner des indéterminismes.

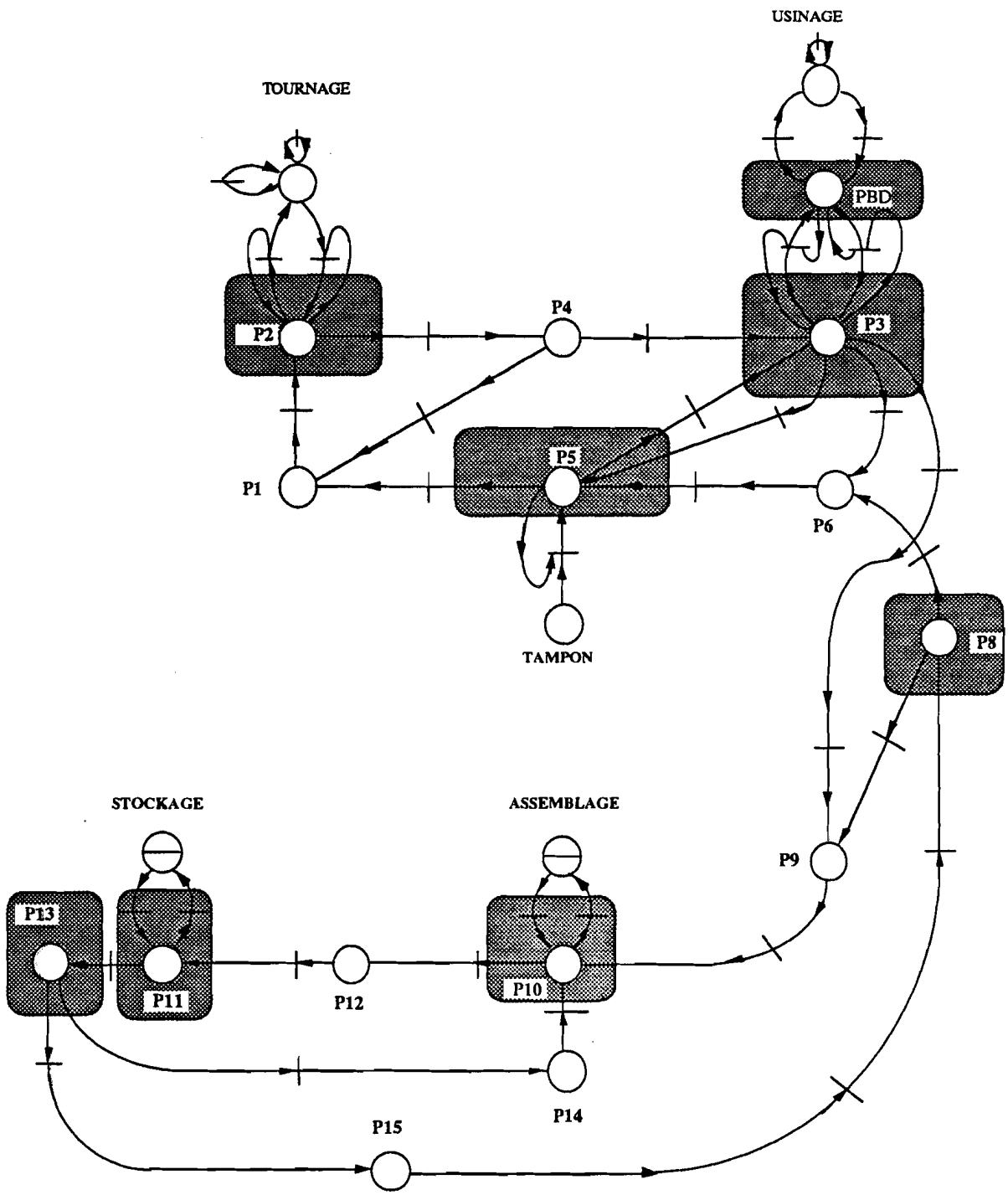



fig 7 : domaines stratégiques

Pour l'implantation effective en Grafcet, nous reprenons les tests, effectués sur les arcs des réseaux de Petri, au niveau des étapes. Les ensembles de couleurs sont représentés par des tableaux qui contiennent l'ensemble des codes admissibles. La taille maximale des tableaux est limitée à 20 types de pièces possibles, ce qui suffit largement dans le cas de l'atelier de l'EC Lille. Afin de sélectionner le chemin possible, les tests d'appartenance se feront avec une série de IF en langage littéral.

### II . 2 . 3 - Exemple d'implantation des domaines de couleurs

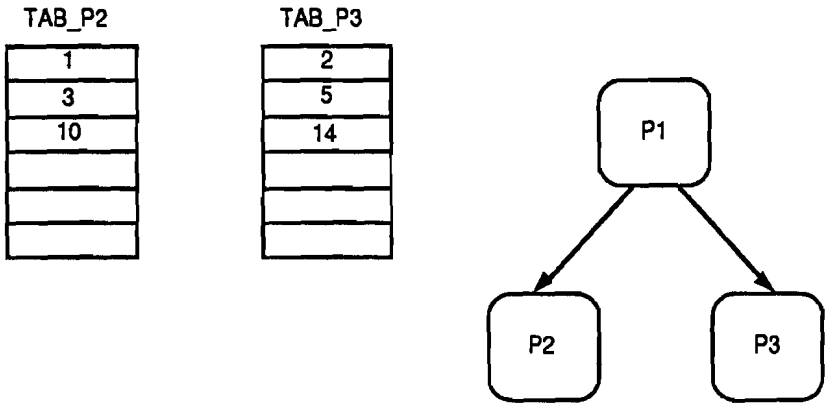



fig 8 : implantation des domaines

TAB_P2 et TAB_P3 contiennent respectivement les codes des pièces admises à circuler vers P2 ou P3.

Un test est effectué au niveau du poste P1.

Si le code de la pièce appartient à TAB_P2 alors la pièce est dirigée vers P2 sinon si le code de la pièce appartient à TAB_P3 la pièce est dirigée vers P3 sinon la pièce reste en attente, au poste P1.

Ces tables sont accessibles en lecture et écriture par le Niveau hiérarchique, dans le cadre de la gestion dynamique des domaines de couleurs du graphe de commande.

Les indéterminismes restants sont levés par le niveau hiérarchique, constitué en fait par un logiciel de coordination.

### II . 2 . 4 - Exemple d'implantation d'un modèle de type lot

Jusqu'à présent nous n'avons représenté que des objets unitaires. Une difficulté d'implantation se pose lorsque les objets sont des lots.

Une solution simple est de gérer les domaines de couleurs de manière différente, en remplaçant les valeurs de couleurs par des adresses de zones mémoires contenant effectivement les valeurs. Ceci oblige à modifier la programmation correspondant aux tests de couleurs.

Prenons le cas d'un lot de 4 pièces : nous réservons une zone mémoire de 5 octets, dont l'adresse de départ est adr. La première valeur est le nombre de pièces, les valeurs suivantes étant le type de chacun des objets.

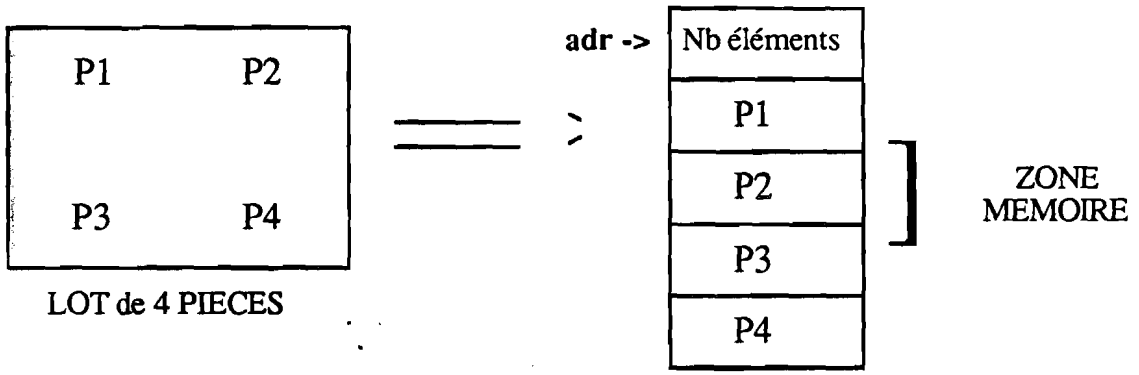



fig 9 : représentation d'un lot

Une pièce seule est considérée comme un lot à une pièce.

La difficulté de cette solution est que cela nécessite beaucoup de place mémoire, et qu'il faut toujours surestimer le dimensionnement des zones mémoires réservées.

Nous avons développé le graphe de commande de l'atelier suivant ce modèle et l'avons implanté. Du fait de la non autonomie du modèle et du caractère non exploitable du graphe, un développement d'un logiciel d'interfaçage s'est avéré nécessaire, nous le présenterons par la suite.

# III - NIVEAU 2 DE LA COMMANDE

## III . 1 - Structuration du Niveau Hiérarchique

Le niveau 2 est le niveau hiérarchique du système de commande. La diversité et la complexité de ces fonctions obligent à élaborer des logiciels, adaptés à chacune d'elles. Ils sont associés à la marche de l'atelier. De ce fait, nous désirons les gérer dans le cadre de la gestion des modes de marche de l'atelier, ce qui permettra de les coordonner et de les structurer de manière cohérente.

La structure retenue du système de pilotage est la coordination de ces logiciels de commande par le prototype de gestionnaire, noyau centralisateur autour duquel s'articulent ces logiciels.

Les logiciels intégrés sont :

- **une interface utilisateur** développée pour permettre la commande manuelle des automatismes et permettre leur configuration. De plus, il permet d'accéder à la gestion manuelle des différents modes de marche du système,

- **un coordinateur d'automates** chargé de résoudre les indéterminismes et les conflits de la partie commande, présenté au chapitre I [CRA 89], qui fonctionne de manière complètement automatique,
- **un module de surveillance** [TOG 90] qui est chargé de la détection des erreurs et des défauts. Un prototype, basé sur un outil de type système expert, est développé actuellement.

L'implantation du gestionnaire de modes de marche nécessite de prévoir son intégration vis à vis des 3 logiciels, développements antérieurs ou en_cours, associés au fonctionnement de l'atelier et ses modes de marche. Nous les avons donc intégrés dans le cadre de la supervision effectuée par le gestionnaire.

Nous montrons comment sont interconnectés les différents modules de commande .

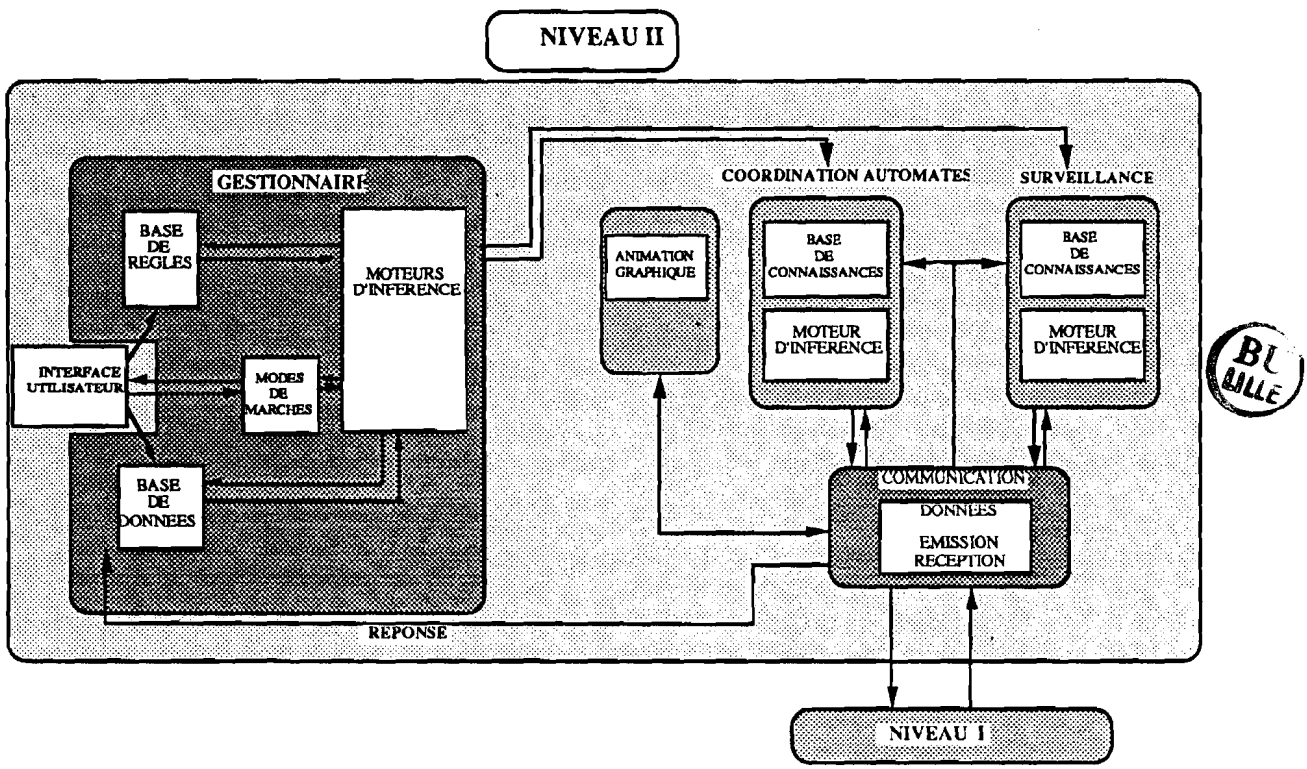



fig 10 : organisation du niveau hiérarchique [BOI 90]

Parallèlement à sa fonction dans la gestion des modes de marche, nous remarquons la place essentielle du gestionnaire vis à vis des autres fonctions du système de pilotage. Il assume la fonction de coordinateur de ces différents modules de conduite (déclenchements, arrêts...), gérés en fonction de l'état de l'atelier (analogie avec un système d'exploitation, présentée au chapitre II). De plus, sa vision est plus large et plus générale par rapport à celles plus restrictives du système de détection d'erreurs ou du logiciel de coordination de la partie commande, aux fonctions plus spécialisées. Le gestionnaire centralise des données vitales à la

gestion des modes de marche, fournies par l'utilisateur ou proposées par les différents modules, en particulier par celui de détection.

**Une partie communication** gère les entrées/sorties des différents logiciels avec les couches de niveaux inférieurs, la partie commande et le procédé. Il s'agit d'un dialogue entre la station de travail GPX et les automates TSX 47, via une liaison série [HUV 90].

L'originalité de l'installation a été de réutiliser un protocole de communication déjà implanté, PROLINK. En effet, un logiciel d'animation, IMAGIN (destiné à aider à la compréhension rapide des résultats et à la maintenabilité), est installé et permet la visualisation de l'état de l'atelier à partir de données provenant des automates. Le principe est de mettre simultanément à jour la base de données de chacun des logiciels de commande, par une opération de lecture. Les données utilisées par les programmes écrits en Le_Lisp, sont donc mises à jour régulièrement, par une requête régulière de lecture complètement transparente à l'utilisateur.

Nous détaillons la structure et les fonctions de l'interface utilisateur et ensuite l'implantation du gestionnaire et ses liens avec les autres logiciels.

### III . 2 - Interface utilisateurs

De façon à pouvoir répondre à l'exigence d'interactivité, un logiciel [GEN 90] a été développé pour faciliter l'interfaçage du niveau hiérarchique avec la partie commande et rendre le dialogue utilisateurs/machines convivial et automatique. Elle permet en fait, un mode manuel et interactif des logiciels destinés à la gestion de l'atelier de l'EC Lille.

Dans cette optique, une bibliothèque de fonctions a été créée, avec l'utilisation des primitives de communication spécifiques au support de transmission, et adaptées aux matériels utilisés.

Nous avons ainsi simplifié et systématisé :

- l'interfaçage de la partie commande de niveau 1 (graphe de synchronisation des automates) avec les logiciels du niveau supérieur,
- la lecture et l'écriture de variables internes aux automates, ce qui autorise ainsi le suivi et le contrôle direct par l'utilisateur,
- l'appel à certaines fonctions système sur automates, initialement non accessibles directement par l'utilisateur,

- la gestion des domaines de couleurs pour les opérations de reconfiguration présentée par la suite.

L'utilisateur a la possibilité ainsi d'accéder aux variables significatives internes aux automates et de directement contrôler les différents modes de marche de l'ensemble des machines en mode manuel.

La base de données est une image de l'état interne des automates (variables, domaines de couleurs...). Elle contient la liste des variables utilisées avec en particulier leurs adresses physiques.

Ce logiciel est développé en langage Le_Lisp [CHA 89] afin de faciliter son intégration vis à vis des autres logiciels de supervision.

Les fonctions de la bibliothèque sont accessibles soit à partir du menu principal de l'interface, appelé directement au niveau de l'environnement Le_Lisp, soit par d'autres programmes.

Deux modules sont proposés :

- **un module d'édition :** l'utilisateur peut définir sa bibliothèque de fonctions. Il est possible de la mémoriser et de la modifier en fonction des évolutions matérielles de l'atelier,
- **un module d'utilisation** manuelle de ces fonctions : l'utilisateur peut intervenir pour configurer la partie commande.

### III . 2 . 1 - Fonctions accessibles

Pour pouvoir commander les différents automates, nous avons cherché à pouvoir accéder directement aux fonctions système des automates à partir du niveau hiérarchique grâce à un programme écrit en partie préliminaire du cycle automate.

L'utilisateur peut donc accéder aux fonctions système tels que par exemple le gel du Grafcet, l'initialisation du Grafcet et la remise à 0 d'étapes. Ces fonctions sont utilisées dans la gestion des modes de marche.

Nous avons mis en place d'autres fonctions "système" pour pouvoir commander et donc accéder directement aux modes de marche des autres machines telles que le tour et le centre de fraisage. L'utilisateur peut de manière automatique, initialiser les machines, les arrêter ou les mettre en marche.

Nous présentons les différentes fonctionnalités sous forme d'organigramme afin d'en expliquer le fonctionnement.

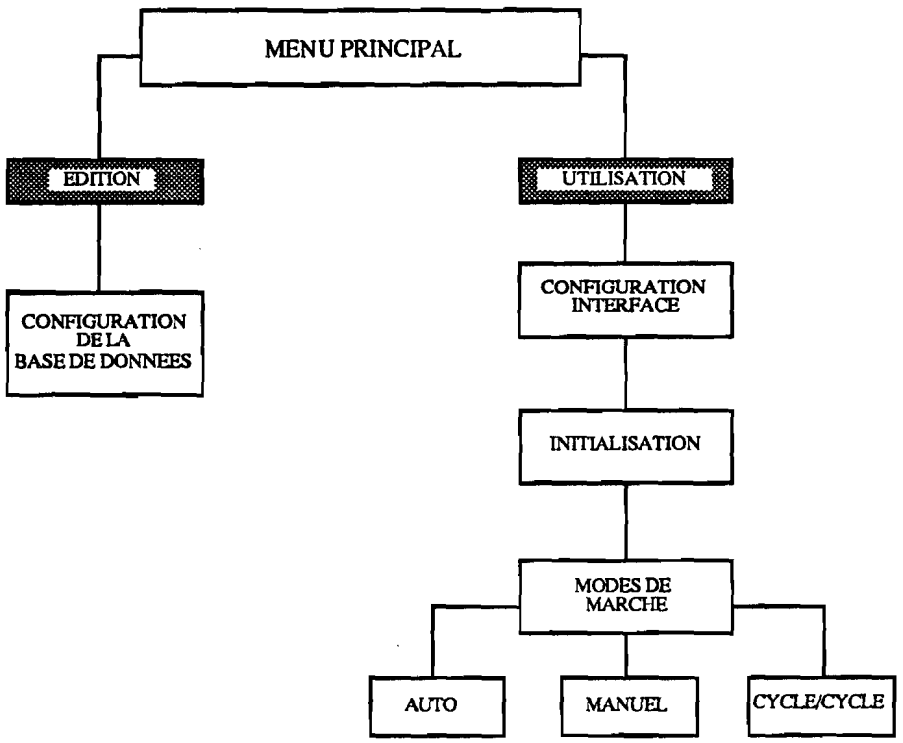



fig 11 : organigramme général

Deux modes sont donc proposés :

**Un mode édition,** où l'utilisateur peut définir des variables pour la configuration de la base de données, spécifique à l'installation matérielle. A tout moment, il est possible, de sauvegarder la base de données en l'état de façon à la réutiliser directement en cas de reprise à chaud. Il est possible de repartir d'une configuration initiale, dans le cas d'une reprise à froid.

**Un mode utilisation,** qui permet d'accéder de manière transparente aux variables significatives internes aux automates. Ces variables sont par exemple des mots, des bits ou des registres. Il est possible ainsi d'accéder en écriture aux différents bits système ce qui permet d'accéder aux différents modes de marche des automates et des différentes machines sous le contrôle des automates.

La principale fonction de cette interface est l'initialisation et la mise à jour du graphe de coordination sur les automates avant toute mise en marche de l'atelier. Le graphe de commande écrit en Grafcet est une image exacte du procédé et reflète la situation exacte de l'état de l'atelier. L'ensemble des pièces et des palettes déjà présentes dans l'atelier est pris en compte. Le graphe de coordination sera contrôlé de façon à évoluer de manière artificielle pour le mettre en

cohérence d'état avec le procédé. Par exemple, les étapes qui correspondent à une attente de pièces seront marquées.

Puisque nous utilisons une liaison série asynchrone comme support physique de communication, nous sommes obligés de développer un programme intermédiaire sur automates car il n'est pas possible d'accéder directement aux bits système. C'est ce qui permet de rendre les appels à ces fonctions, transparents à l'utilisateur. Nous pouvons par exemple, accéder en écriture aux bits système des automates tels que [RES 85]:

- le bit SY21 qui correspond à l'initialisation du Grafcet,
- le bit SY22 qui correspond au gel du Grafcet,
- le bit SY23 qui correspond à la remise à zéro du Grafcet,
- le bit SY24 qui correspond à la remise à zéro des macro-étapes.

### III . 2 . 2 - Gestion dynamique des domaines de couleurs

L'une des principales fonctions intégrées par l'interface, est la gestion des domaines de couleurs. Dans le but d'accroître la flexibilité de l'atelier, nous avons utilisé le principe de la gestion dynamique des domaines de couleurs. Son principe consiste à pouvoir modifier, "en_ligne", les différents domaines de couleurs du graphe de commande et de maîtriser ainsi les flux de pièces circulant dans l'atelier.

Les objectifs sont :

- le règlement local des indéterminismes,
- la gestion des différents modes de marche.

L'introduction d'une nouvelle gamme opératoire, consécutive à un changement de stratégie ou à une dégradation du système de production, nécessite un nouveau paramètrage de la commande à condition évidemment que ce soit en accord avec les gammes opératoires déjà présentes.

**La modification des domaines de couleurs** : il suffit d'aller changer sur les automates le contenu de tables en y introduisant ou en y modifiant des valeurs. L'intégration d'une nouvelle gamme revient à enrichir les domaines de couleurs déjà existants.

L'originalité de cette méthode est qu'elle évite de remettre en cause la structure générale de la partie commande déjà implantée. De plus, dans le cas où surviendraient certains problèmes de fonctionnement (pannes de machines) ou dans le cas de changement de stratégies de production, la mise à jour des domaines permet de prendre en compte rapidement ces problèmes. Ainsi par exemple si une machine de tournage devient indisponible, l'ensemble des

tables contenant des codes correspondant à une opération de tournage sont transformées de façon à tout d'abord intégrer la panne et ensuite à prévoir des chemins de remplacement.

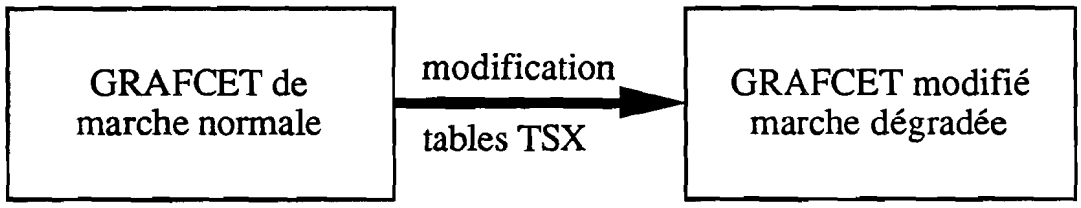



fig 12 : passage Grafcet normal, Grafcet modifié

La difficulté de cette méthode réside dans le fait qu'il faut utiliser une importante structure de données pour la gestion des domaines de couleurs. En effet, il s'agit de modéliser chaque domaine et son contenu.

Par ailleurs, en optimisant la gestion des domaines, nous introduisons la notion de délocalisation du niveau hiérarchique. Les indéterminismes directionnels sont gérés localement ce qui permet de soulager le niveau hiérarchique.

Plusieurs solutions sont possibles. Nous essayons par exemple, d'avoir toujours des domaines de couleurs complémentaires.

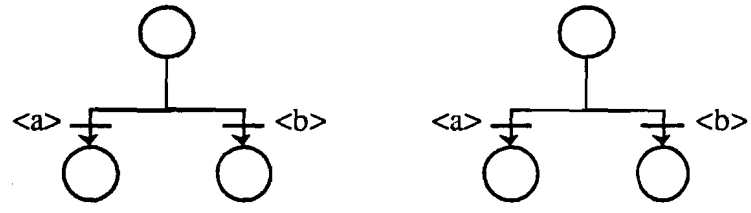



fig 13 : exemple de configuration

La figure (a) présente un indéterminisme directionnel. Un jeton dont la couleur serait y aurait deux directions possibles. L'indéterminisme est soulevé à la figure (b) en retirant une marque au domaine <a>. Il ne reste plus qu'une seule direction possible.

### III . 2 . 2 . 1 - Fonctions de gestion des domaines de couleur

Quatre fonctions importantes ont été développées pour pouvoir modifier facilement et automatiquement les domaines de couleurs par l'utilisateur.

La fonction "**mise_à_jour**" permet de mettre à jour un domaine de couleurs sur un automate.

La fonction "**lire_domaine**" permet d'aller lire le contenu d'un domaine de couleurs.

Les fonctions "**enlève_couleur**" et "**ajout_couleur**" permettent de modifier une gamme :

- La fonction "enlève_couleur" enlève un ou plusieurs codes sur une partie des domaines de couleurs,
- La fonction "ajout_couleur" ajoute un ou plusieurs codes sur une partie des domaines de couleurs.

Ces fonctions sont utilisées soit dans le cadre d'une reprise à chaud, pour mettre à jour le module de supervision, soit dans le cadre du reprise à froid pour initialiser le procédé.

Par exemple **en marche manuelle**, l'utilisateur peut être amené, en cas de reprise, à initialiser et à configurer des données : dans le cas d'une reprise à froid, il s'agit d'initialiser complètement la base de données automates. Dans le cas d'une reprise à chaud, il s'agit d'initialiser la base de données du gestionnaire par la lecture exhaustive des mémoires automates (nous utilisons la possibilité de la reprise directe des automates, qui n'est pas possible avec un ordinateur) afin d'avoir une image cohérente et à jour du procédé.

Il est possible de modifier alors les mémoires automates on_line, par l'appel aux fonctions. De plus, elles sont accessibles par menus ou utilisées par différents programmes.

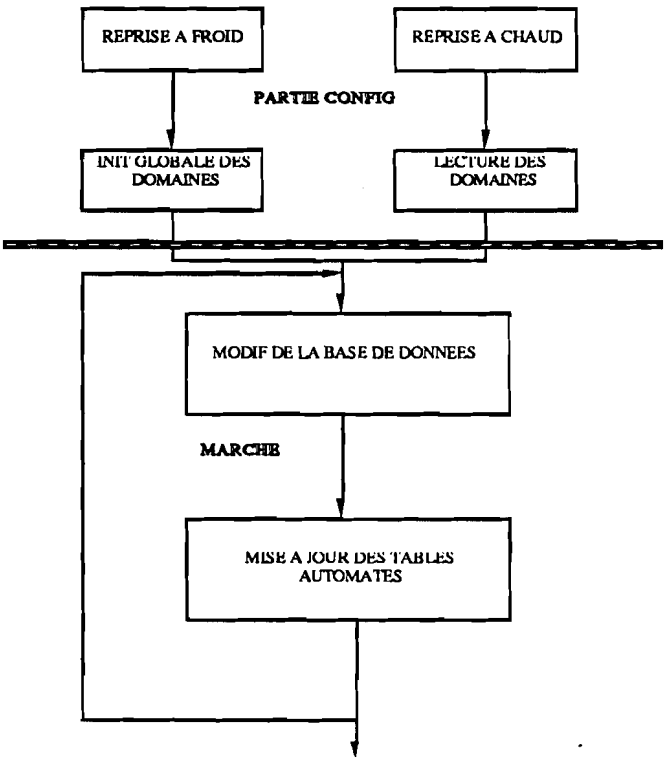



fig 14 : marche manuelle

### III . 2 . 2 . 2 - Exemple de reconfiguration

Nous allons présenter un exemple de reconfiguration du graphe de commande pour intégrer, un changement de mode de marche d'une machine, par exemple une panne du tour.

Nous nous limiterons au graphe de commande correspondant aux transferts d'une pièce vers le tour et venant du tour, qui est décrit ci-après :

Dans le cas où le poste P2 est libre (place NOP marquée) et qu'il y a une pièce présente en file d'attente (P2/PROD non vide), une palette est alors dirigée vers le poste P2. Si la palette contient une pièce à tourner, celle-ci est alors dépalettisée vers le tour (marquage de la place P2->T/DEPAL). Si la palette est vide, elle doit demeurer jusqu'à ce qu'une pièce provenant du tour soit palettisée (marquage de la place T->P2/WAIT), sinon elle est dirigée vers le poste suivant P4 (marquage de la place P2/REQ).

Les domaines de couleurs correspondants sont décrits sur les arcs.

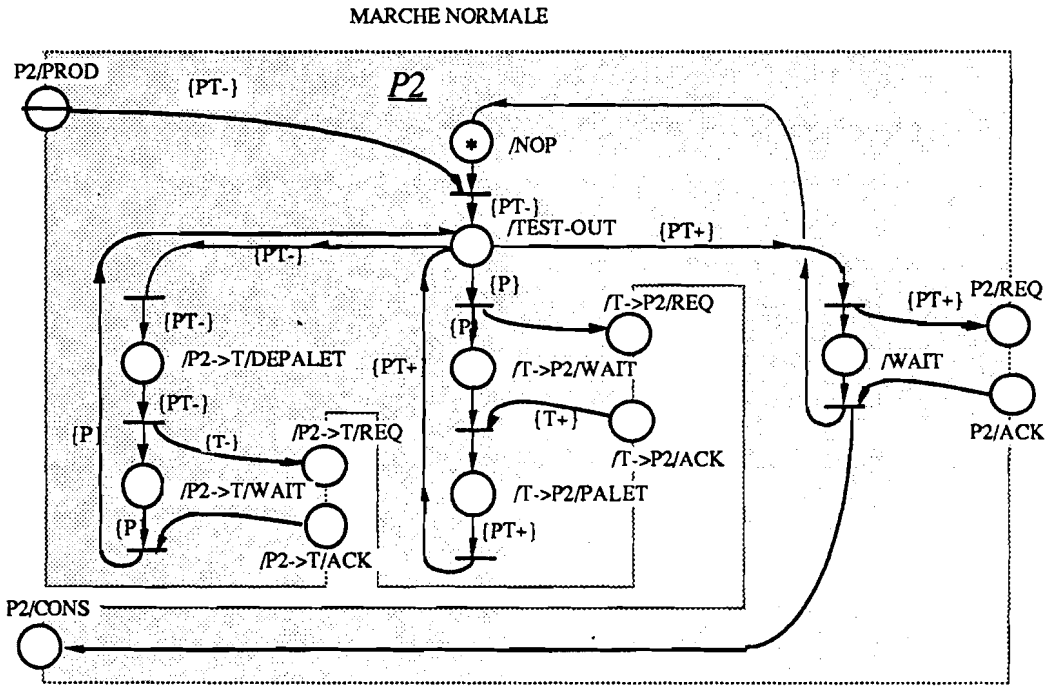



fig 15 : processus P2

Si le tour devient inutilisable, le gestionnaire doit alors modifier le graphe de commande de façon à empêcher le transfert d'une pièce vers le tour. Les palettes, vides ou non, sont alors directement dirigées vers le poste suivant P4.

Ceci se traduit par la modification dynamique des domaines de couleurs. Les couleurs (pT-) correspondant au transfert des pièces vers le tour sont retirées et les domaines correspondants au transfert vers le poste suivant incluent ces couleurs. De plus le graphe traduit le fait qu'il est toujours possible de sortir la pièce du tour pour la palettiser, mais dans un état brut.

Les domaines modifiés sont signalés par des flèches.

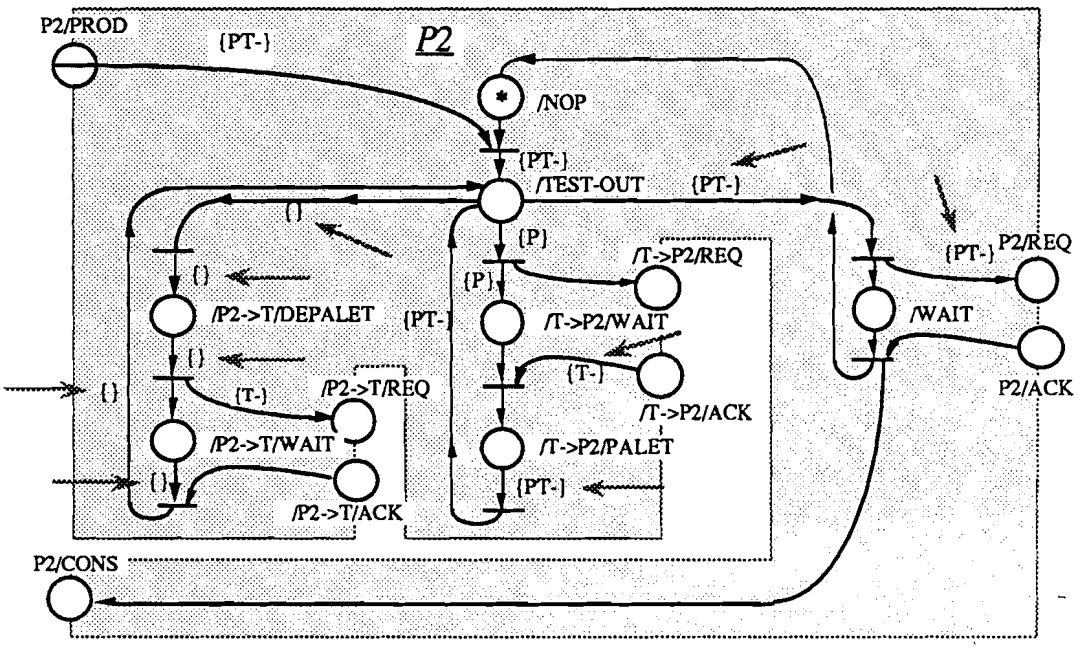



fig 16 : processus P2 après reconfiguration

### III . 2 . 3 - Mécanisme de mise en correspondance du Grafcet

Une autre fonction de l'interface est la mise en correspondance du graphe de commande en cas d'initialisation.

A la figure suivante, l'étape TEST_OUT est marquée pour symboliser la présence d'une pièce au poste correspondant. Le graphe évolue artificiellement pour avoir un état correspondant à celui du procédé.

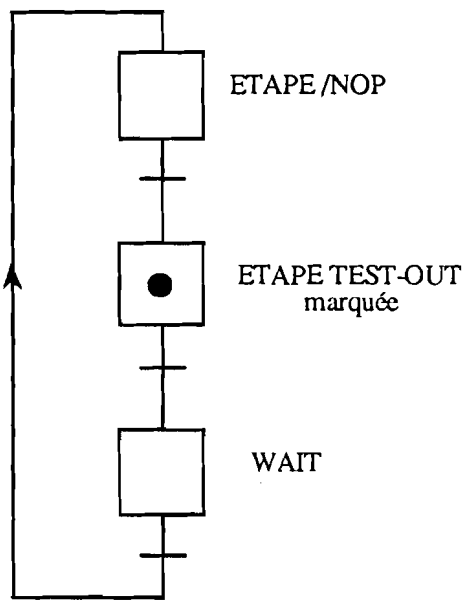



fig 17 : exemple de configuration

Chaque file d'attente à un poste est représentée par une variable de type FIFO au niveau de la mémoire des automates. Au moment de l'initialisation du système et donc du Grafcet, il est rangé autant de pièces qu'il y en a de présentes dans l'atelier. L'utilisateur propose le n° du poste et les types des pièces présentes à ce poste. Le logiciel va alors communiquer ces informations aux automates où un programme ira ensuite empiler ces différentes valeurs dans les FIFO.

### III . 2 . 3 . 1 - Exemple d'application

Considérons le démarrage du convoyeur1. Trois pièces palettisées sont disposées sur le convoyeur au niveau du poste P1. L'utilisateur doit alors configurer le graphe de commande en mode interactif.

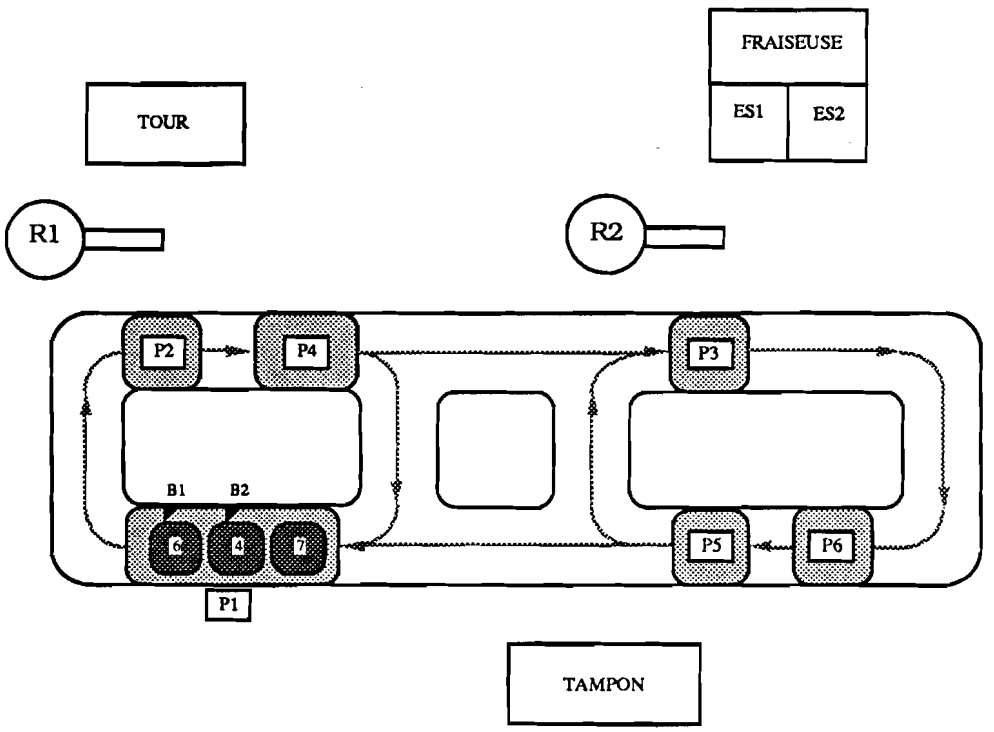



fig 18 : schéma du convoyeur1

La gestion du poste P1 est décrite sur le schéma suivant : nous avons une étape /NOP qui correspond à l'absence de pièces sur le poste, une étape TEST-OUT correspondant à la présence d'une palette, une étape WAIT correspondant à la libération du poste.

Nous avons représenté la variable FIFO associée, qui au départ ne contient aucun élément, de même que la variable utilisée pour contenir le type de la pièce située au niveau du poste P1 : W1.

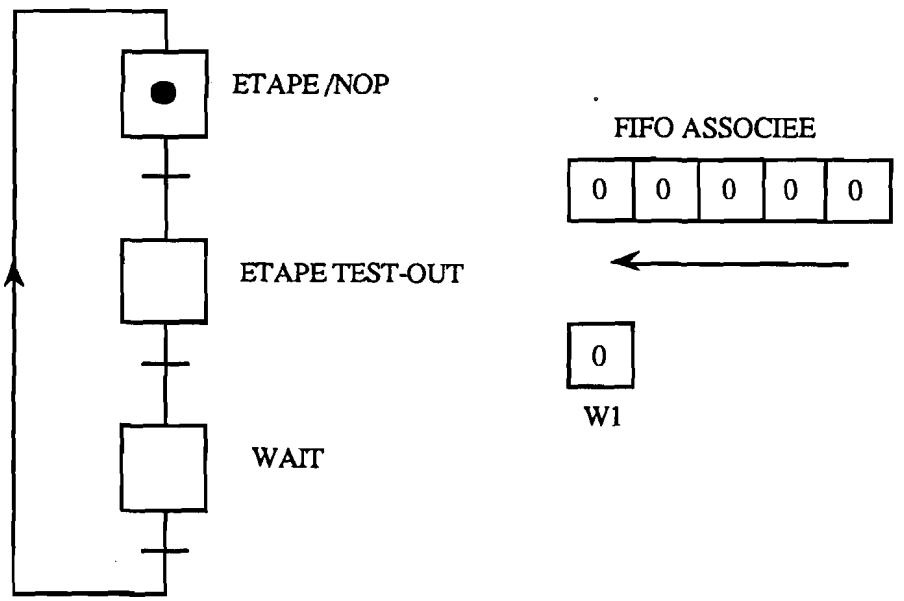



fig 19 : Grafcet utilisé dans la gestion du poste P1 avant configuration

La configuration est effectuée (fig 20).

Le Grafcet est modifié par l'activation de l'étape TEST-OUT et la désactivation de l'étape /NOP, ce qui correspond à la présence d'une pièce au niveau du poste P1.

Les variables sont mises à jour : W1 contient le type de la pièce se situant au niveau de la butée 1, de type 6 et la FIFO contient le type des deux pièces suivantes, 4 et 7.

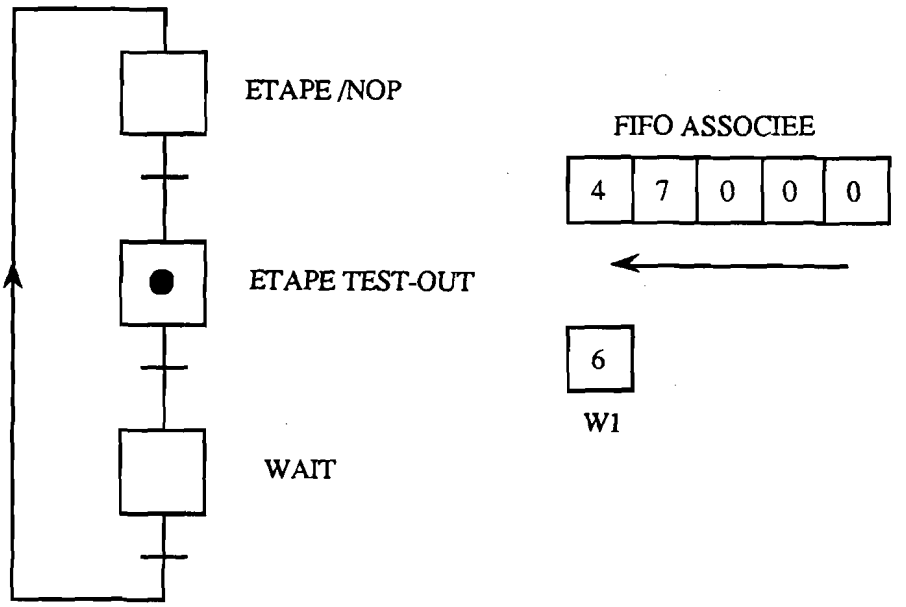



fig 20 : Grafcet utilisé dans la gestion du poste P1 après configuration

### III . 2 . 4 - Base de données utilisées

La base de données est liée à la configuration du système. Elle prend en compte les caractéristiques des automates programmables et des postes d'attente sur le convoyeur.

Les informations relatives aux A.P.I sont les commandes système qui permettent d'accéder aux modes de marche des automates.

Ces données sont stockées sous forme de P_LIST (liste de propriétés) qui permettent d'attribuer à chaque variable une série de propriétés.

L'éditeur permet de modifier cette base de données pour la configurer.

Nous avons donc, une spécification pour chaque poste et pour chaque domaine de couleurs significatif.

#### Cas d'un poste

Nous avons retenu comme caractères, le nombre maximal de pièces par poste, les variables (bits et mots) utilisées sur les automates pour la configuration et par des variables de commande associées, qui permettent sa manipulation. Prenons, par exemple, la gestion d'une FIFO interne à un automate.

Sa structure P_LIST en Le_Lisp est décrite ainsi :

```
(P_LIST 'NOM_FIFO '(
   NATURE          FIFO
   NOM_PROC        "nom du processus associé"
   LONGUEUR        "taille de la FIFO"
   MOT_ASSOC       "mot automate symbolisant le nombre de places vides "
   MOT_COM         "mot qui réceptionne la valeur stockée dans la FIFO"
   BIT_COM         "commande l'envoi d'une pièce dans la FIFO"
   TYPE            "liste des éléments"
))
```

#### Cas d'une zone de transfert

Les zones de transfert sont en général, à l'origine des indéterminismes directionnels, résolus par l'utilisation de domaines de couleurs dont la structure de données est caractérisée par :

- l'adresse de la zone mémoire sur l'automate réservée au tableau associé,
- les variables utilisées pour la configuration des tableaux,

- les couleurs initiales retenues par défaut.

Sa structure P_LIST en Le_Lisp est décrite ainsi :

```
(P_LIST 'nom_dom '(
NATURE          DOMAINE
NOM_DOM         "nom du domaine auquel se réfère la P_LIST"
ADR_DÉPART      "adresse de départ dans la table"
LG_DYN          "nb d'éléments en cours dans le domaine"
LG_STAT         "nb d'éléments maximum"
TYPE            "liste des éléments"
))
```

Par exemple :

```
(P_LIST 'P2->TOUR '(
NATURE          DOMAINE
NOM_DOM         P2->TOUR
ADR_DÉPART      W200_0
LG_DYN          3
LG_STAT         20
TYPE            (t-, f+t-, t-t-)
))
```

### III . 2 . 5 - Cycle d'écriture sur les automates

La modification des données et en particulier le paramètrage du graphe de commande, pose un problème de synchronisation et de gestion des transitoires. Il n'est pas possible de modifier la configuration en cours de fonctionnement sans prendre de risque : il s'agit de porter une attention particulière au moment où elle est envisagée. En effet, il n'est pas possible d'écrire ou de modifier un domaine de couleurs simultanément au moment où il est consulté. Ce risque est augmenté par le fait qu'une configuration nécessite beaucoup de temps vis à vis de celui nécessaire au cycle automate.

La solution retenue est de bloquer l'évolution du processus associé au domaine de couleur modifié pendant le temps de l'écriture ou de la lecture du domaine. Des variables de type bit, sont à cet effet utilisées. Le principe est d'adjoindre à chaque partie de graphe de commande associée à la gestion d'un poste, une variable de gel/dégel. Celle-ci est systématiquement testée lors de l'évaluation des conditions de chacune des transitions de cette portion de graphe. Si le

bit est à 1, le déclenchement de la transition est possible, sinon (bit à 0) il est bloqué. Cette technique, simple, mais lourde, permet de geler une partie du graphe, fonction impossible à réaliser par l'utilisation des "bits système".

### III . 2 . 5 . 1 - Exemple de configuration d'une FIFO

Une variable de type FIFO mémorise le type des pièces en attente à l'entrée d'un poste. Si le poste se libère, la première pièce en attente y est alors acheminée (dans ce cas, dépilement de la marque associée). Si une pièce vient se positionner en attente, la marque associée est alors empilée dans la FIFO.

Dans le cas d'une reprise à chaud de la production, il s'avère nécessaire de mettre en conformité le graphe de commande et sa structure de données associées vis à vis de l'état de l'unité de production. Les pièces qui sont donc physiquement en attente devant un poste de production sont ainsi prises en compte par une mise à jour de la FIFO associée.

Nous décrivons le principe d'écriture : le cycle de configuration débute par le gel du processus dédié à la gestion du poste, par l'activation du bit associé. Un bit commande l'écriture dans une FIFO. Il est activé pour le rangement de la valeur correspondant au type de la pièce. Ce procédé est réalisé autant de fois qu'il y a de pièces présentes dans la file d'attente. Le processus est alors dégelé (mise à 0 du bit associé). Dans ce cas, le principe de gel/dégel de la partie commande n'est pas utile car effectué en phase préparatoire mais elle le serait dans le cas d'une configuration on_line de la commande : l'utilisateur découvre qu'une pièce palettisée est défectueuse : il va alors retirer cette pièce et modifier en conséquence le graphe de coordination.

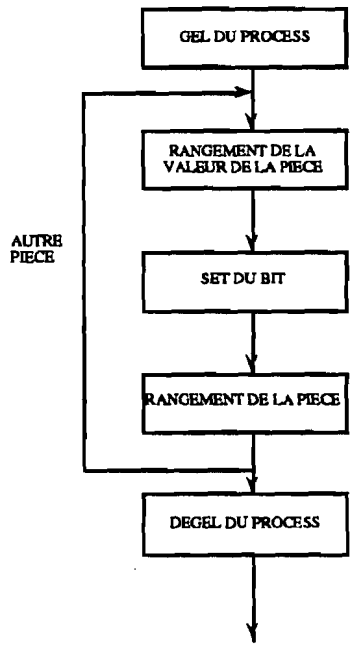



fig 21 : configuration d'une FIFO

### Conclusion

Nous avons donc montré comment nous utilisons la puissance de notre modèle de commande pour en effectuer une configuration dynamique et un paramètrage on_line de la commande. La difficulté résidait dans l'accessibilité aux données nécessaires, mais qui a été résolue par le développement de l'Interface_utilisateur. Celle-ci permet un fonctionnement manuel et interactif avec l'utilisateur et s'intègre dans l'ensemble des modules de supervision du Niveau Hiérarchique dont nous allons présenter le module principal, le gestionnaire de modes de marche.

## IV - PRESENTATION DU GESTIONNAIRE

Nous avons présenté dans le chapitre II, le cas de l'atelier de l'EC Lille comme exemple de notre démarche.

Nous intégrons donc ces résultats dans la conception du système de pilotage, avec l'exploitation de la base de connaissances élaborée à partir de la modélisation.

Il reste à préciser les actions susceptibles d'être générées vers le procédé, à partir du gestionnaire.

## IV . 1 - Description de la marche générale du gestionnaire

Les objectifs couverts par le gestionnaire sont de maintenir un taux de production régulier de l'atelier, par l'optimisation de ses différents modes de marche. L'exploitation de la modélisation présentée au chapitre II, conduit à l'élaboration d'un prototype destiné à la supervision de l'atelier de l'EC Lille qui permet d'automatiser complétement les opérations inhérentes aux changements de modes de marche et à leurs transitoires. De plus, il assiste et remplace l'utilisateur dans sa tâche de pilotage avec l'idée de réduire son temps de réflexion pour les prises de décision et de lui permettre d'avoir une action sûre et efficace dans le suivi et la commande des machines.

L'outil retenu est de type système expert : il permet simultanément de répondre à ces spécifications et d'intégrer aisément la modélisation présentée auparavant. Son fonctionnement permet de distinguer deux phases distinctes de raisonnement, qui correspondent en fait, à celles que respecte l'utilisateur lorsqu'il se trouve confronté au problème d'une prise de décision lors du pilotage : une phase d'analyse du système de production afin de déterminer les éventuels problèmes de fonctionnement, qui aboutit à la formulation de solutions possibles, suivie d'une phase de résolution et de test de ces solutions qui aboutit à l'émission d'actions effectives.

**Un mode automatique** est possible où le gestionnaire, complètement autonome, se comporte comme un système en boucle fermée. D'après l'état du process, il va corriger les écarts de comportement de l'atelier afin de le remettre dans un état cohérent. Les informations retenues concernent l'état des machines et de leurs éléments ; une mise à jour de la base de faits dynamique est effectuée par une scrutation automatique (mode espion) du procédé avant chaque inférence. Le gestionnaire en déduit automatiquement une série d'ordres à proposer pour la régulation du process.

De plus, **en mode interactif ou manuel**, l'utilisateur est susceptible d'apporter de nouvelles informations pour enrichir la base de connaissances et peut, parallèlement au gestionnaire, proposer une ou plusieurs demandes de modifications de modes de marche, d'après sa connaissance personnelle de l'état des machines. Le gestionnaire, dans ce cas, en vérifie alors la cohérence par rapport à l'état du procédé et des machines (il s'agit de ne pas envoyer de commande risquant d'occasionner des problèmes de fonctionnement) puis émet les actions correspondantes licites.

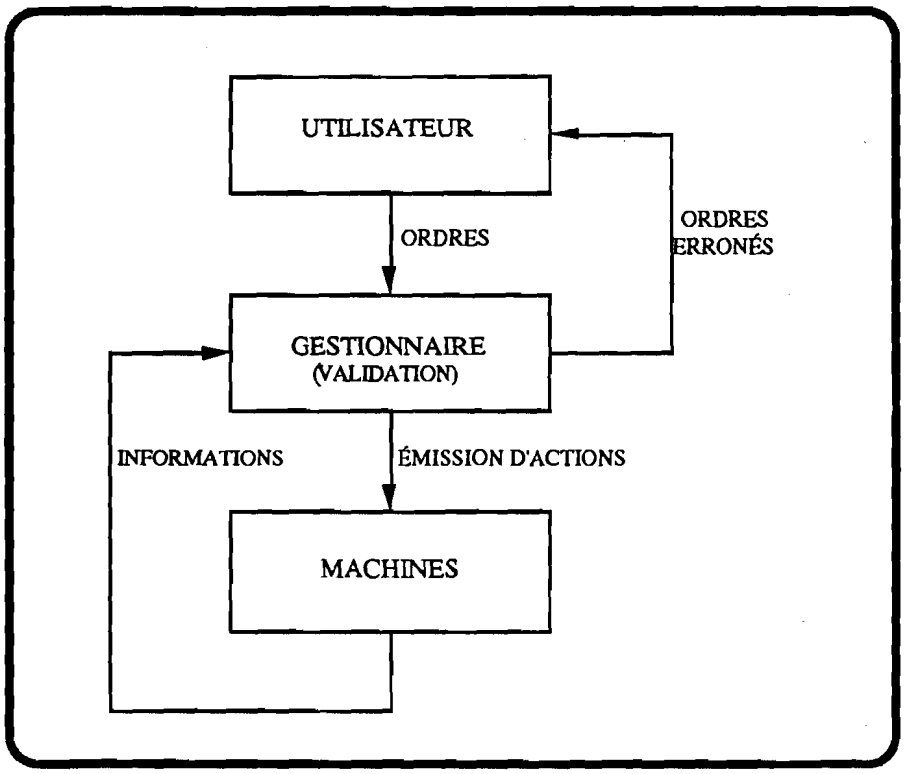



fig 22 : fonctionnement général du gestionnaire

Le prototype a été développé en Le_Lisp de l'INRIA. Nous nous sommes orientés vers ce langage, de façon essentiellement à conserver une démarche homogène vis à vis des travaux antérieurs du laboratoire. Les avantages de ce langage sont sa forte interactivité (langage interprété) et sa capacité à ne pas différencier les données des programmes. Cependant, il est difficilement adapté aux tâches de pilotage en temps réel, de par son manque d'ouverture vers les fonctions des systèmes d'exploitation.

Nous utilisons plusieurs moteurs d'inférence. En effet, nous essayons de reproduire les différentes phases du raisonnement humain. Face à un problème à résoudre, l'être humain a une démarche structurée qui débute par une phase d'analyse de ce problème, la détection des solutions possibles et enfin la résolution effective du problème. Par analogie, nous avons conçu notre système de façon équivalente. Nous avons un moteur d'inférence en chaînage avant et un moteur d'inférence en chaînage arrière. Ils reproduisent le même schéma de raisonnement.

Ce sont deux moteurs d'ordre $0^+$. Le choix de l'ordre 0+ se caractérise par l'absence de variables dans la modélisation des connaissances. Les données restant très spécifiques aux situations modélisées, il n'était pas nécessaire de généraliser les connaissances. Les règles sont donc spécifiques et d'ailleurs plus faciles de compréhension. La spécificité du matériel modélisé autorise d'ailleurs, ce niveau d'abstraction peu élevé.

Nous allons décrire les deux types de démarches :

- celle de la génération automatique d'ordres,
- celle de la résolution de ces ordres.

### IV . 1 . 1 - Génération automatique d'ordres

Cette phase consiste à aider l'utilisateur dans sa démarche d'analyse du fonctionnement général, en proposant à un instant donné, le meilleur ordre possible susceptible de conserver l'état de l'atelier dans un état toujours cohérent et correct par un rebouclage permanent sur le procédé.

Le principe réside sur la comparaison entre **l'état antérieur** (image conservée au niveau de la base de données depuis la dernière phase d'analyse) et **l'état effectif** (état réel ou calculé) à partir de l'ensemble des faits. Le lancement d'une inférence n'est effectué que dans le cas d'une ou plusieurs différences entre ces deux états, concernant au moins un fait.

Dans ce cas, le cycle de marche débute par une phase d'analyse de l'état du process pour détecter les incohérences d'états entre machines et à partir de là, générer un but à résoudre susceptible de remédier à cette situation. Dans le cas de multiples incohérences d'états, le système est capable de remonter jusqu'à la détection du problème le plus important. Cette analyse repose sur **une déduction en chaînage avant.**

**Un ordre** sera donc un changement direct de mode de marche d'une machine. Supposons qu'une information indique que l'outil d'une machine de production soit brisé. L'ordre qui sera alors émis, correspondra à terme, en un arrêt sous défaut dans l'optique d'un traitement ou d'une procédure de dégagement.

La situation classique est l'occurrence simultanée de plusieurs alarmes [OBS 89]. L'utilisateur se trouve confronté au problème de déterminer celles à prendre en compte et à privilégier, par ordre d'importance, et qui imposent directement une action immédiate de correction. Celles qui ont seulement une conséquence de ces défauts de base, donc moins importantes, sont éliminées. La déduction en chaînage avant nous permet de déterminer le plus important problème. Sa résolution permettra de résoudre des sous_problèmes moins importants et conséquents de ce problème initial.

Si l'utilisateur désire un changement de mode de marche d'une machine, il propose alors l'état dans lequel il désire la positionner. Le gestionnaire va alors rechercher le meilleur ordre possible qui inclura la demande, tout en tenant compte des contraintes de fonctionnement. L'inférence en chaînage avant permet de connaître l'ensemble des machines concernées par la demande de changement de modes de marche avec ensuite la génération d'un ordre approprié.

Par exemple, dans le cadre le l'atelier de l'EC Lille, le TOUR et le robot R1, liés par une contrainte de coopération, sont dans l'état "marche normale". L'utilisateur désire arrêter le TOUR et propose alors cette information. Elle sera considérée comme le nouvel état du TOUR par le gestionnaire. Il va alors calculer successivement le nouvel état de M1, MV1... L'état antérieur de M1 étant "marche normale" et son état calculé étant "tension-on", un ordre de correction sera déduit par le gestionnaire qui est (?arrêt M1), qui aura pour effet d'arrêter le TOUR et R1, en accord avec la contrainte de coopération CC1. M1 est la machine de plus haut niveau concernée par la demande de changement de mode de marche du TOUR.

### IV . 1 . 2 - Phase de résolution des ordres

La phase d'analyse terminée, celle de la résolution commence et repose sur une inférence en **chaînage arrière avec un raisonnement de type planification**. Les demandes d'ordres ont donc été, soit générées automatiquement, soit proposées par l'utilisateur. Par analogie avec la planification, nous identifions la demande d'ordre comme un but à résoudre.

Il y a en premier lieu, un classement des buts à résoudre suivant leur niveau d'importance. Une phase de restriction débute alors par l'optimisation de la base de connaissances avec l'utilisation de méta-règles. Cette restriction utilise un moteur en chaînage avant.

Ensuite, le système poursuit par la résolution du problème le plus important, avec une technique de génération de plans. Lorsqu'un des buts est complètement résolu, les actions associées sont alors déclenchées et le but suivant est ensuite traité. La cohérence des buts est garantie par le fait qu'ils sont générés lors de la même inférence de déduction dont le raisonnement monotone ne peut remettre en cause des faits préalablement établis.

La validité d'un ordre généré automatiquement est garantie par le fait que l'ordre est déduit en tenant compte des états de chacune des machines impliquées par sa résolution.

De plus, la validité d'un ordre est réalisée implicitement lors de la résolution. Si l'inférence en chaînage arrière aboutit systématiquement à une impasse dans la satisfaction du but, cela signifie qu'il n'est pas valide. Dans ce cas, aucune action n'est émise. La validité d'un ordre signifie qu'il est cohérent vis à vis des contraintes de fonctionnement présentées au chapitre II.

En reprenant l'exemple précédent, si l'utilisateur désire remettre en marche la machine, l'ordre sera refusé tant qu'une information précisant la correction de l'outil n'aura pas été reçue.

Prenons le cas du TOUR et du robot AFMA de l'atelier de l'EC Lille. Supposons que le robot AFMA de chargement/déchargement tombe en panne. Le tour devient alors inutilisable

dans un fonctionnement automatique de l'atelier. En tenant compte de la contrainte de coopération qui traduit le fonctionnement indissociable de ces deux machines, le gestionnaire va alors proposer d'émettre vers le tour un ordre de repli qui consistera à son arrêt.

Trois systèmes à base de règles sont donc utilisés, chacun d'eux correspondant à une fonction bien précise et ayant un moteur d'inférence bien spécifique. Le schéma suivant explicite la démarche générale et son séquencement.

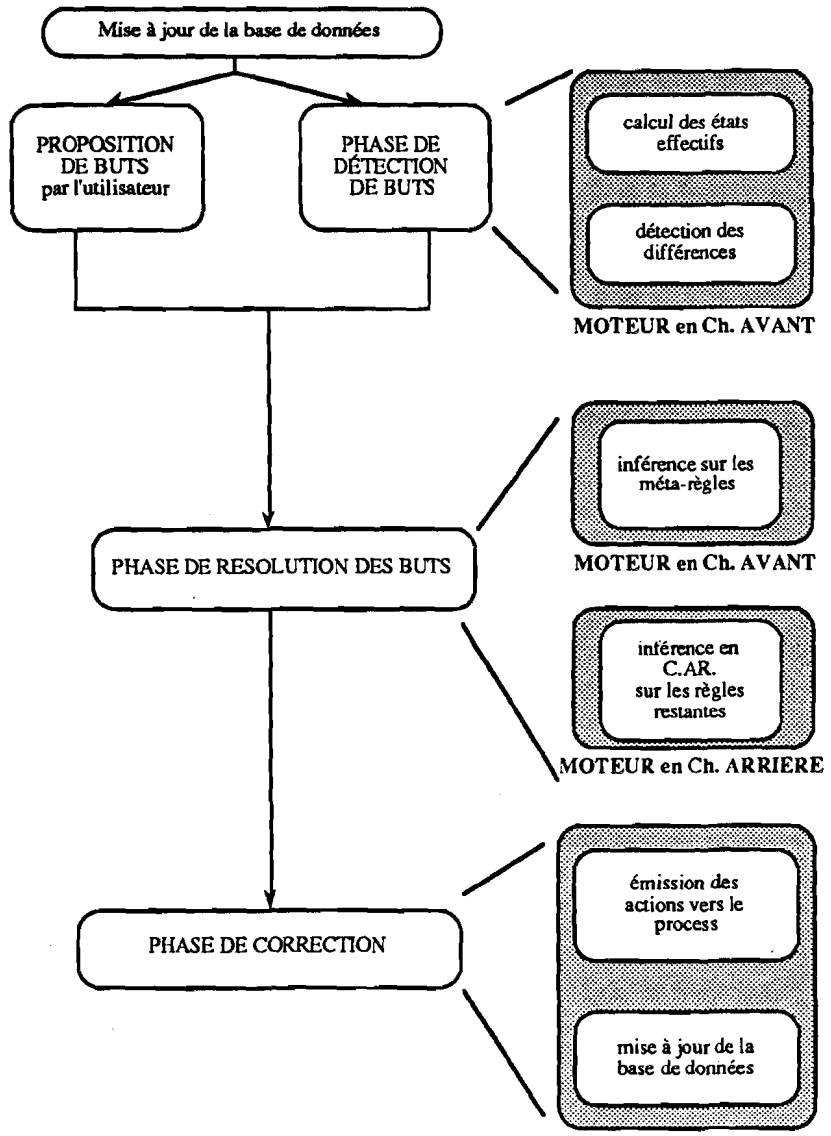



fig 23 : organigramme de fonctionnement

## IV . 2 - Base de données

Elle est constituée de faits et de règles. La base de faits est commune à tous les moteurs d'inférence qui utilisent, cependant des règles au format bien spécifique.

Un système d'édition permet de décrire la base de données utilisées par les moteurs d'inférence. Un système de menus successifs permet d'une part, de construire la représentation structurée, et d'autre part, d'éditer les règles manuellement. Une analyse syntaxique est effectuée systématiquement.

Une amélioration du gestionnaire serait la validation de la base de connaissance par une vérification de la complétude et de la cohérence.

### IV . 2 . 1 - Les faits

La base de faits peut être assimilée à la mémoire de travail. Elle est constituée de deux sortes de données :

- les faits permanents ou statiques,
- les faits spécifiques ou dynamiques.

**Les données statiques** sont constituées des descriptions des lieux physiques, avec le répertoire du matériel, fixé en général, associées aux descriptions des éléments importants des machines. Ceci concerne plus précisément les structures de graphes d'état et leurs relations établies au départ, lors de la modélisation.

**Les données dynamiques** sont des faits qui sont susceptibles d'évoluer. Le formalisme de description est important car il peut influer sur la compréhension de chacun des faits. De plus, cela met en cause la structure et la maintenance des connaissances. Ceci correspond par exemple, aux marquages des graphes d'état.

Notre base de faits contient les renseignements nécessaires à la marche du système expert concernant les machines. Elle est renseignée soit directement, par réception de données venant du process, soit indirectement par l'utilisateur qui peut informer le gestionnaire d'événements qui seraient par ailleurs indétectables à l'aide de capteurs.

Nous construisons une liste de propriétés par machine, ces propriétés étant :

**des données statiques :**

- le type de la machine, effective ou virtuelle,
- le niveau de la machine dans la structure hiérarchisée,

- le nom de ses sous-machines, dans le cas d'une machine virtuelle,
- le nom de ses éléments, dans le cas d'une machine effective,

**des données dynamiques :**

- l'état antérieur de la machine,
- l'état calculé ou effectif de la machine.

L'état antérieur est une image de l'état de la machine, dernier état obtenu lors des inférences.

Ces faits sont susceptibles d'être utilisés ou modifiés tout au long du cycle d'inférence :

- lors de la lecture des entrées -> modification de l'état effectif,
- après la phase de déduction -> modification de l'état calculé,
- après la résolution de buts -> modification de l'état antérieur.

### IV . 2 . 2 - La base de connaissance

Elle est constituée de règles de production qui sont les connaissances opératoires et qui permettront de tirer des conclusions, et de méta-règles.

#### IV . 2 . 2 . 1 - Les règles

Nous avons pour chaque moteur, un type bien défini de règles, spécifiques syntaxiquement et fonctionnellement :

- des règles pour le chaînage avant,
- des règles pour le chaînage arrière.

Deux composantes interviennent dans l'élaboration de ces règles : les actions et les buts.

##### IV . 2 . 2 . 1 . 1 - Les actions

Les actions sont les opérations qui permettent de passer d'un état à un autre. Elles sont déclenchées dans le cas d'un succès dans la résolution d'un but.

Ces actions sont de natures diverses :

- il peut s'agir d'actions effectives de changements d'état sur les machines (mise en arrêt ou mise en marche du tour),
- cela peut être des actions consistant à lancer un processus indépendant ou un logiciel suivant l'état des machines (logiciel de coordination entre machines). Ces actions peuvent être des opérations de gel, de configuration, d'arrêt ou de démarrage de processus dont les fonctionnements sont associés à des tâches particulières avec éventuellement un passage de paramètres lors de l'appel de la fonction,
- il peut s'agir d'actions de modifications de la base de données,
- il peut s'agir d'actions de configuration automatique de la partie commande avec en particulier son initialisation en fonction de l'état du process et de sa mise à jour, en fonction des différents modes de marche demandés.

Les actions sont spécifiques aux machines et aux protocoles de communication, et sont configurables par l'usager lors de l'implantation effective du système de supervision. Le module d'interface avec le procédé nous permet de définir explicitement les actions en langage Le_Lisp.

Toutes les actions, concernant le passage d'un état à un autre, sont dépendantes par le fait qu'elles doivent être générées dans le même cycle d'inférence. L'arrêt effectif d'une machine nous amène à prendre en compte de façon parallèle :

- une action effective d'arrêt,
- une reconfiguration de la partie commande en vue d'intégrer le changement de mode de marche, par l'adaptation de la coloration (technique de gestion dynamique des domaines de couleurs), en retirant des couleurs concernant cette machine.

Nous regroupons ces actions en **une seule Macro_action**. Chaque macro_action regroupe plusieurs actions, celles-ci étant classées par ordre de priorité. Les macro-actions, déclenchées lors du même cycle d'inférence, sont déclenchées en séquence dans l'ordre dans lequel elles apparaissent lors de la résolution.

#### IV . 2 . 2 . 1 . 2 - Exemple de gestion de processus

**Cas des automates TSX 47** : au fonctionnement des automates est associé celui du logiciel de coordination. Son exploitation correcte serait d'associer son lancement à la mise en marche des automates. Ainsi, nous intégrons le lancement de ce logiciel dans l'action

correspondant à la mise en marche des automates, et son arrêt dans l'action correspondant à l'arrêt des automates. De plus, il est possible de paramètrer le lancement en partitionnant la base de règles du module de coordination : nous distinguons celles destinées à l'un ou l'autre des automates. Si un seul automate fonctionne, une seule partie de la base de règles sera alors utilisée. Ceci permet d'optimiser le temps de cycle du moteur d'inférence du module de coordination.

#### IV . 2 . 2 . 1 . 3 - Exemple de macro-action

Les macro-actions sont des combinaisons d'actions qui doivent être émises simultanément lors du même cycle d'inférence. Elles incluent toutes une émission d'ordre vers le procédé. Pour l'instant, l'absence de liaison directe GPX/NUM nécessite la mise en oeuvre d'un programme intermédiaire sur les TSX. L'installation d'une liaison UNI-TELWAY permettra de simplifier ce dispositif avec l'utilisation de requêtes prédéfinies.

Prenons comme exemple, l'arrêt du tour. Cela combine :

- l'arrêt effectif du tour (requête UNI-TELWAY N° 25H) [MAN 89],
- la mise à jour des domaines de couleurs (retrait et ajouts des marques t- afin de ne pas orienter les pièces destinées au tournage vers le tour),
- la mise à jour de la base de données, au niveau de la P_LIST,
- éventuellement l'affichage d'un message à l'utilisateur,
- la modification du synoptique.

#### IV . 2 . 2 . 1 . 4 - Les buts

Ce sont en fait classiquement des demandes d'ordres qui peuvent être transmises au système de production, par l'utilisateur ou le gestionnaire lors de sa phase de déduction :

- demande de mise en marche/arrêt d'une machine ou autre changement d'état,
- demande de modification du paramètrage de la commande traduisant un changement de stratégie en privilégiant telle gamme de production ou telle machine,
- demande de modification de la marche normale : passage en mode automatique, manuelle ou pas à pas,
- proposition de solutions pour la maintenance, en mode manuel.

**Buts ou ordres possibles**

Nous allons lister les buts susceptibles d'être déduits par le gestionnaire ou proposés par l'utilisateur. Un but étant une demande de changement d'état d'une machine, chacun d'eux correspond à un arc au niveau des graphes d'état, et il y a autant de buts que d'arcs. Un changement d'état conduit à l'émission d'une macro-action.

Chaque arc du graphe d'état est donc associé à la fois à une action de changement d'état et à un but :

Nous présentons la liste exhaustive des buts possibles :

| ARC | ACTION ASSOCIEE | BUT ASSOCIE |
|---|---|---|
| 0->1 : | mettre_sous_tension | ?mise_sous_tension |
| 1->0 : | arrêter_tension | ?arrêt_tension |
| 1->1 : | vérifier_état | ?vérif_état |
| 1->2 : | initialiser | ?initialisation |
| 1->3 : | passer_en_déf | ?passage_déf |
| 1->4 : | mettre_en_marche | ?mise_en_marche |
| 2->0 : | arrêter_tension_après_init | ?arrêt_tension_après_init |
| 2->3 : | passer_en_déf_après_init | ?passage_déf_après_init |
| 2->4 : | mettre_en_marche_après_init | ?mise_en_marche_après_init |
| 3->0 : | arrêter_tension_sous_déf | ?arrêt_tension_sous_déf |
| 3->1 : | traiter_déf | ?trait_déf |
| 4->1 : | arrêter | ?arrêt |
| 4->3 | arrêter_direct_sous_déf | ?arrêt_direct_sous_déf |
| 4->5 : | passer_en_marche_sous_déf | ?passage_marche_sous_déf |
| 5->3 : | arrêter_machine_sous_déf | ?arrêt_machine_sous_déf |

Chacune des actions est en fait une macro-action qui inclue une série d'actions qui est à définir au départ.

**IV . 2 . 2 . 1 . 5 -** Formalisme des règles pour le chaînage avant

Ces règles ont un formalisme de règles de production dont la structure est :

- un identificateur de règle,
- une syntaxe de type SI {prémisses} ALORS {conséquents}.

Cette syntaxe signifie que, si toutes les prémisses d'une règle sont vérifiées, alors ses conséquents sont déclenchés.

La partie "prémisses" constitue la condition logique de déclenchement de la règle. Le résultat de ces conditions dépend de l'état de la base de données. La partie "conséquents" ou "actions" modifie la base de faits et l'état du procédé.

Nous avons défini deux opérateurs utilisables dans la partie prémisses :

- un opérateur qui teste si l'état d'une des sous-machines ou d'un des éléments d'une machine a évolué, **test_état,**
- un opérateur qui effectue une comparaison de l'état antérieur et de l'état effectif pour une machine, **éval_état.**

Nous avons défini deux opérateurs utilisables dans la partie conséquents :

- un opérateur de calcul d'état d'une machine d'après l'état de ses sous-machines ou de ses éléments, **calcul_état,**
- un opérateur de génération de buts, **calcul_but.**

En utilisant ces opérateurs, deux types de règles sont possibles : un premier est destiné au calcul du nouvel état d'une machine dans le cas où un de ses éléments ou une de ses sous-machines a changé d'état ; le deuxième est destiné à l'évaluation d'un but dans le cas où le nouvel état est différent de l'état antérieur. Nous avons préféré bien séparer ces deux étapes afin de bien montrer la différence d'analyse.

Les règles sont structurées respectivement de la façon suivante :

(test_état machine) --> (calcul_état machine)

(éval_état machine) -> (calcul_but machine)

**Exemple de règle :** cas du TOUR

Deux règles sont proposées, R1 et R2 :

R1 : (test_état TOUR) --> (calcul_état TOUR)

R2 : (éval_état TOUR) -> (calcul_but TOUR)

La règle R1 signifie que si un des éléments du TOUR a changé d'état (test_état TOUR est vrai), alors le nouvel état de TOUR est calculé. La règle R2 signifie que si l'état antérieur et

l'état calculé de TOUR sont différents, un but est alors calculé. Cela correspond, par exemple, à la situation suivante : l'outil du TOUR se détériore, et le fonctionnement du TOUR n'est plus correct (passage en marche sous défaut). Il s'agit alors d'arrêter le TOUR afin d'effectuer des opérations de correction.

#### IV . 2 . 2 . 1 . 6 - Formalisme des règles pour le chaînage arrière

Les règles pour le chaînage arrière sont des règles qui permettent de décomposer un but en sous-buts ou de résoudre un but.

Leur syntaxe est particulière [FAR 87] : la partie gauche de ces règles constitue la partie "déclencheurs" de la règle, la partie droite la partie "prémisses".

**Les déclencheurs** sont constitués de deux sortes de faits :

- un but ou un problème à résoudre,
- des faits concernant l'état du système à vérifier.

**Remarque** : si nous appliquions strictement la définition du chaînage mixte, nous considérions ces règles, dédiées à un moteur d'inférence de ce type. En effet, la structure de ces règles montre que les déclencheurs invoquent à la fois des faits à résoudre et des faits résolus, correspondant à un raisonnement simultanément guidé par les faits et les buts, ce qui est la définition du chaînage mixte. Cependant, comme les règles sont toujours invoquées dans le même sens, et puisqu'elles ne comportent pas toujours de faits prouvés comme déclencheurs, notre système conduit à un raisonnement en chaînage arrière.

Dans chaque ensemble de déclencheurs, nous n'avons qu'un seul but à résoudre. Par contre, nous pouvons avoir un ou plusieurs faits à vérifier pour le déclenchement de la règle.

**Les prémisses** sont constituées de trois sortes de faits :

- un ou plusieurs sous-buts à résoudre,
- une ou plusieurs actions à déclencher,
- une contrainte à respecter et à établir.

La définition syntaxique des règles est donc de type (donnée sous la forme BACKUS-NAUR) :

| | | | | |
|---|---|---|---|---|
| <règle> | ::= | <déclencheurs> | ← | <conséquents> |
| <déclencheurs> | ::= | <but> \| <fait> | | |
| <but> | ::= | <opérateur><argument> | | |
| <conséquents> | ::= | <but> \| <action> \| <contrainte> | | |
| <action> | ::= | <opérateur><argument> | | |
| <contrainte> | ::= | <constante> | | |

Les opérateurs dans la partie but sont les différents buts possibles. Ceux dans la partie action sont les différentes actions possibles. les arguments sont des noms de machines.

La règle classiquement obtenue est :

(but_à_résoudre) (état_du_système) <- (sous-buts)
(contrainte)
(actions)

Cette règle signifie que pour résoudre le "but_à_résoudre", sachant que le système se trouve dans un état donné, il faut résoudre les sous-buts et vérifier la contrainte. Dans ce cas, les actions associées sont rangées dans une table en attente d'être générées.

**Exemple :**

(?arrêt TOUR) (état TOUR marche_normale) <- (arrêter TOUR)

L'arrêt du TOUR sera possible si il se trouve dans l'état de marche et sera effectif par la réalisation de l'action (arrêter TOUR).

Le fait prouvé (état_du_système) impose une contrainte supplémentaire sur le déclenchement de la règle et donc permet d'orienter plus rapidement la recherche (analogie avec la programmation sous contraintes). Nous associons de plus, à chaque règle, un niveau de priorité donné par l'utilisateur. Celui-ci peut ainsi favoriser telle règle par rapport à telle autre, en jouant sur les priorités.

### IV . 2 . 2 . 2 - Les méta-règles

Nous nous sommes rendu compte de l'importance du nombre de règles était très important, aussi avons nous cherché à définir une base de méta-règles. Celles-ci optimisent l'utilisation de la base de règles, par retrait ou ajout de règles. Dans notre cas, les méta-règles

permettent de retirer ou d'ajouter des règles suivant l'état des machines impliquées dans la résolution des buts. Leur formalisme est du type règle de production, décrit précédemment.

Des opérateurs ont été définis pour la gestion de la base de règles, avec en particulier des opérations de retraits de règles.

Nous décrirons comment nous obtenons ces règles et méta-règles, dans la suite du rapport.

### IV . 3 - Description des moteurs d'inférence

Nous allons décrire précisément le fonctionnement de chaque moteur d'inférence utilisé, avant et arrière.

#### IV . 3 . 1 - Déduction en chaînage avant

Ce moteur est destiné à l'étude des états des différentes machines du procédé. Parler d'ailleurs de moteur d'inférence peut s'apparenter à un abus de langage. En effet, l'inférence consiste seulement en un enchaînement de déductions qui permettent de calculer et d'analyser l'état global du système de production (phase des calculs des états des machines).

Les caractéristiques de ce moteur sont :

- **un fonctionnement en largeur** : il déclenche absolument toutes les règles dont les prémisses sont satisfaites, avant d'intégrer leurs conclusions. Il cherche à produire le maximum de faits à partir des informations connues. C'est le principe même d'une déduction "aveugle" qui n'est pas guidée par des heuristiques de recherche,
- **un régime irrévocable** : nous considérons que le déclenchement des règles est irrévocable sans remise en cause des conclusions,
- **un fonctionnement monotone** : nous gérons une liste de faits, mise à jour parallèlement à la base de faits, que nous utilisons lors du cycle d'inférence. Nous ajoutons dans la liste, initialement vide, des faits nouveaux afin de prendre en compte l'évolution de l'état du système modélisé, au fur et à mesure du déclenchement des règles. Comme un fait n'est jamais remis en cause ni retiré de la liste, ceci caractérise un fonctionnement monotone.

L'objectif de ce moteur est le calcul des différents états de chaque machine virtuelle du système afin de déterminer les nouveaux buts à partir de la configuration courante.

La mise à jour consiste à avoir une image exacte des machines et de leurs états dans la base de données. Certaines informations permettent une modification directe de la base de données (en particulier, des informations concernant des éléments physiques qui facilitent la connaissance de leurs états tels que les outils, automates...). Les informations concernant l'état des machines virtuelles ou de machines physiques ne permettant pas une connaissance directe de leurs états, ne peuvent être déduites que par le traitement d'autres données et sont acquises indirectement par le calcul d'état.

Lors de la mise à jour, nous faisons une acquisition systématique de toutes les données. En effet, le principe de communication nous oblige à cette organisation, mais le nombre relativement faible de variables utilisées autorise cette méthode, tout en demeurant néanmoins un facteur limitatif dans la rapidité de réponse.

Méthode de calcul d'état : par une démarche ascendante, les états des machines virtuelles sont calculés à partir des états des machines effectives. La démarche ascendante permet de calculer les états des machine de niveau supérieur et donc du système complet : c'est une propagation des informations de niveau bas.

Cette démarche ascendante d'évaluation correspond au fonctionnement en largeur. En effet, pour calculer l'état d'une machine de niveau N, il faut tout d'abord calculer celui de toutes les machines de niveau inférieur N-1. Le calcul est itératif de la base vers la racine, avec l'intégration des contraintes. Le calcul d'un nouvel état consiste à vérifier tout d'abord, pour chaque machine, si l'une de ses sous-machines ou l'un de ses éléments a changé d'état. Dans ce cas, le nouvel état de la machine est évalué par la méthode exposée au chapitre II. Nous obtenons ainsi l'état du système complet.

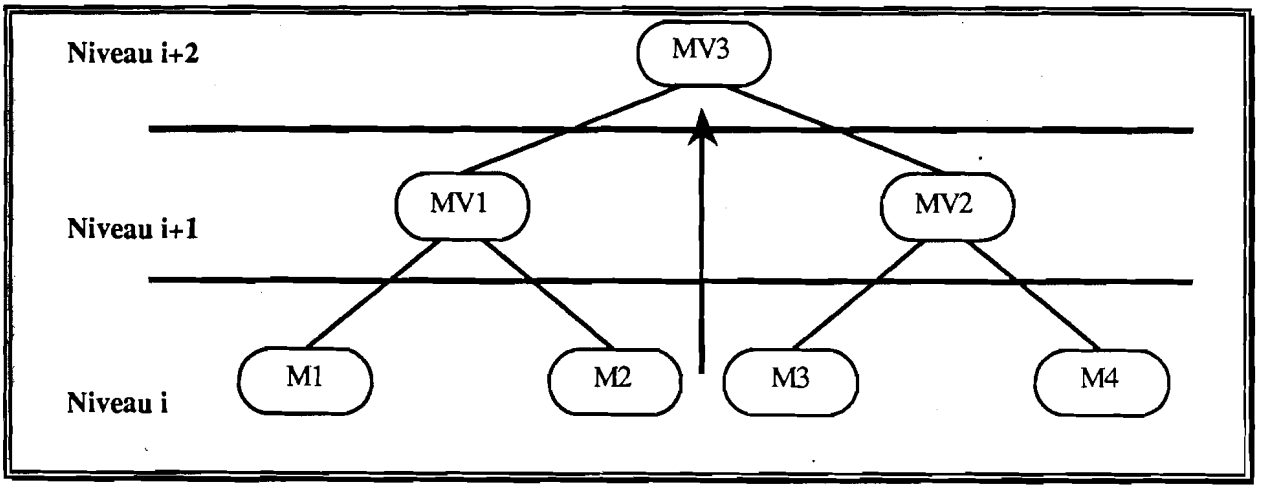



fig 24 : démarche ascendante

Les états des machines de niveau i+1 sont calculés à partir des états des machines de niveau i. De même, les états des machines de niveau i+2 sont calculés à partir des états des machines de niveau i+1.

$$ETAT(MV1) = F(ETAT(M1), ETAT(M2))$$

Génération de buts : l'élaboration d'un but repose tout d'abord sur la comparaison entre l'état antérieur et l'état calculé de chacune des machines virtuelles ou effective du système. Au moins une différence signifie que le système a évolué depuis la dernière inférence. Il y a donc une possibilité d'instabilité de fonctionnement au niveau du système de production par le fait que les états des machines ne sont pas forcément en accord et n'ont pas évolués conformément aux règles de fonctionnement imposées par les contraintes. Il faut alors nécessairement remettre le système complet dans une position stable. Des actions de correction seront alors émises. Pour cela, le système propose un ou plusieurs buts, de façon à ce que toutes les sous-machines soient dans des états compatibles et cohérents, en tenant compte des contraintes.

Le but proposé est déduit par la comparaison entre l'état antérieur et l'état calculé des machines de plus haut niveau.

L'idée est d'amener effectivement l'état du système complet, de l'état antérieur à l'état calculé par une série d'actions, ce qui aura pour effet de faire évoluer toutes les sous-machines de manière cohérente. Pour cela, il faut générer un but approprié. Deux situations sont possibles d'après la structure des graphes d'état :

- si l'état antérieur et l'état calculé sont directement liés par un arc, le but associé à cet arc est alors proposé,
- si les états ne sont pas directement liés, plusieurs buts sont possibles qui correspondent aux différents chemins possibles pour passer d'un état à un autre. Nous avons prédéterminé le choix du but en particularisant chaque situation. Nous retenons le but le plus simple et qui engendre le moins d'actions.

**Exemple de génération de buts**

**cas 1** : l'état antérieur est l'état 4 "marche normale" et l'état calculé est l'état 1 "tension_on". Le but généré est alors (?arrêt) d'après le tableau du II . 2 . 2 . 1 . 4.

**cas 2** : l'état antérieur est l'état 4 et l'état calculé est l'état 0 "tension-off" (en considérant qu'il s'agisse d'une machine virtuelle, il s'agit de la situation où une ses sous-machines a été accidentellement mise hors tension. La machine virtuelle est donc considérée comme hors tension). Le but généré est alors (?arrêt_tension) qui aura pour effet de positionner effectivement cette machine virtuelle hors tension.

Cette étape d'analyse est primordiale pour la bonne marche du système de production, en particulier, lors d'un fonctionnement autonome. Ce principe de comparaison entre l'état obtenu lors de la dernière inférence, état courant, et l'état obtenu lors de l'inférence en_cours, état calculé, permet de toujours focaliser l'attention du gestionnaire sur les éventuels problèmes de fonctionnement.

**Choix du chaînage avant**

Ce module de génération de buts à résoudre nécessite de prendre en compte le maximum d'informations. Les buts qui doivent être déduits sont au départ totalement inconnus et il n'y a pas de raison d'accentuer la recherche sur un fait ou un autre.

La première idée est de connaître l'état réel du système de production à partir d'informations assez précises par une phase d'évaluations et de calculs d'états. En fait, à partir d'informations de niveau inférieur, nous déduisons des informations de niveau supérieur, utilisées dans un raisonnement d'abstraction assez élevé. Ceci permet de plus, de compléter éventuellement la base de données. Aucune nouvelle information n'est oubliée lors de l'analyse.

L'avantage est que toute nouvelle inférence se fait sur des faits qui ont été mis à jour. Cela permet donc un rebouclage permanent sur le procédé.

Le chaînage avant a pour avantage de balayer toute la base de connaissance et donc de générer le maximum de buts.

Il est évident que nous déduisons des faits inutiles, défaut du principe du chaînage avant. Mais il est minimisé par le fait qu'il y a peu de règles étudiées : deux par machine (une destinée au calcul d'état, l'autre à la déduction des buts), présentées auparavant.

| | | | |
|---|---|---|---|
| R1 | (test_change machine) | -> | (calcul_état machine) |
| R2 | (nouvel_état_déduit machine) | -> | (calcul_but machine) |

### IV . 3 . 2 - Résolution en chaînage arrière

La résolution repose sur une exploration du domaine de recherche, modélisé par des arborescences, dont nous montrons tout d'abord la méthode de génération.

### IV . 3 . 2 . 1 - Génération des arbres de recherche

Il s'agit de créer un espace de travail pour le système d'inférence. Cet espace est modélisé par des arbres de recherche, arbres que nous construisons à partir de la représentation structurée proposée au chapitre II.

A partir de l'arborescence structurée, nous obtenons directement les arbres de recherche. Chaque noeud devient un but à résoudre. Il y a autant d'arbres qu'il y a de buts possibles. Cette décomposition correspond à celle retenue lors d'une inférence en chaînage arrière avec décomposition de buts en sous-buts. La traduction se fait en remplaçant systématiquement chaque noeud par un but à résoudre.

L'étude d'un noeud de niveau N sera effective lorsque la résolution des buts associés à ses fils sera réalisée. Un fils peut être un but ou une contrainte à vérifier.

Nous avons une représentation structurée et hiérarchisée complète avec intégration des contraintes. Nous transformons cette représentation en arbre de recherche.

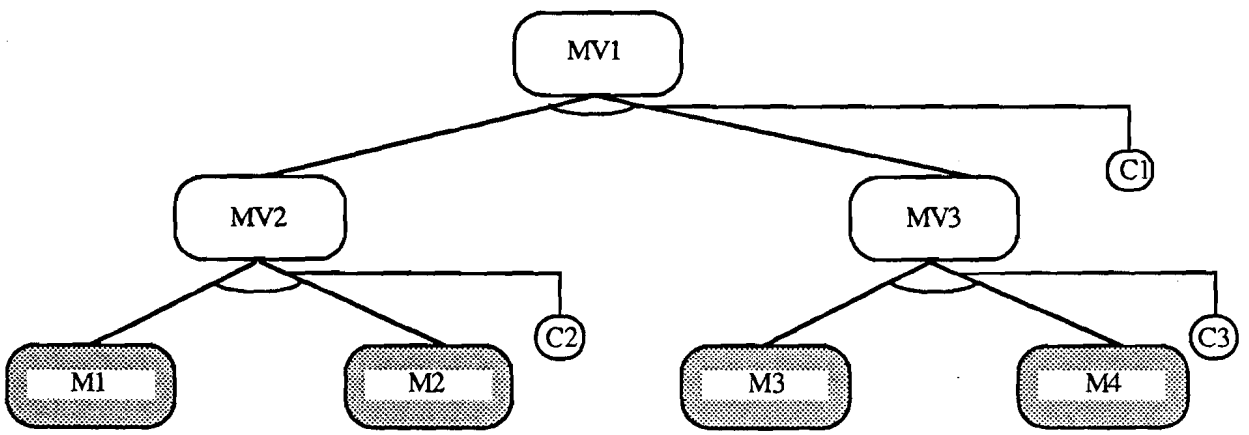



fig 25 : arbre initial

Afin de limiter le nombre de représentations, en illustrant la transcription sur un exemple générique. Le problème à résoudre n'est pas précisé et s'appelle (?Pb) : il s'applique à tous les buts possibles présentés au point II . 2 . 2 . 1 .4. Nous obtenons donc autant d'arbres qu'il y a de buts possibles. Cela dépend ainsi du nombre d'arcs des graphes d'état associés aux machines du système.

Nous obtenons la représentation suivante :

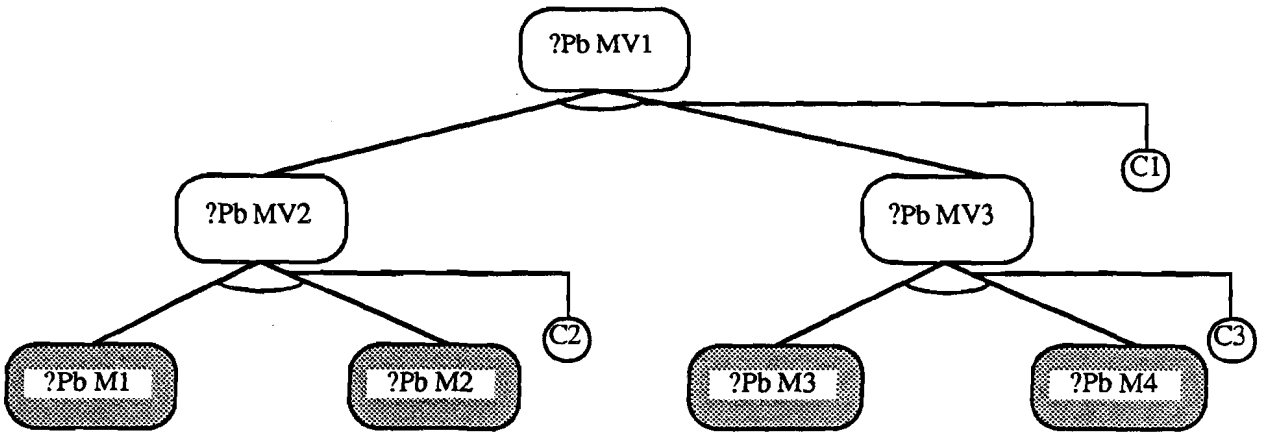



fig 26 : arbre de recherche

Nous retrouvons la structure classique d'un arbre de recherche de type ET/OU. Le problème concernant MV1 sera résolu lorsque ceux concernant M1, M2, M3 et M4, ainsi que les contraintes C1, C2 et C3 seront résolus. Cette structure sera exploitée lors de la phase de résolution en chaînage arrière.

**Exemple : cas de l'atelier de l'EC Lille**

Nous reprenons l'exemple de l'atelier de l'EC Lille. Nous traduisons l'arbre de la représentation structurée en arbre de recherche.

Plusieurs arbres de recherche sont générés, chacun étant dédié à un but particulier à résoudre. Il y a autant d'arbres qu'il y a de buts possibles.

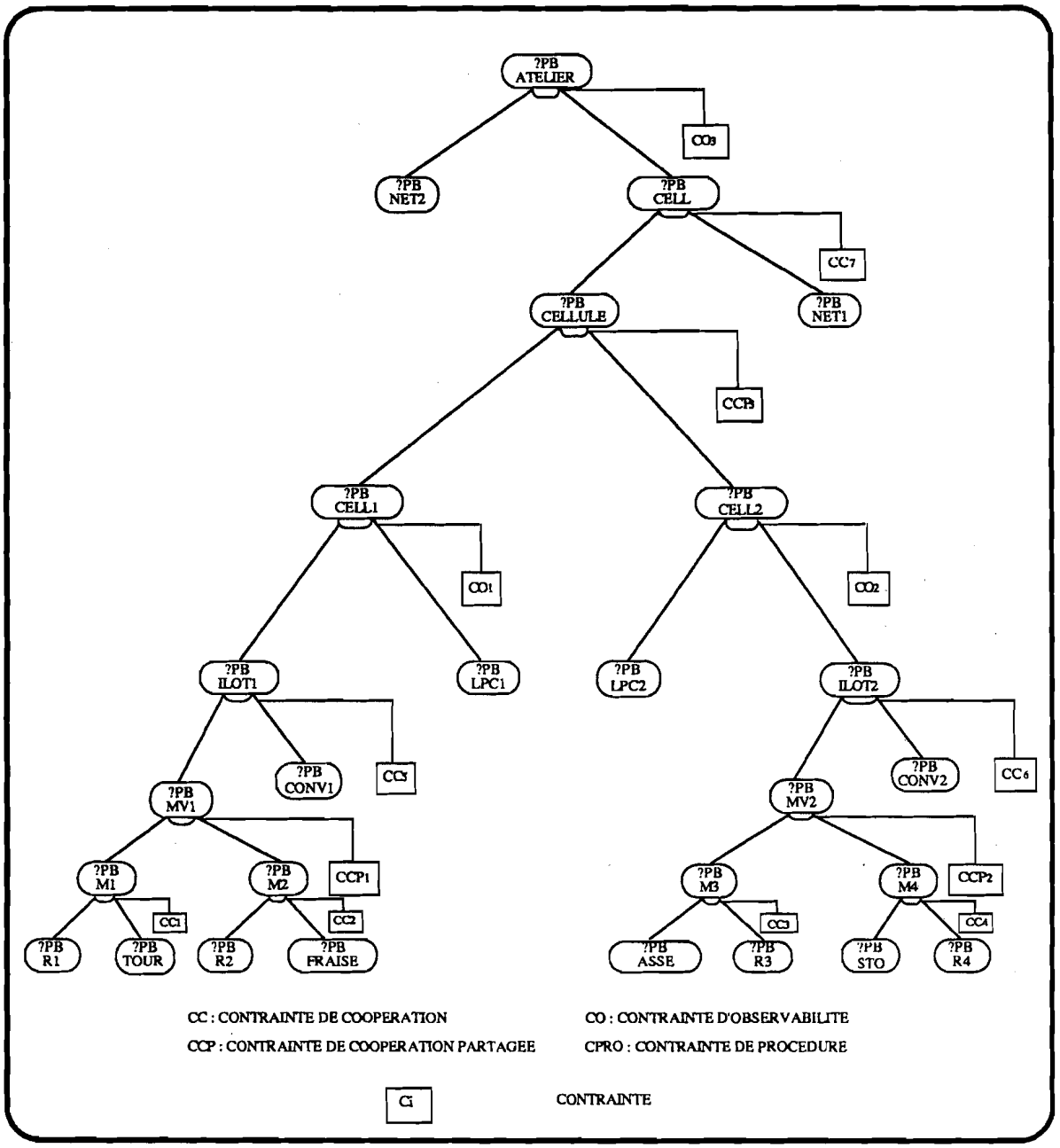



fig 27 : arbre de recherche, cas général

**Signification de l'arborescence** :

(?PB ATELIER), un but concernant la machine ATELIER, sera résolu lorsque (?PB NET2) et (?PB CELL), les sous-buts concernant les machines NET2 et CELL, seront résolus, avec la vérification de la contrainte CO3.

### IV . 3 . 2 . 2 - Choix du chaînage arrière

L'avantage du chaînage arrière est qu'il résoud plus rapidement le but en focalisant la recherche dans sa direction. La phase précédente de déduction de buts en chaînage avant, plus générale, nous autorise maintenant à particulariser une situation.

Nous avons de plus, privilégié le fonctionnement en profondeur d'abord. En effet, plusieurs solutions sont parfois possibles et notre intérêt est d'en obtenir une le plus rapidement possible. Nous considérons alors que la meilleure est celle détectée en premier, avec le respect des contraintes inhérentes au système de production. Nous orientons cependant la recherche par un système de priorité entre les règles afin de privilégier un certain type de solutions, lors de la sélection des règles. Par exemple, il faut chercher à privilégier le mode "marche normale".

### IV . 3 . 2 . 3 - Exploration dans l'arbre de recherche

Elle permet de résoudre les buts. Ceux-ci sont, soit déduits à partir des caractéristiques décrites précédemment, soit proposés par l'utilisateur. Les buts sont donc résolus un par un d'après leur importance. Une tentative d'effacement de buts en sous-buts s'amorce jusqu'à atteindre un but non effaçable. Les contraintes sont prises progressivement en compte, au fur et à mesure de l'évaluation des problèmes rencontrés.

Dans le cas où la recherche s'avère infructueuse, il y a retour en arrière jusqu'au dernier point non complètement exploré. Dans le cas où un but est résolu, il y a activation de certaines actions pour corriger le système.

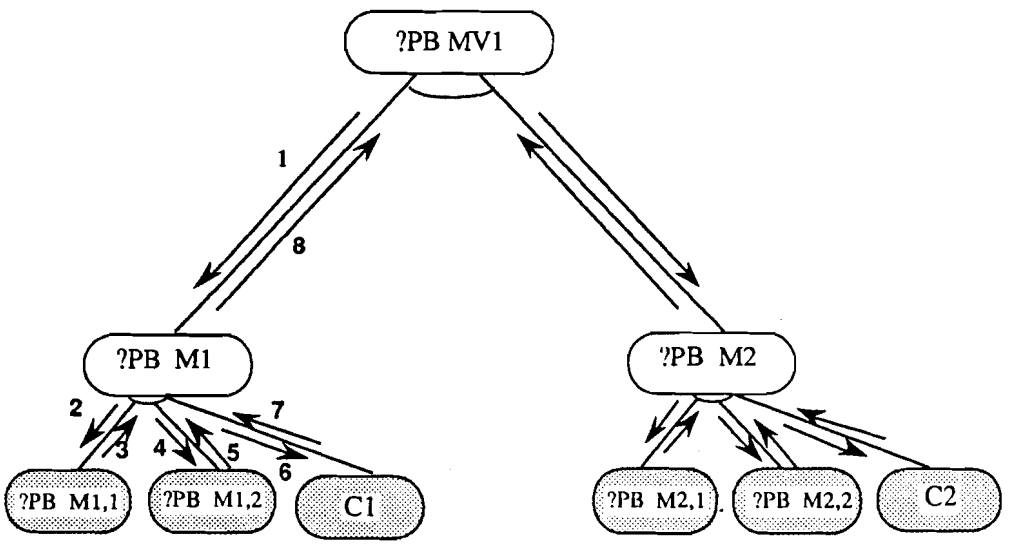



fig 28 : recherches dans l'arborescence

La résolution de (?PB MV1) sera effective lorsque toutes les feuilles de l'arborescence auront été explorées positivement (résolution dans l'ordre de (?PB M1,1) (?PB M1,2) (C1) (?PB M2,1) (?PB M2,2) et (C2).

Cela peut être, en reprenant l'exemple de l'EC Lille, le démarrage de la machine virtuelle MV1. Il correspond aux démarrages simultanés de R1, du TOUR, de R2 et de la FRAISEUSE, avec la prise en compte des contraintes de coopération CC1 et CC2 et de la contrainte de coopération partagée CCP1.

### IV . 3 . 2 . 4 - Structure du moteur d'inférence

Il est de nature classique, avec trois étapes :

- une phase de sélection par l'intermédiaire des méta-règles,
- une phase d'analyse avec un filtrage et une résolution de conflit,
- une phase d'exécution.

Nous décrivons ces différentes étapes.

#### IV . 3 . 2 . 4 . 1 - Etape de sélection

Cette étape consiste à déterminer un ensemble de règles qui sont susceptibles d'intervenir dans la phase de résolution, avec élimination de celles qui n'ont aucune chance d'être retenues lors de l'inférence.

Cette sélection se fait à l'aide de *méta-règles*, avec lesquelles nous pouvons restreindre le nombre initial de règles. Nous retirons ou ajoutons des règles pour optimiser le nombre de règles et surtout éviter une scrutation exhaustive de toutes les règles lors de la recherche en arrière.

Les stratégies retenues s'appliquent au moment de l'étape de restriction et du filtrage des règles. La manière d'écrire et de classer les règles influe sur leur ordre d'évaluation. Une stratégie possible est la minimisation du nombre des actions, en affectant des priorités en fonction du nombre d'actions qui sont associées à chaque règle.

Les méta-règles utilisent des informations déduites de l'état des graphes et de la décomposition structurelle de l'unité de production. Ces règles sont établies par une étude des graphes et des arborescences, basée sur une technique de réduction de l'espace de recherche, présentée plus en avant dans le rapport.

Le prototype développé a montré que 75% des règles sont retirées avant l'inférence, en appliquant ces techniques.

### IV . 3 . 2 . 4 . 2 - Phase d'analyse

Le moteur est caractérisé par un fonctionnement par tentatives avec des possibilités de retour arrière et un régime non-monotone.

A la suite de l'étape de détection, une liste de buts est proposée au début de l'inférence dans BASE_DES_BUTS.

- ◊ Si BASE_DES_BUTS est vide : le système commandé n'a pas évolué ou l'environnement (système de GPAO, utilisateur) n'a pas imposé de conditions susceptibles de modifier le procédé. Le système de pilotage n'a pas besoin de poursuivre l'analyse.

- ◊ Si BASE_DES_BUTS n'est pas vide : il s'agit alors de résoudre un but ou de vérifier la cohérence d'un but ou d'un ordre donné par l'utilisateur.

Etape d'ordonnancement de la liste des buts

Comme plusieurs buts sont générés, il s'agit de les classer afin d'en privilégier certains.
La liste de buts BASE_DES_BUTS est ordonnée en fonction de l'importance des machines impliquées dans la satisfaction du but. Il s'agit de privilégier le but qui met en cause la machine dont le niveau est le plus élevé dans la hiérarchie. Ce classement est réalisé pour plusieurs raisons :

- il faut avoir la vision la plus globale du système de production. Une modification de l'état d'une machine de haut niveau a des répercussions indéniables sur des machines de bas niveau,
- la résolution d'un but associé à une machine de haut niveau a de fortes chances de résoudre des buts déduits de l'état de machines de niveau inférieur.

**L'étape de résolution de conflits** est simple. Il s'agit d'un tri (tri Quicksort) de la base des règles obtenue après sélection, en tenant compte de leurs numéros d'identification dans un premier temps, de leurs priorités dans un second temps. Le tri n'est effectué qu'en début de chaque inférence, car les priorités ne sont pas dynamiques et l'ordre des règles ne sera donc pas modifié.

**L'étape de filtrage** consiste à comparer la partie déclencheur d'une règle par rapport à l'ensemble des faits et des buts à déduire, afin de vérifier sa compatibilité vis à vis de la situation en-cours.

Lors de l'analyse, le moteur déclenche la première règle dont le déclencheur-but et les déclencheurs-faits sont respectivement compatibles avec le but analysé et la base de faits. Dans ce cas, des conséquents (des sous-buts et éventuellement des actions) sont générés et sont rangés dans une pile de conséquents. Ces conséquents sont ensuite évalués en priorité, ce qui correspond à la notion de recherche en profondeur.

L'évaluation se fait dans l'ordre dans lequel étaient écrits les conséquents dans les règles. Cela signifie que les sous-buts sont tout d'abord vérifiés sans à priori, avec ensuite la vérification de contraintes.

- ◊ Si l'ensemble des conséquents est vérifié, le problème initial est alors résolu.
- ◊ Si l'évaluation de conséquents mène à une impasse (contraintes ou sous-buts non vérifiables), le moteur cherche une nouvelle règle susceptible d'être déclenchée en accord avec la situation.
- ◊ Sinon le moteur effectue un retour arrière sur la situation précédente et tente d'appliquer une nouvelle règle jusqu'à l'épuisement de la base de règles.

Si parmi les conséquents se trouvent des actions, elles sont alors empilées dans BASE_DES_ACTIONS jusqu'à la résolution finale du but. Dans le cas d'un retour arrière, elles sont alors dépilées.

Le moteur analyse but par but, en commençant par le plus important jusqu'à obtenir une satisfaction de chaque but. L'arrêt de la résolution se fait lorsque il n'y a plus de buts à résoudre (BASE_DES_BUTS vide) et/ou lorsque toutes les règles de la base de règles initiale ont été évaluées et retirées de cette base.

**La non-monotonie** se caractérise par le fait que le système évolue dynamiquement avec l'utilisation de règles dont les conséquents peuvent remettre en cause les faits, qui ont permis de les déclencher. Ceci se rapproche d'un système dynamique [VOY 87] qui permet d'intégrer le temps avec le séquencement des différents états que peuvent prendre les composants du système modélisé. La planification d'actions nécessite d'intégrer des changements d'état. A cet effet, nous gérons une liste de faits dans laquelle nous ajoutons et nous retirons des faits, caractérisant l'état des machines, au fur et à mesure de l'évolution des changements occasionnés par le déclenchement de règles. Cette liste est utilisée comme paramètre lors des appels de fonctions récursifs. Elle est passée par valeur ce qui permet de mémoriser les états intermédiaires et autorise la notion de retour arrière à une situation initiale dans la démarche de recherche. Nous avons préféré gérer une liste de faits par rapport à la base de faits globale, par souci de rapidité et de gain de place en mémoire centrale.

**Moyens de contrôle**

Ce sont les techniques utilisées pour garantir un temps optimum de recherche et qui permettent un filtrage des règles.

Le but du système est de générer la meilleure solution possible en un minimum de temps. Pour cela, nous considérons que la première solution détectée, répondant aux contraintes, est la bonne. Dans cette optique, il faut orienter la recherche vers la solution la plus profitable par différents moyens. Nous utilisons lors de la recherche en arrière, plusieurs techniques différentes.

La complexité de la résolution dépend du nombre de règles car une recherche dans un arbre ou dans un graphe équivaut au déclenchement d'une combinaison de règles, chaque combinaison étant un chemin possible. La complexité du domaine (nombre de règles) est liée au nombre de machines à contrôler.

Les moyens de contrôle sont :

- une affectation de niveaux de priorité aux règles : à l'aide de niveaux de priorité sur les règles, l'utilisateur peut favoriser des règles par rapport à d'autres. Ces règles sont évaluées en priorité. Ceci permet de favoriser un mode de marche par rapport à un autre et d'intégrer les différentes stratégies possibles,
- une configuration dynamique de la profondeur de recherche : en fonction de la machine impliquée dans la recherche, la profondeur maximale de recherche est calculée avant la phase d'analyse, ce qui permet de restreindre le temps de recherche. La profondeur est en fait la longueur maximale d'un chemin entre deux états,
- une restriction dynamique de la base de règles : de façon à éviter les problèmes de bouclage dans la recherche dans un graphe d'état, qui engendrerait des pertes de temps, nous choisissons de retirer une règle après son déclenchement pour qu'elle ne soit pas évaluée plusieurs fois,
- une restriction dynamique de la base de buts : dans le cas où un sous-but dégagé lors de la résolution fait partie de la base des buts initiaux non résolus, celui-ci est alors retiré de cette base. Ceci garantit le fait qu'un but ne sera pas rencontré plusieurs fois lors du même cycle d'inférence.

### IV . 3 . 2 . 4 . 3 - Phase d'exécution

Lorsque la résolution est terminée, les actions sont alors déclenchées dans l'ordre inverse dans lequel elles ont été empilées dans BASE_DES_ACTIONS. En effet, cela correspond à la philosophie du chaînage arrière : nous voulons prouver un but en le décomposant en sous-buts, pour arriver à une situation connue. Les actions qui sont associées à ce changement d'état sont forcément générées en ordre inverse dans lequel elles doivent être physiquement exécutées. La base de faits est alors effectivement modifiée.

**Application dans la gestion des modes de marche**

Le gestionnaire fonctionne en mode automatique : prenons comme exemple de résolution, le cas de la machine virtuelle M1 constituée des deux sous-machines R1 et TOUR, liées par une contrainte de coopération.

Au départ, toutes les machines sont dans l'état "marche normale".

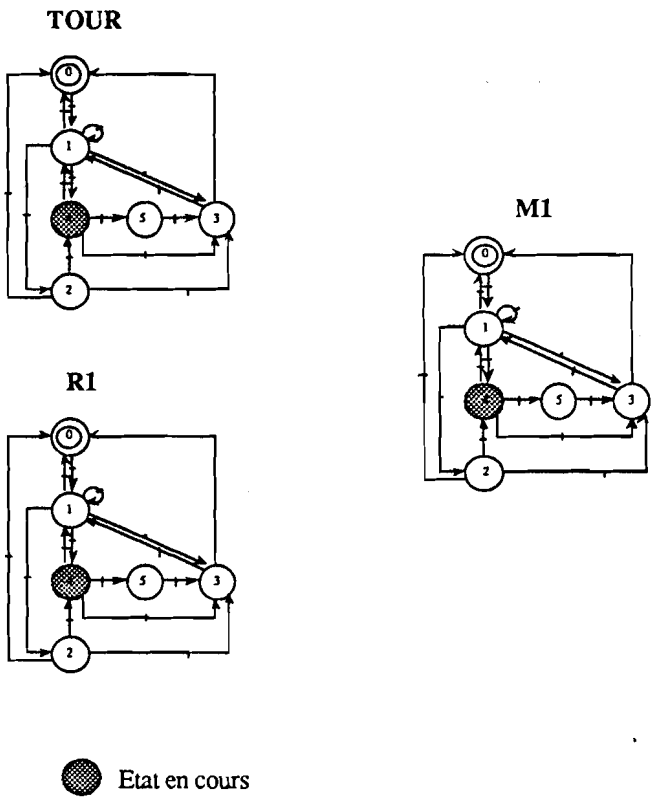



fig 29 : état initial

L'une d'elles (le TOUR) s'arrête (passage dans l'état "tension-on") : il s'agit alors d'intégrer cet arrêt en arrêtant les deux autres. Par convention, l'état "tension-on" correspond à la position de repli.

Par une résolution en chaînage avant qui consiste à calculer l'état de la machine virtuelle M1 avec application des règles de calculs, le gestionnaire détermine que la machine virtuelle se trouve dans l'état "tension_on". La règle R1 permet cette déduction.

R1 : (test_état M1) --> (calcul_état M1)

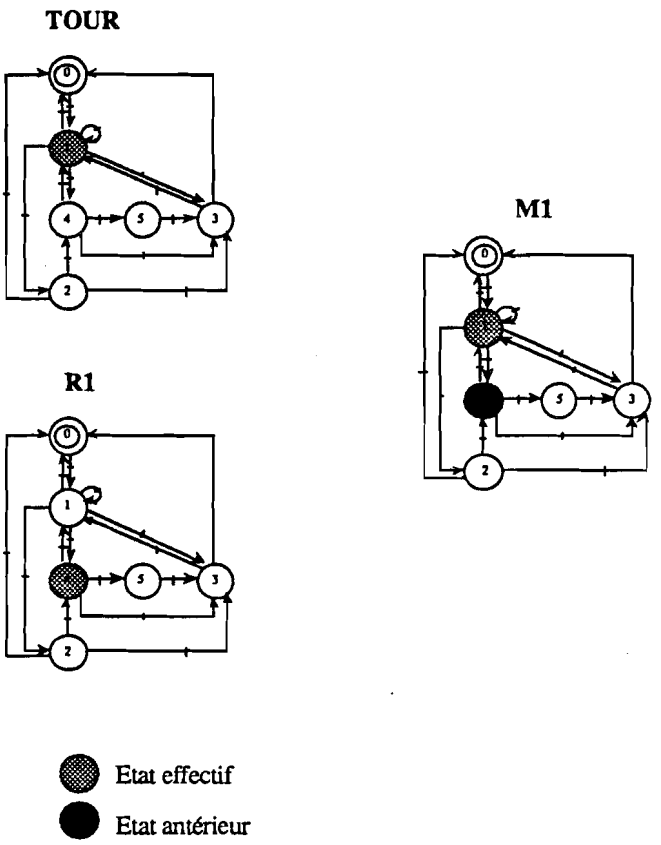



fig 30 : état transitoire

Toujours par chaînage avant, et par comparaison état antérieur/état effectif (déclenchement de la règle R2), le gestionnaire émet le but "?arrêt M1".

R2 : (éval_état M1) -> (calcul_but M1)

Le gestionnaire va alors résoudre le but par une inférence en chaînage arrière. Ceci aura pour effet d'arrêter effectivement les sous-machines R1 et TOUR. Ceci correspond à une démarche descendante pour la propagation des états. Chacune des machines se retrouve dans l'état "tension-on".

| | | | |
|---|---|---|---|
| R3 : | (?arrêt M1) | <- | (?arrêt TOUR)<br>(?arrêt R1)<br>(Contrainte CC1) |
| R4 : | (?arrêt TOUR)<br>(état TOUR marche_normale) | <- | (arrêter TOUR) |
| R5 : | (?arrêt R1)<br>(état R1 marche_normale) | <- | (arrêter R1) |

Le déclenchement de la règle R3 permet de dégager deux sous_buts (?arrêt TOUR) et (?arrêt R1) avec la contrainte de coopération CC1 à respecter. Ces sous_buts sont respectivement résolus par le déclenchement des règles R4 et R5 avec la proposition de deux actions à émettre (arrêter TOUR) et (arrêter R1). L'étude de la proposition permettra de vérifier la contrainte de coopération et autorise l'émission des actions : les deux machines passent dans l'état "tension-on", avec la mise à jour de la base de données.

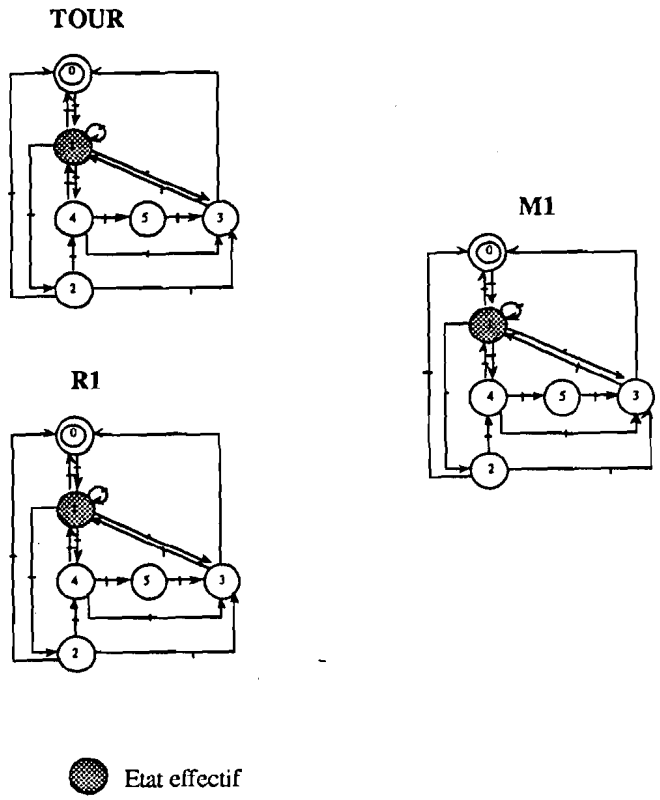



fig 31 : état final

Le principe de **la démarche descendante** permet la mise à jour des états des machines de niveau inférieur, à partir de l'état des machines de niveau supérieur. Cette démarche est

effectuée après correction de l'état du système. En général, les machines de niveau inférieur prendront le même état que celui de la machine virtuelle.

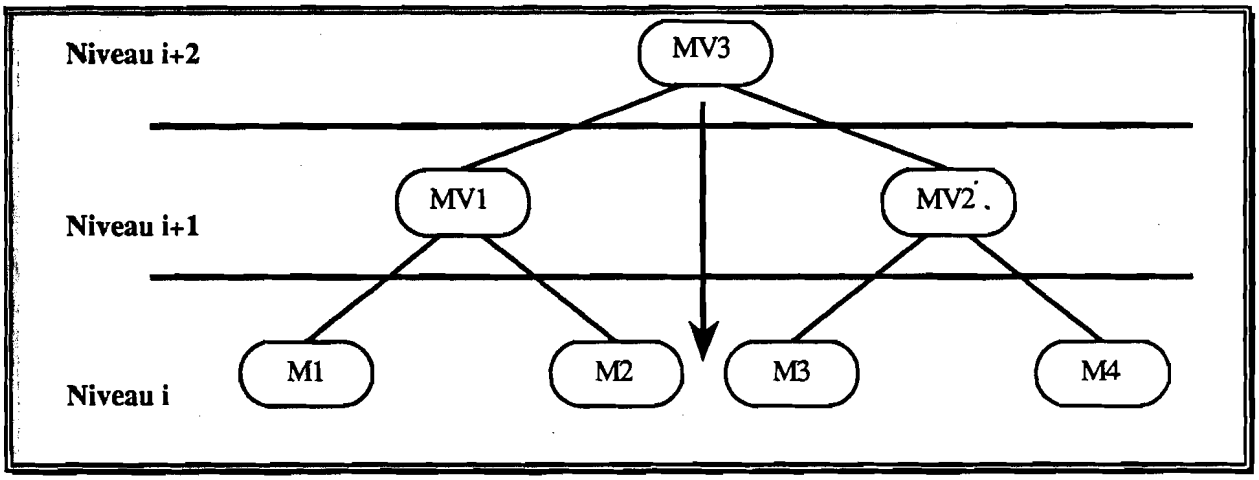



fig 32 : démarche descendante

Un cas particulier subsiste : celui des changements sur occurrence d'événements incontrôlables tels qu'une panne (exemple de la panne d'un composant ou d'un élément). Il n'est pas logique de considérer une machine en panne si elle ne l'est pas effectivement, sous prétexte que la machine avec laquelle elle est associée, le soit. Il s'agit alors de la mettre en position de repli. La solution retenue est de modifier la structure des actions émises dans cette situation. En effet, à tout but correspond une macro-action qui permet de le résoudre. Les actions "arrêter_direct_sous_déf", "passer_en_marche_sous_déf", et "arrêter_machine_sous_déf" qui correspondent respectivement aux buts "?arrêt_direct_sous_déf", "?passage_marche_sous_déf" et "?arrêt_marche_sous_déf", ne sont valides que si l'état de la machine est effectivement en panne. Dans ce cas, les actions associées à cette panne, telles que la reconfiguration de graphe, sont déclenchées. Sinon, la solution retenue est de faire passer systématiquement les machines associées en état "tension-on", considéré comme position de repli.

**L'exemple classique** est l'avénement d'une panne dans une architecture de production en série. Nous avons un îlot de fabrication constitué de machines groupées en série. Si l'une d'elles tombe en panne, l'information est alors reportée au niveau de la base de données avec l'évolution du marquage de son graphe d'état. Le gestionnaire calcule par chaînage avant le nouvel état de la machine virtuelle, considérée comme en panne (l'ensemble des machines ne peut plus globalement fonctionner) et déduit le but "?arrêt_direct_sous_déf", par comparaison des états antérieur et effectif. Par résolution arrière, ce but est décomposé en sous-buts identiques, chacun d'eux concernant une des machines de l'îlot. Les machines en état correct seront mises alors en position de repli et des actions seront déclenchées pour considérer la machine défaillante comme en panne.

### IV . 3 . 3 - Prise en compte du temps

L'acquisition des données se fait une fois pour toutes en début de cycle d'inférence. En effet, le support de communication que nous utilisons actuellement nous oblige à transférer le maximum de données ce qui évite d'aller chercher des données en cours d'inférence. Tout se passe comme si le process était gelé pendant le temps du cycle d'inférence.

Nous pouvons estimer le temps maximum de calculs, l'espace de recherche étant connu au départ. Ce temps est donc majoré et il devra être inférieur aux constantes de temps du process.

### IV . 3 . 4 - Evolution et maintenance de la base de connaissances

Si la structure physique du système évolue, alors une redéfinition de la base de connaissances s'avère nécessaire car la structuration des règles est fondamentalement changée.

Cependant, si la structure globale demeure avec uniquement modification de paramètres, il suffit d'aller modifier et configurer la base de connaissances avec un éditeur intégré. Par exemple, si l'utilisateur veut privilégier un mode de fonctionnement, il suffit d'aller modifier la priorité des règles destinées à la gestion de ce mode de marche.

Le monde créé est un monde ouvert : toute connaissance se doit d'être explicitement représentée et instanciée dans la base de connaissance. Le monde ouvert s'oppose au monde clos dans lequel il est possible de formuler des hypothèses en cas d'absence de données.

## V - GENERATION DES REGLES

Les règles sont formalisées de telle façon qu'il est possible de les générer automatiquement à partir de la modélisation présentée précédemment.

Pourquoi une génération automatique? Le nombre de règles nécessaire pour modéliser complètement le système est important et nécessite l'analyse de la cohérence et de la complétude de la base de connaissances. Générer automatiquement les règles permet de règler partiellement cette analyse.

De plus une génération permet une implantation beaucoup plus rapide du système informatique.

A partir de la modélisation retenue lors de la phase de spécifications, l'usager définit ses priorités et les contraintes, qui seront prise en compte lors de l'écriture des règles. A partir de

là, les règles peuvent être générées automatiquement. Cependant la phase de spécifications reste importante.

Deux classes de règles sont possibles :

- la première contient celles concernant le fonctionnement d'une seule machine, et sont construites à partir des graphes d'état,
- la seconde contient celles concernant le fonctionnement de plusieurs machines et sont construites à partir des arborescences structurées.

Ces règles ont la même syntaxe mais ne répondent pas à la même idée de résolution. Elles privilégient deux modes de recherche bien distincts. Elles correspondent, en fait, soit à l'analyse du comportement individuel d'une machine (modélisé par un graphe d'où un parcours dans un graphe), soit à l'analyse de comportement d'une machine virtuelle, (modélisé par un arbre d'où un parcours dans un arbre). La différence réside dans le fait que leurs heuristiques de recherches sont différentes, bien qu'une structure d'arbre soit un graphe particulier.

### V . 1 - Cas des machines effectives

Nous montrons la correspondance des règles avec les graphes.
Deux types de règles sont distingués :

Les règles, "**simples**", sont les règles de base qui permettent de passer directement d'un état initial à un état final, sachant que nous nous trouvons dans l'état initial. A chaque arc correspond une série d'actions.

Ce sont en fait les conditions de passages d'un état à l'autre sur le graphe d'état.
Ces règles introduisent un seul pas de réflexion.

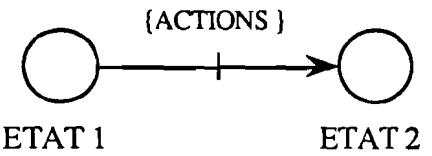



fig 33 : "règle simple"

La règle équivalente associée aura pour syntaxe :

(? Passage état 1 vers état 2) <- (Emettre les actions)
(Présence état 1)

Ceci correspond à la situation suivante : sachant que nous nous trouvons dans un état de départ, l'état 1, le but est de passer de l'état 1 à l'état 2. Une ou plusieurs actions sont donc émises afin d'effectuer ce changement d'état. Ces règles sont terminales dans la démarche d'inférence, par le fait qu'elles n'apportent pas d'autres sous-buts à résoudre.

Le deuxième type de règles correspond à une situation plus **complexe**.

Le but est de passer d'un état à un état final, en sachant que nous ne nous trouvons pas dans cet état, ce qui oblige à tenir compte des états antérieurs qui mènent à cet état intermédiaire.

Cela correspond par exemple, à la situation où le but est de passer de l'état 4 à l'état 5 en émettant l'hypothèse, par exemple, que nous nous trouvons dans l'état 2.

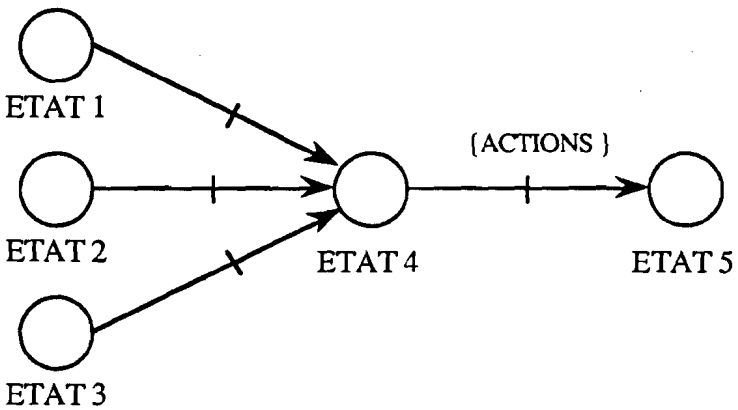



fig 34 : "règle complexe"

Trois règles traduisent cette situation :

(? Passage état 4 vers état 5) <- (Emettre les actions )
(? Passage état 2 vers état 4)

Cette règle signifie que pour résoudre le but "passer de l'état 4 à l'état 5", il faut émettre les actions correspondantes ET résoudre le but "passer de l'état 2 à l'état 4".

De même, nous obtenons :

(? Passage état 4 vers état 5) <- (Emettre les actions )
(? Passage état 1 vers état 4)

(? Passage état 4 vers état 5) <- (Emettre les actions )
(? Passage état 3 vers état 4)

Ces règles permettent en fait le cheminement dans le domaine de recherche, il s'agit alors d'écrire autant de règles que d'états antérieurs existants. Ces règles sont essentielles par le fait qu'elles permettent d'émettre des hypothèses.

### V . 2 - Cas des machines virtuelles

Les règles sont différentes du point de vue de leurs interprétations. Il s'agit, en l'occurrence, de la traduction des arborescences de recherche sous forme de règles. Cela signifie qu'elles serviront à la décomposition de problèmes en sous-problèmes lors des inférences de résolution. Ce sont des règles qui serviront d'intermédiaires dans la démarche d'inférence.

Nous allons tout d'abord montrer la philosophie du passage aux règles puis nous détaillerons les différents cas possibles.

A partir d'un arbre de recherche, il s'agit de considérer chaque combinaison de branches comme une règle qui exprime la décomposition d'un problème en sous-problèmes.

Considérons l'arbre de recherche présenté ci-dessous

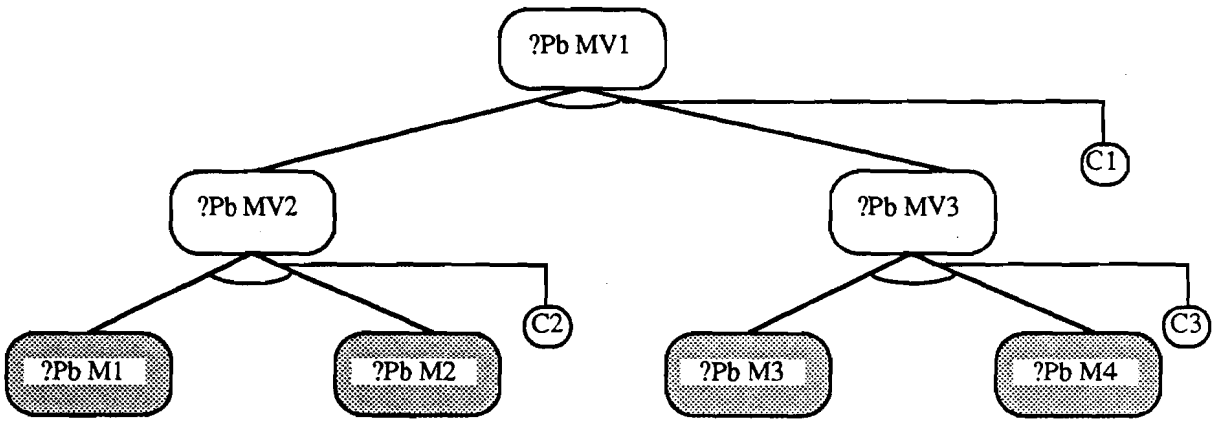



fig 35 : élaboration des règles

Nous n'avons que des contraintes ET avec variantes. Ce schéma signifie que le problème quelconque concernant MV1, sera résolu en résolvant le même problème sur MV2 et MV3 en tenant compte de la contrainte C1.

La syntaxe de la règle équivalente respecte la structure suivante :

(? Pb MV1) <- (? Pb MV2)
(? Pb MV3)
(C1)

De la même façon, nous construisons le même type de règles pour chaque machine virtuelle MV2 et MV3.

Les feuilles de l'arbre correspondent aux problèmes terminaux non décomposables.

### V . 3 - Génération à partir des arborescences

Nous allons montrer comment il est possible d'utiliser la représentation arborescente afin de générer directement les règles de façon systématique. Nous allons différencier le cas d'un arbre de type ET d'un arbre de type OU.

#### V . 3 . 1 - L'arbre ET

**Représentation arborescente**

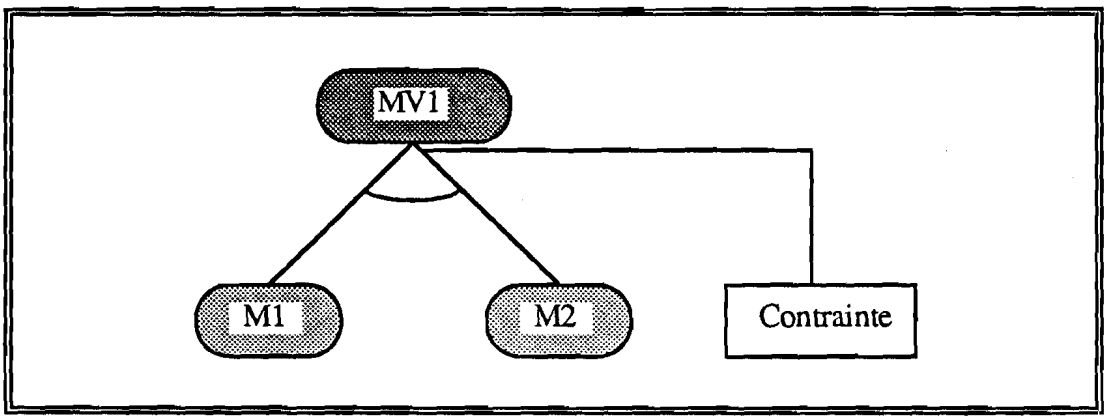



fig 36 : arbre ET

**Traduction en règle**

**R1 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M1) |
| | | (?PB M2) |
| | | (contrainte) |

Cette règle signifie que pour résoudre un problème concernant MV1, il suffit de résoudre ce problème pour M1 et M2, tout en respectant la contrainte indiquée.

### V . 3 . 2 - L'arbre OU

**Représentation arborescente**

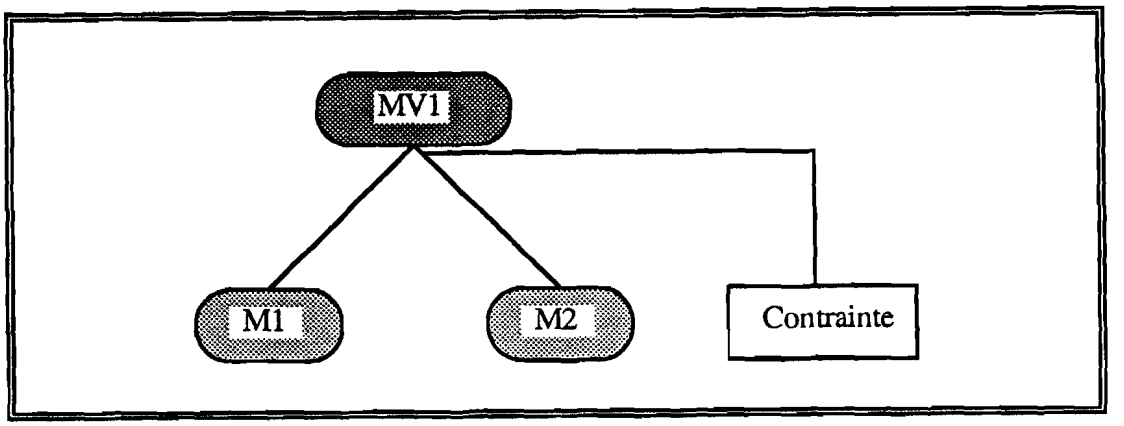



fig 37 : arbre OU

**Traduction en règle**

**R1 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M1) |
| | | (contrainte) |

**R2 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M2) |
| | | (contrainte) |

Deux règles sont nécessaires pour montrer le caractère OU de l'arborescence. Ces règles signifient que pour résoudre un problème concernant MV1, il suffit de résoudre soit ce problème pour M1, soit pour M2, tout en respectant la contrainte indiquée.

L'écriture permet de favoriser une machine par rapport à une autre :

- soit en mettant un niveau de priorité supérieur,
- soit, à niveau égal de priorité, un numéro de règle supérieur.

Nous détaillons la démarche pour chaque type de contrainte.

### V . 3 . 3 - Contrainte de coopération CC

Deux ou plusieurs machines coopèrent de manière étroite.

**Représentation arborescente**

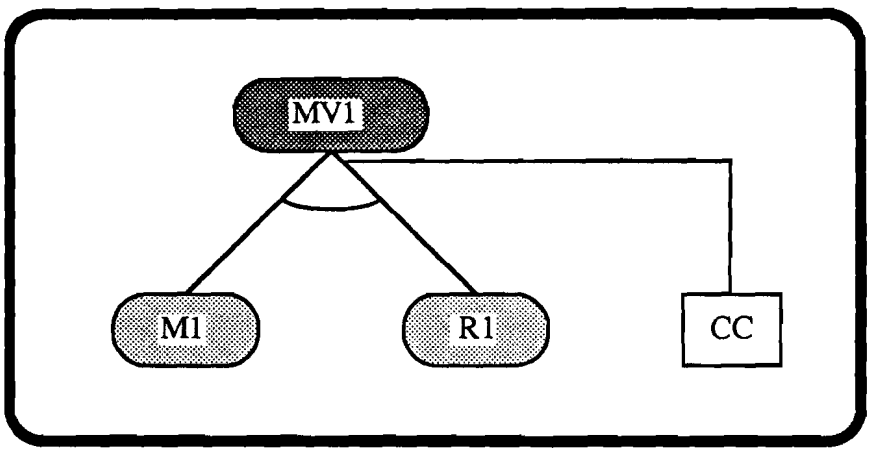



fig 38 : coopération totale

**Traduction en règle**

(?PB MV1) <-- (?PB M1)
(?PB R1)
(contrainte CC)

### V . 3 . 4 - Contrainte de coopération partagée CCP

Deux ou plusieurs machines partagent une ressource.

**Représentation arborescente**

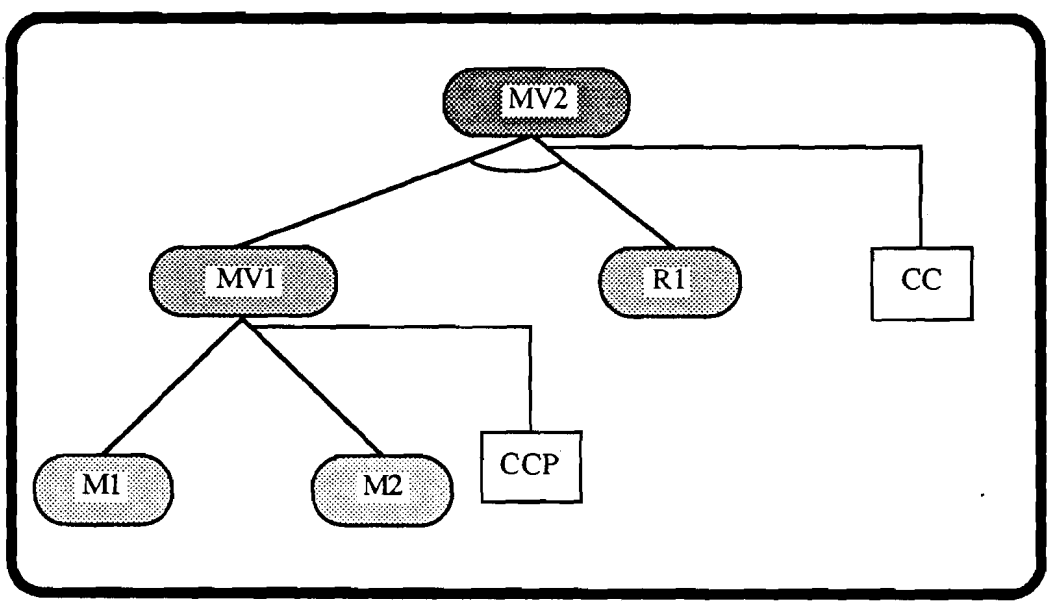



fig 39 : coopération partagée

**Traduction en règle**

Une première règle est nécessaire pour gérer le comportement de la machine virtuelle MV2.

**R1 :**

| | | |
|---|---|---|
| (?PB MV2) | <-- | (?PB MV1) |
| | | (?PB R1) |
| | | (contrainte CC) |

Deux règles sont nécessaires pour gérer le comportement de la machine virtuelle MV1 :

**R2 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M1) |
| | | (contrainte CCP) |

**R3 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M2) |
| | | (contrainte CCP) |

Le problème concernant MV1 sera résolu, soit en résolvant le même problème concernant M1 ou M2, tout en tenant compte de la contrainte CCP

**V . 3 . 5 -** Contrainte d'exclusion CE

Deux ou plusieurs machines ne doivent pas être simultanément dans des états déterminés.

**Représentation arborescente**

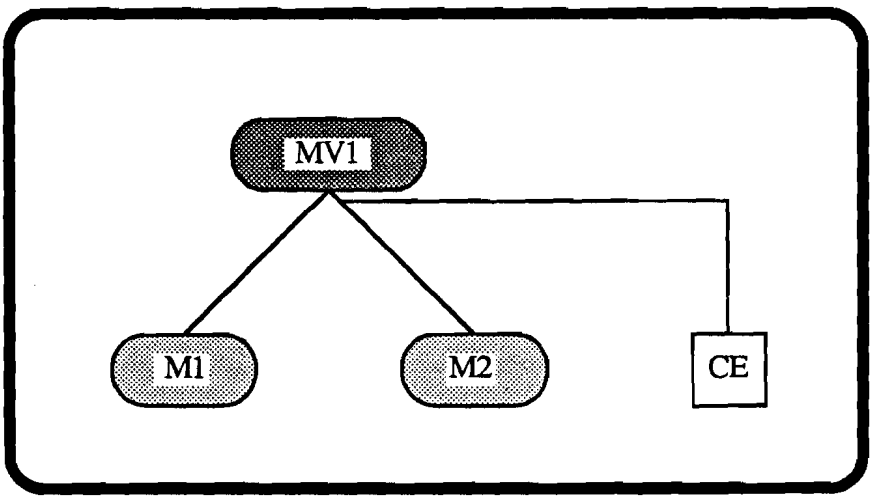



fig 40 : contrainte d'exclusion

**Traduction en règle**

**R1 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M1) |
| | | (contrainte CE) |

**R2 :**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M2) |
| | | (contrainte CE) |

Deux règles sont nécessaires pour traduire le graphe. La contrainte d'exclusion est une contrainte sur état et permet de gérer le comportement de chaque machine de manière individuelle, tout en tenant compte de la contrainte.

### V . 3 . 6 - Contrainte procédurale CPRO

Deux ou plusieurs machines sont liées par des procédures de fonctionnement.

**Représentation arborescente**

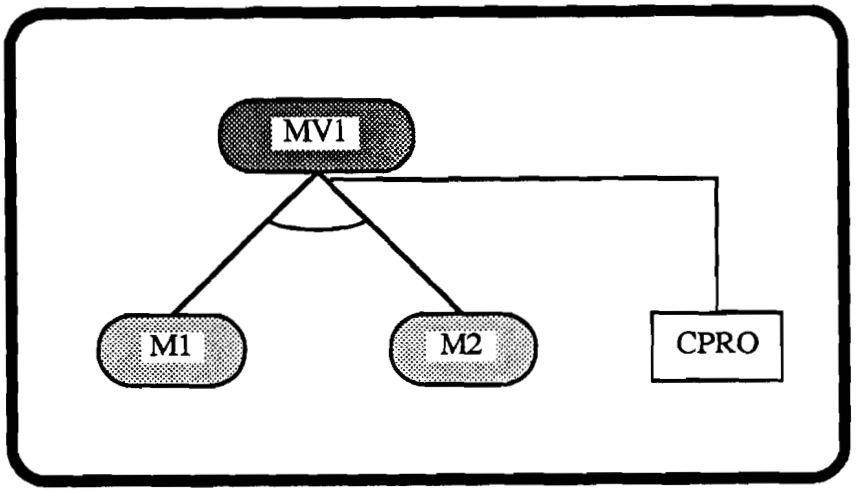



fig 41 : contrainte procédurale

**Traduction en règle**

| | | |
|---|---|---|
| (?PB MV1) | <-- | (?PB M1) |
| | | (?PB M2) |
| | | (contrainte CPRO) |

### V . 3 . 7 - Contrainte d'observabilité CO

C'est la possibilité ou non d'accéder aux informations nécessaires à la commande d'une ou plusieurs machines.

**Représentation arborescente**

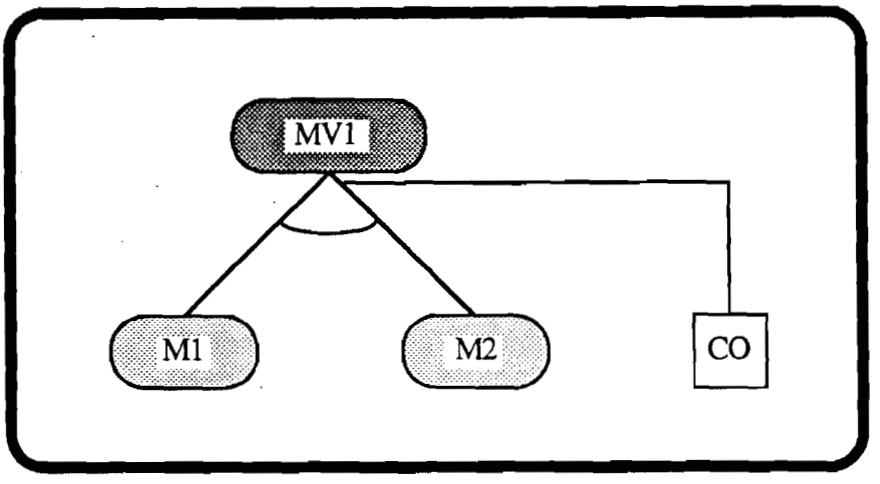



fig 42 : contrainte d'observabilité

**Traduction en règle**

(?PB MV1) <-- (?PB M1)
(?PB M2)
(contrainte CO)

**V . 3 . 8 -** Exemple de l'atelier

Nous détaillons plus précisément le cas du but "arrêt".

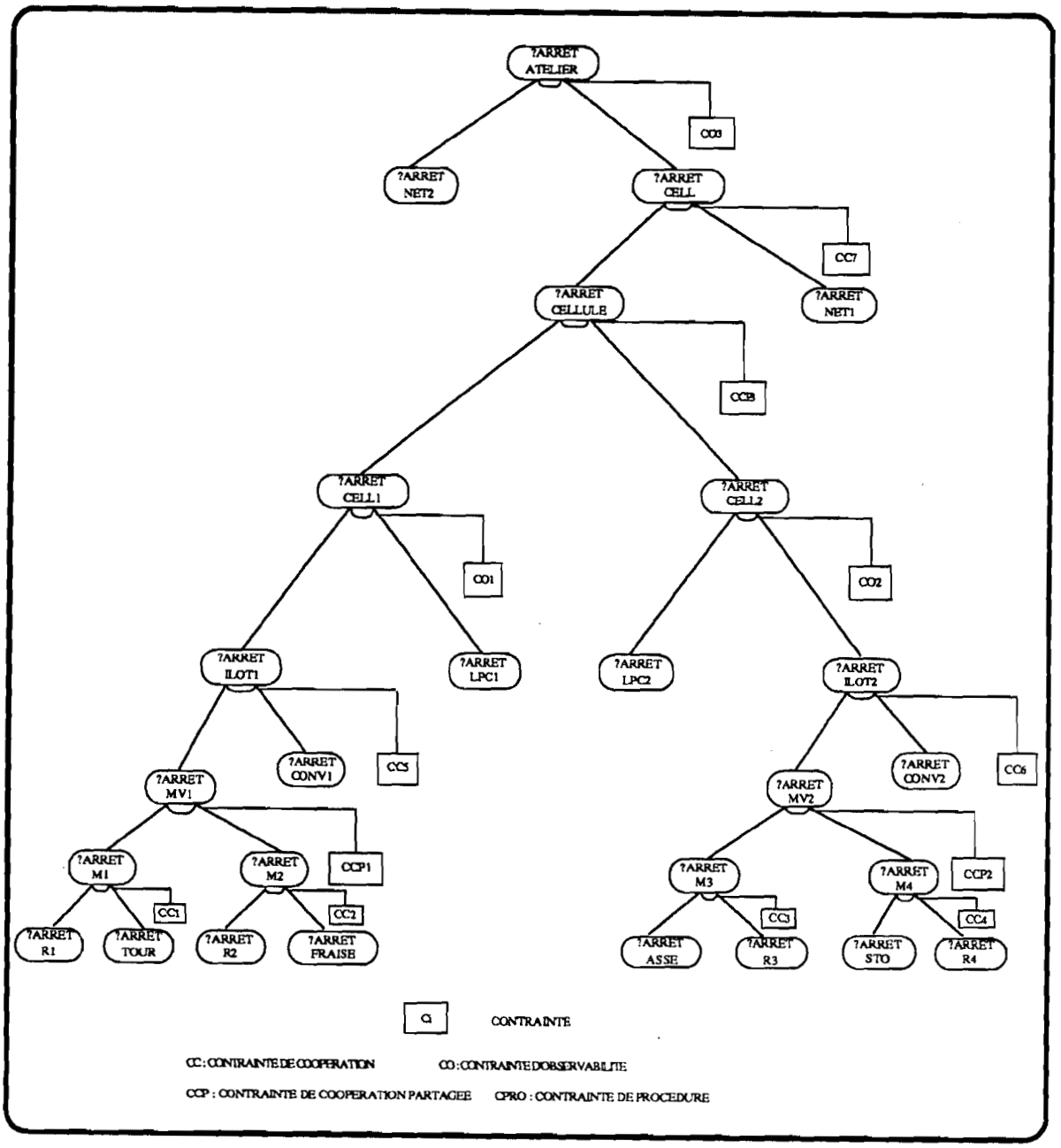



fig 43 : arbre de recherche pour la commande d'arrêt

(?ARRET ATELIER) sera résolu lorsque (?ARRET CELL) et (?ARRET NET2) seront résolus : l'arrêt de l'atelier sera effectif lorsque l'arrêt du réseau et l'arrêt de la machine Cell seront réalisés.

Nous allons traduire cet arbre de recherche en règles, qui concerne donc la résolution du problème ?arrêt de l'atelier.

Nous citons quelques règles significatives :

**Règles concernant les machines virtuelles :**

R1 : (?arrêt ATELIER) <- (?arrêt NET2)
(?arrêt CELL)
(Contrainte CO3)

R2 : (?arrêt CELL) <- (?arrêt CELLULE)
(?arrêt NET1)
(Contrainte CC7)

R3 : (?arrêt CELLULE) <- (?arrêt CELL1)
(?arrêt CELL2)
(Contrainte CCP3)

R4 : (?arrêt CELL1) <- (?arrêt ILOT1)
(?arrêt LPC1)
(Contrainte CO1)

R5 : (?arrêt MV1) <- (?arrêt M1)
(?arrêt M2)
(Contrainte CCP1)

R6 : (?arrêt M1) <- (?arrêt R1)
(?arrêt TOUR)
(Contrainte CC1)

R7 : (?arrêt M2) <- (?arrêt R1)
(?arrêt FRAISEUSE)
(Contrainte CC2)

**Règles concernant les machines effectives :**

R8 : (?arrêt R1) <- (arrêter R1)
(état R1 marche_normale)

R9 : (?arrêt R2) <- (arrêter R2)
(état R2 marche_normale)

R10 : (?arrêt LPC1) <- (arrêter LPC1)
(état LPC1 marche_normale)

R11 : (?arrêt CONV1) <- (arrêter CONV1)
(état R1 marche_normale)

R12 : (?arrêt TOUR) <- (arrêter TOUR)
(état TOUR marche_normale)

R13 : (?arrêt FRAISEUSE) <- (arrêter FRAISEUSE)
(état FRAISEUSE marche_normale)

## V . 4 - Génération des méta-règles

En utilisant les modèles de représentation, nous proposons de générer automatiquement la base de méta-règles qui permettra, par la suite, d'optimiser la recherche de solutions.

Ces règles traduisent en fait les techniques de réduction de graphes d'état et de réduction d'arborescence qui existent pour limiter le domaine de recherche.

Deux méta-règles par machine effective ou virtuelle sont ainsi générées :

- une pour la réduction du graphe d'état,
- une pour la réduction de l'arborescence.

### V . 4 . 1 - Réduction des graphes d'état

Le but de cette démarche est l'optimisation du parcours de recherche dans un graphe. En effet, dans le cas où l'on désire passer d'un état i à un état j, plusieurs chemins sont possibles. L'idée est de retenir le chemin le plus court. Le système d'inférence ne peut reconnaître les chemins inutiles ou possibles logiquement, mais impossibles techniquement. Par exemple, pour

passer de l'état 0 à l'état 2, c'est-à-dire la mise en marche d'une machine, plusieurs solutions sont possibles :

- mise sous tension (passage 0->1), vérification de l'état (passage 1->1), mise en marche (passage 1->4), arrêt de la marche (passage 4->1), et initialisation (passage 1->2),
- mise sous tension (passage 0->1), mise en marche (passage 1->4), arrêt de la marche (passage 4->1), et initialisation (passage 1->2),
- mise sous tension (passage 0->1), vérification de l'état (passage 1->1), et initialisation (passage 1->2),
- mise sous tension (passage 0->1), et initialisation (passage 1->2).

Certaines solutions sont complètement impensables techniquement. Il faut donc les éliminer. De plus, d'après le marquage des graphes des machines virtuelles ou effectives, nous étudions le passage d'un état initial, état en cours de la machine, à l'état final recherché. Par une étude du graphe, il est possible de savoir quels sont les chemins ou arcs qui ne seront pas utilisés pour passer de l'état initial à l'état final.

Nous associons une fonction de coût à chaque trajet. La fonction de coût est simple. A chaque pas ou franchissement d'une transition est associé un coût de 1. Un état de départ est défini et le total des coûts est réalisé pour chaque état atteint.

Le principe de réduction est de fixer à priori, un coût maximum à ne pas dépasser. Nous éliminons tout arc dont la fonction de coût est supérieure au maximum.

Deux critères sont utilisés :

- la restriction de profondeur de façon à limiter la longueur de parcours ce qui est réalisée par un calcul de coûts et l'évaluation des différents chemins possibles.

Critère d'utilisation

- si coût > coût_max
  alors abandon de la solution,
- si coût <= coût_max
  alors solution possible.

Le coût_max correspond en fait au nombre de transitions qu'il est possible de franchir et donc au nombre d'actions (macro_actions) qu'il est possible d'émettre

vers le procédé lors de la même phase de planification. Le coût maximum dépend du nombre de machines mises en cause.

- l'optimisation des états intermédiaires pour :

  * éviter les bouclages (émissions de deux actions antagonistes),
  * optimiser le passage d'un état à un autre (minimisation des coûts).

Le rejet des bouclages, associé à la limitation de profondeur oriente la recherche vers des solutions fiables techniquement.

Un exemple de calculs est proposé : il concerne un trajet avec pour état initial, l'état 0.

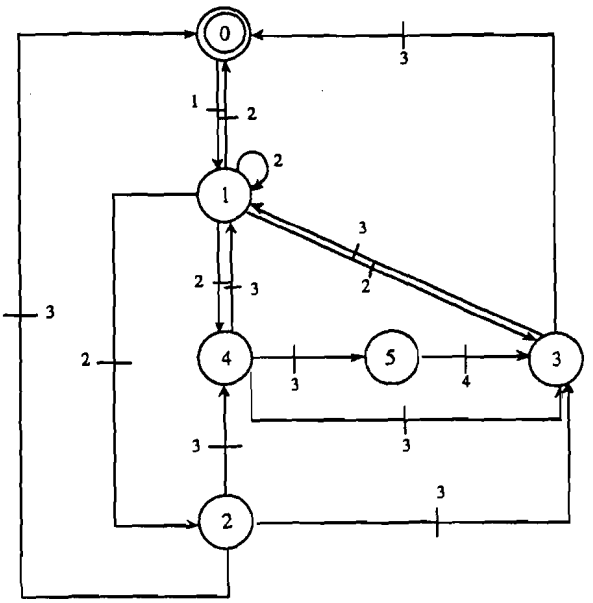



fig 44 : calcul des états accessibles avec l'état initial 0

Une autre technique est de supprimer les "allers-retours" entre deux états. Nous fixons un seul arc entre deux états. De même, nous supprimons tous les arcs qui convergent vers l'état de départ. En effet, il est logique de ne pas repasser par l'état initial.

### V . 4 . 2 - Application

Nous allons montrer comment nous utilisons ce calcul. Le graphe de base est celui présenté auparavant dans le chapitre II.

Nous calculons les coûts des trajets à partir de l'état 0. Nous ne retenons que ceux dont le coût est inférieur ou égal à 3. En effet, il s'agit du coût minimal pour passer normalement d'un

état à un autre. Tout coût supérieur à 3 signifie qu'il y a eu des bouclages dans le cheminement dans le graphe.

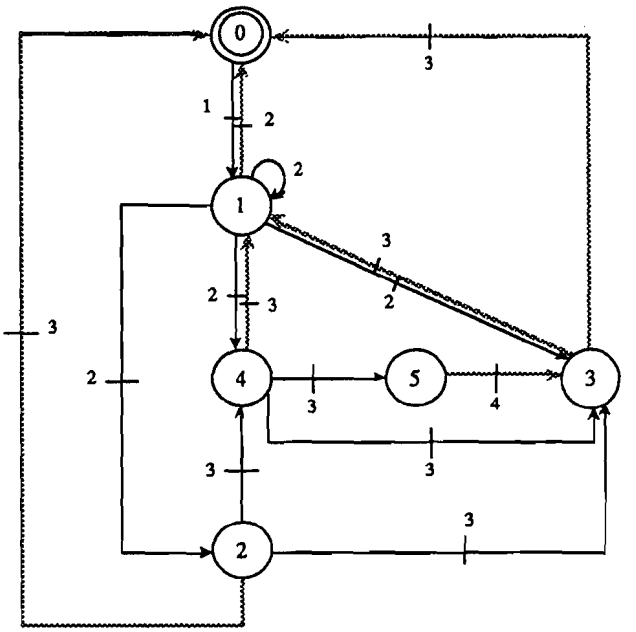



fig 45 : mise en évidence des arcs inutiles

Les arcs détectés par l'application des techniques précédentes sont mises en évidence. Les arcs possibles sont marqués. Nous allons retenir uniquement les chemins qui permettent d'arriver à l'état 4 en un coût acceptable vis à vis du critère initial.

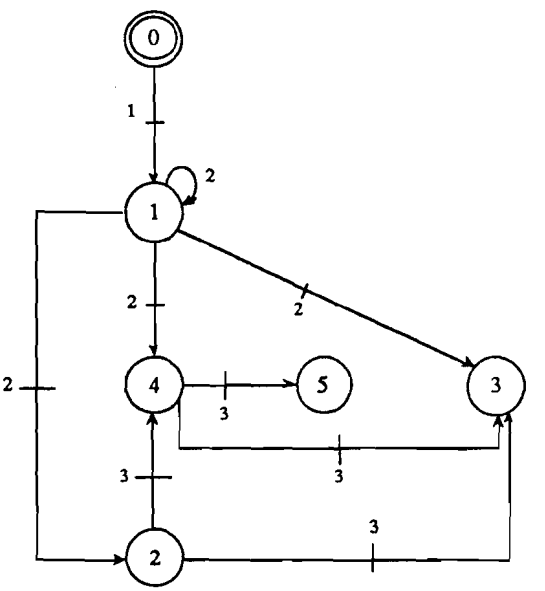



fig 46 : transitions possibles

Le graphe de référence est retenu en tenant compte de ces coûts. Ce graphe montre les chemins qui sont conservés.

Nous pouvons remarquer qu'il subsiste toujours plusieurs trajets possibles entre l'état initial et un état d'arrivée, en particulier pour l'état 3 et qu'il demeure un indéterminisme de fonctionnement.

Pour remédier à cela, nous posons que le chemin retenu sera le trajet le moins coûteux, à état initial et état final équivalent. Ceci permettra de privilégier les règles intéressantes, c'est à dire celles dont la fonction de coût associée est la moins élevée. A coût égal, l'utilisateur choisit sa solution, c'est à dire celle qui lui paraît la plus judicieuse.

Nous pouvons donc alléger le graphe en supprimant les arcs redondants.

Une autre méthode serait de classer les règles suivant leurs importances, en jouant sur leur numérotation et leur niveau de priorité. Le choix de la règle se fera lors de l'étape de sélection. Cette méthode a le mérite d'offrir plusieurs solutions, mais de surcharger la base de règles.

### V . 4 . 3 - Réduction des arborescences

Le but de cette étude est de réduire les arbres de recherche.

La recherche en arrière consiste en un parcours d'arbre. L'idée est de restreindre la recherche à la partie d'arborescence effectivement parcourue. Pour cela, nous analysons le but à résoudre qui indique le niveau et le sommet à partir desquels il faut commencer la recherche. Toutes les règles correspondant aux comportements de machines n'appartenant pas au "sous-arbre" sont retirées. Cela focalise la commande sur les machines effectivement impliquées dans la conduite.

Méthode utilisée

départ de l'analyse : **l'ensemble des buts à résoudre**

* sélection de la machine de plus haut niveau,
* réduction de l'arborescence par élagage de la partie non concernée.

**Exemple de réduction**

Nous allons montrer comment, à partir de l'analyse d'un but, nous réduisons l'arborescence de recherche par l'intermédiaire de méta-règles.

Par exemple, le but à résoudre est l'arrêt de MV2. La racine de l'arborescence de recherche devient le noeud correspondant à ce but. Pratiquement, ne sont conservées que les règles correspondant à cette arborescence. Ceci permet de réduire notablement la base de règles.

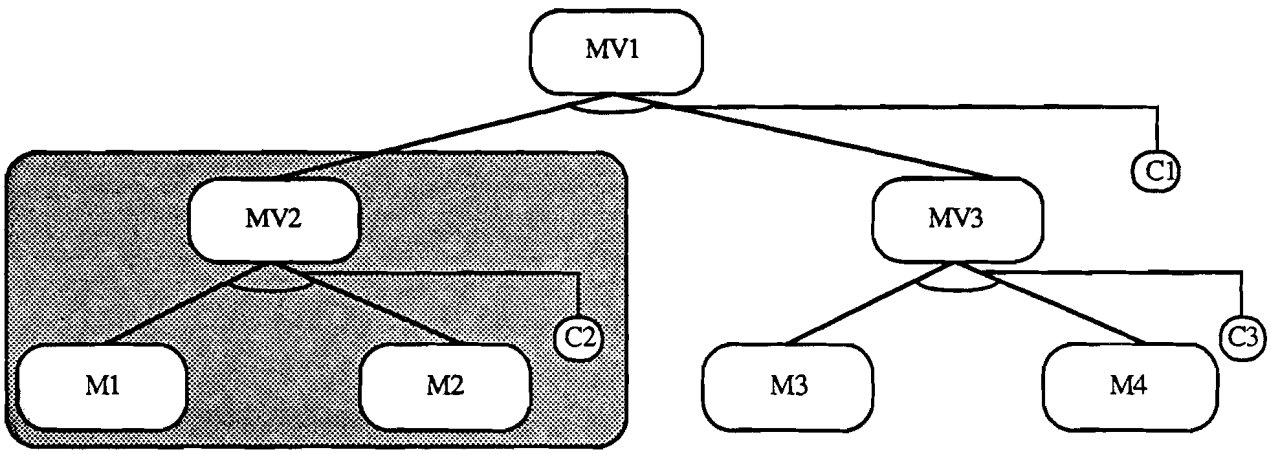



fig 47 : arbre initial

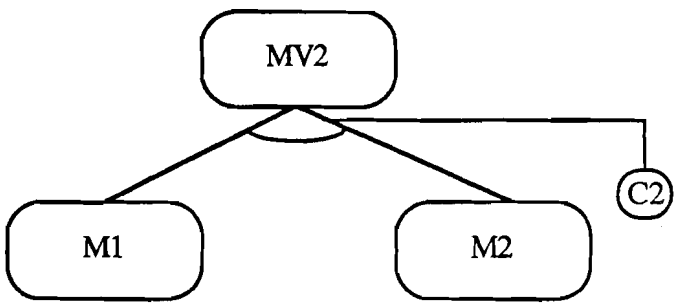



fig 48 : arbre réduit

Le nouvel arbre obtenu a pour racine MV2.

### V . 4 . 4 - Ecriture des méta-règles

Les méta-règles seront construites afin de respecter ces réductions de graphes et d'arbres. Il s'agit donc de retirer toutes les règles correspondant aux arcs et aux branches supprimés respectivement, lors du passage du graphe initial au graphe réduit et de l'arbre initial à l'arbre réduit.

Le formalisme utilisé est :

## r1

**SI** l'état_de_départ = état i **ALORS** retirer les règles non concernées

Comme il y a autant de méta-règles qu'il y a d'états possibles, nous les agrégeons en une seule par machine concernant la réduction du graphe. C'est une fonction qui est déclenchée une seule fois par inférence sur la base des méta-règles et qui d'après l'état de chaque machine du système de production enlève les règles nécessaires en accord avec la méthode de réduction.

## r2

**SI** machine_citée = machine **ALORS** retirer les règles non concernées

### V . 4 . 5 - Exemple de l'atelier

Nous reprenons l'arbre de recherche, correspondant au problème (?arrêt). Le but à résoudre est par exemple (?arrêt Cell1).

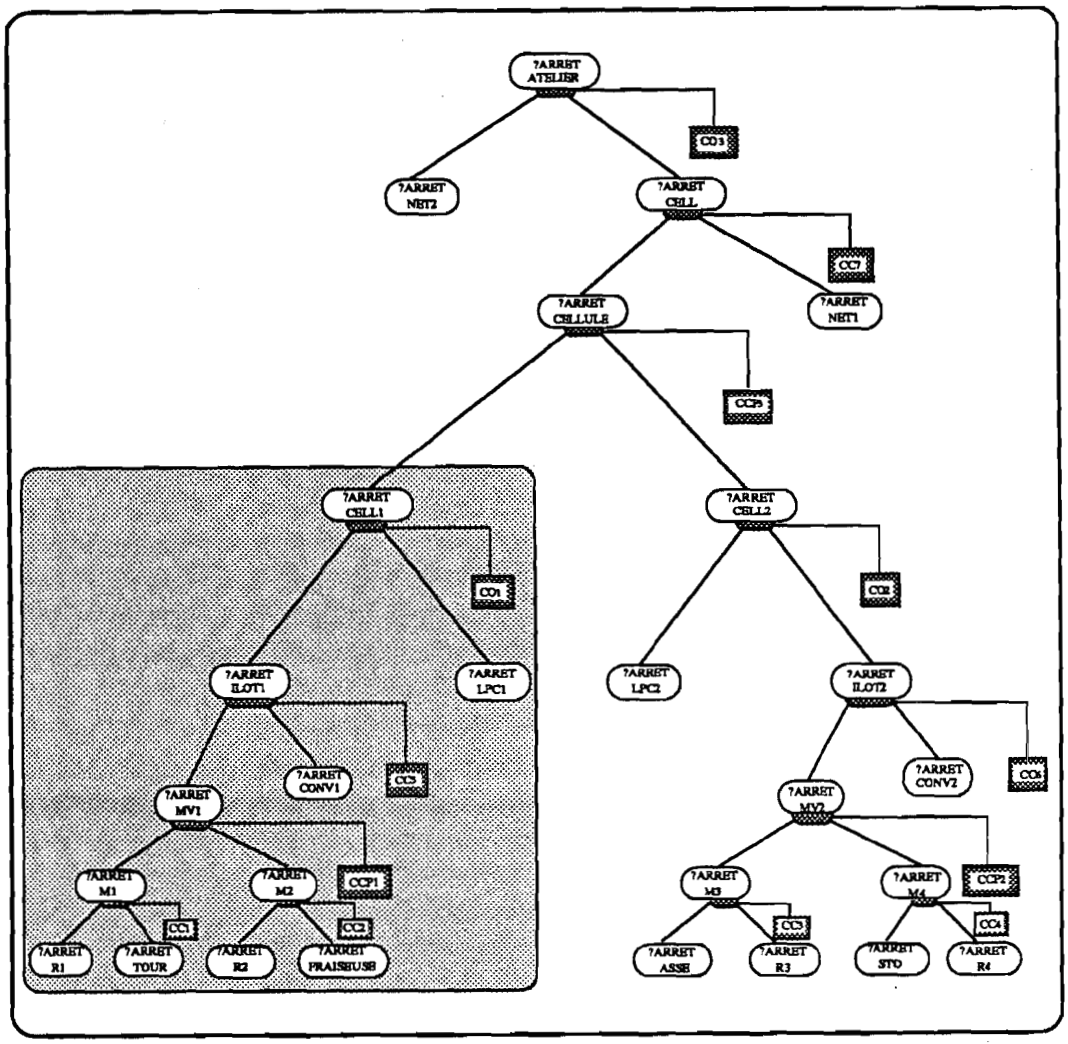



fig 49 : arbre initial

Le nouveau sommet de l'arborescence devient donc (?arrêt Cell1) et le reste de l'arborescence est retiré.

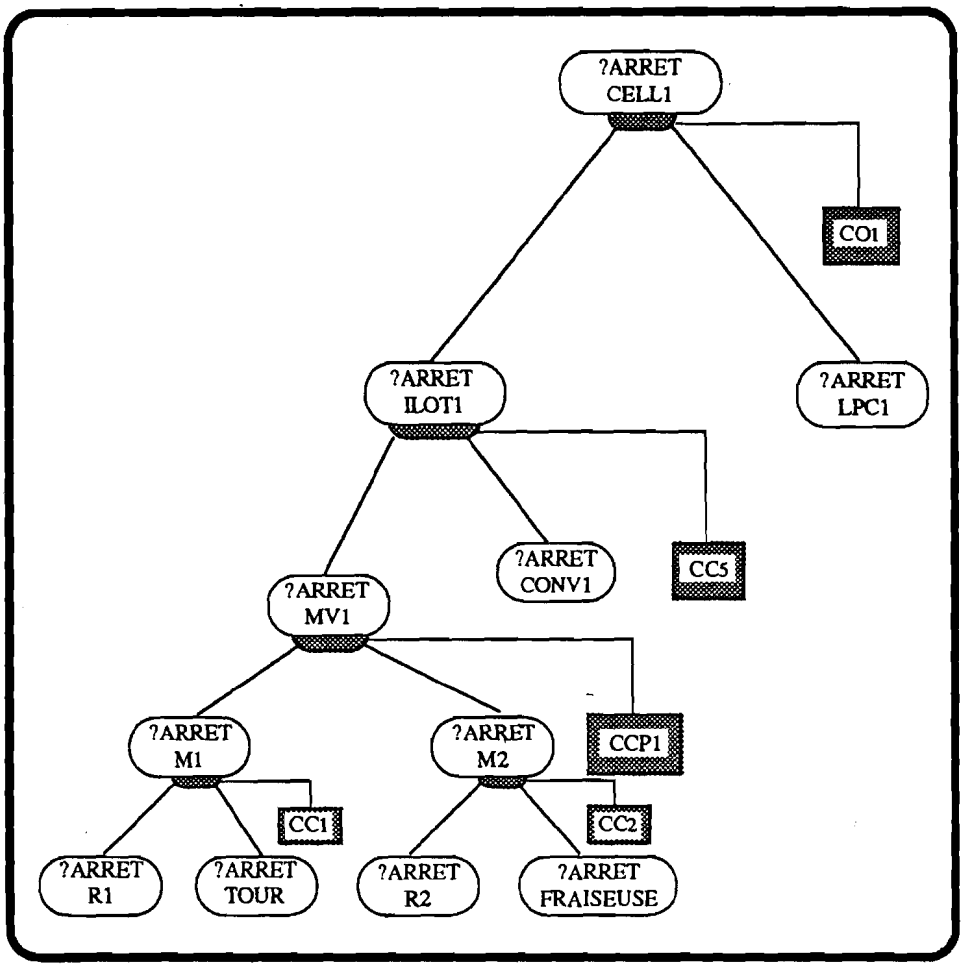



fig 50 : arbre réduit

### V . 5 - Nombre de règles

Nous pouvons estimer le nombre de règles utilisées. Dans le cas d'une machine dont le comportement est modélisé par le graphe d'état, il faut en moyenne 30 règles pour le chaînage arrière, 2 règles pour le chaînage avant et 2 méta-règles.

### V . 6 - Bilan de mise en oeuvre

Le prototype dans un premier temps, a été testé complètement déconnecté du procédé. Par une mise à jour artificielle de la base de données, nous avons vérifié tout d'abord la bonne génération des buts, et dans un second temps le bon séquencement des actions (qui consistaient alors en un simple affichage d'un message à l'écran).

Dans une deuxième phase, le prototype a été connecté à quelques machines pour vérifier l'ensemble du cycle de communication.

La mise en place définitive est en voie de réalisation. Le principal travail réside dans l'écriture des programmes intermédiaires sur automates, pour la collecte des différents états des machines.

Le système de pilotage développé est réellement autonome et capable de prendre des décisions, et n'est donc pas un simple tableau de bord. L'avantage de la structure logicielle adoptée permet la centralisation des informations essentielles, au niveau du gestionnaire. Les relations homme/machines sont simplifiées et limitées au dialogue du superviseur, mais autorise l'accès à toutes les données du procédé de manière transparente.

**Remarque** : il faut distinguer le cas des machines "ouvertes" industrielles et de celui de celles "fermées" de type micro-ordinateur. Ce sont dans ces situations que nous pouvons voir l'avantage des machines spécifiques. Intégrer des machines non dédiées au contexte industriel reste un problème difficile.

**Problèmes liés à la lecture et à l'envoi des informations** : les informations essentielles qui nous concernent sont celles qui montrent l'état des machines supervisées. En général, la situation contrôlée par un automate est aisée à connaître. Ce dernier comporte toujours des variables accessibles en lecture qui permettent de déterminer précisément l'état à un moment donné. Dans la configuration actuelle, l'absence de liaison directe GPX/machines (NUM, robots) de type réseau, nous oblige à systématiquement utiliser un code intermédiaire sur les TSX pour la communication avec le procédé. L'installation d'un réseau UNI-TELWAY permettra de s'en affranchir : un système de requêtes, communes au réseaux TELWAY 7 et UNI-TELWAY est mis en oeuvre, dont le nombre et la variété permettent de répondre à nos besoins. Citons en particulier les requêtes (sources TELEMECANIQUE) :

- **d'accés aux données**
  - * lecture d'un bit/mot,
  - * écriture d'un bit/mot,
- **d'usage général**
  - * status (renseigne sur l'état d'un équipement),
  - * transferts de fichiers,
- **de modes de marche**
  - * RUN (mise en RUN d'un équipement),
  - * STOP (mise en STOP d'un équipement),
  - * AUTOTEST (Autotest d'un équipement),
  - * INIT (initialisation d'un équipement).

L'utilisation de ce système de requêtes autorisera un fonctionnement plus riche en possibilités et plus rapide dans le cadre, non seulement d'une gestion de modes de marche, mais également dans celui d'une supervision classique.

### V . 6 . 1 - Problème du temps de réponse

Les constantes de temps du système sont très variables et donc imposent des obligations de temps de réponse très différentes :

- réponse immédiate dans le cas d'un dysfonctionnement (temps de l'ordre de la seconde),
- réponse différée dans le cas de la gestion d'un mode de marche d'une machine, dont les temps d'usinage peuvent être de l'ordre de la minute.

# CONCLUSION

Nous avons présenté l'exemple de contrôle et de supervision d'une unité de production flexible. Nous en avons décrit les différentes fonctionnalités et les différentes solutions adoptées pour résoudre tous les problèmes de fonctionnement.

Cette réalisation a permis de valider deux démarches conjointes :

- celle concernant le projet CASPAIM, qui aboutit à l'implantation systématique du modèle RdPSAC, en intégrant les notions de flexibilité et de supervision,
- celle proposée dans le rapport pour la gestion des modes de marche avec la présentation d'un prototype. Nous avons montré comment nous intégrons ce gestionnaire dans le système général de pilotage et la place primordiale qu'il occupe vis à vis des autres tâches de conduite. L'étude des modes de marche autorise un fonctionnement rigoureux et autonome du système de production.

Tout au long de ce chapitre, nous nous sommes intéressés à l'implantation des différents niveaux de commande d'une unité de production flexible, la partie commande et le niveau hiérarchique et leurs associations. Ceci a justifié la décomposition en plusieurs niveaux des ressources informatiques, qui permet d'obtenir un système de pilotage structuré et cohérent. L'utilisation de différents outils informatiques a été réalisée d'un point de vue CIM, avec l'informatisation et l'automatisation des différentes fonctions de pilotage, garantissant un haut niveau de disponibilité.

**L'implantation de la partie commande** (répartition et modèle correct de représentation) a été résolue, par l'utilisation du Langage Grafcet. Le modèle issu de la transposition RdPSAC/GRAFCET se présente comme facilement exploitable et implantable. De plus, l'originalité de notre approche montre que nous avons pu intégrer du matériel complètement hétérogène, mettant ainsi en évidence l'importance des outils de communication.

**L'élaboration du niveau hiérarchique,** seconde étape de notre démarche, est basée sur la coopération de plusieurs modules, chacun dédié à une tâche particulière, coordonnés par le prototype de gestionnaire de modes de marche. De plus, celui-ci gère le fonctionnement des différentes machines. Il en résulte une commande flexible et disponible.

La conception du gestionnaire repose sur une approche originale qui est l'exploitation de la modélisation structuro-fonctionnelle du processus de fabrication, mettant en évidence ses diverses contraintes de fonctionnement général, pour assurer la gestion des modes de marche.

De plus, l'identification du comportement individuel de chaque machine sous forme de graphes d'état autorise la génération automatique des règles destinées à la base de connaissance du gestionnaire. La traduction systématique de la modélisation en règles garantit la cohérence et la validité de la base de connaissance, en supposant les spécifications initiales correctes.

La difficulté de cette méthode est qu'elle demande une connaissance profonde du procédé et de son système de commande.

Les spécifications de notre gestionnaire sont suffisantes pour valider le principe de la démarche et l'utilisation d'un système expert. Nous avons décrit les principales fonctionnalités que requière l'élaboration de notre gestionnaire. Des techniques très intéressantes d'un point de vue prospectif sont celles qui utilisent des notions telles que la logique floue ou le raisonnement hypothétique.

Une évolution intéressante serait la construction d'une base de connaissance avec une approche objet [AMA 90], parfaitement appropriée.

Par ailleurs, **un moteur d'ordre 1** avec l'introduction de variables aurait permis une réduction importante du nombre de règles. La construction d'arbres identiques est parfaitement appropriée et suggère en effet un moteur d'ordre 1. Cependant la spécificité du comportement d'une machine rend difficile la généralisation et la systématisation des règles. Ajoutons que si le nombre de règles est considérablement diminué, le temps d'analyse et de calcul serait par contre beaucoup plus long, par l'explosion combinatoire qui risquerait d'être engendrée.

# CONCLUSION GENERALE

Nous avons présenté et mis en évidence une méthode de gestion des modes de marche dans l'exploitation d'un système automatisé. Nous en avons montré la répercussion sur les différents niveaux de commande. Cela nous a conduit à proposer l'élaboration d'un logiciel de pilotage, chargé de la gestion automatique des modes de marche.

La démarche comporte deux étapes importantes :

- la modélisation du procédé,
- l'exploitation de la modélisation.

La démarche générale nous a permis, dans un premier temps, de construire un modèle du procédé, par une analyse ascendante des contraintes et des liens de fonctionnement entre machines. De plus, la liste exhaustive des contraintes de fonctionnement, contraintes sur état ou sur changement d'état, a été spécifiée et introduite par la construction de tables. Cette technique de modélisation permet, de manière systématique, d'aboutir à **un modèle arborescent du système de production**. La représentation structurelle fournie met en évidence les différents niveaux de commande du système et fait ressortir les principes d'organisation et de conduite des installations. Par ailleurs, le comportement individuel de chaque machine, effective ou virtuelle, est décrit à l'aide de graphes d'état.

Dans une seconde démarche, nous avons montré que l'intégration des tables de contraintes aboutit à une décomposition structurelle et fonctionnelle : elle est obtenue sous la forme d'un schéma arborescent de recherche déductive des modes de marche des différentes machines composant le procédé. Ce modèle est facilement exploitable pour être utilisé par la suite, en vue du pilotage temps réel de l'unité de production par un raisonnement sur la logique de dépendance ou d'enchaînement de ses modes de marche.

L'originalité de cette approche réside dans la possibilité d'intégrer tous les composants possibles d'une unité de production, y compris les réseaux informatiques, généralement très peu pris en compte, à ce niveau de préoccupation, dans les architectures de pilotage.

Nous avons également indiqué comment l'exploitation de la modélisation permet de générer la base de règles destinée au gestionnaire. Cette phase de traduction est basée sur la transcription directe de l'arborescence en un arbre de recherche avec intégration des tables de contraintes. La traduction de cette arborescence et des graphes d'état associés, sous forme de règles, constitue un résultat original. En effet, si la structure même des graphes d'états est difficilement exploitable directement, l'assistance d'un système expert permet d'en exploiter toutes les potentialités. De plus, la possibilité de modifier rapidement la base de connaissances

est un des points forts du système de pilotage et des systèmes experts en général par rapport à une programmation classique. Les techniques de l'intelligence artificielle prennent ici tout leur intérêt en permettant une conduite de processus de qualité avec la prise en compte de connaissances diverses et des contraintes temps réel avec ordonnancement implicite des tâches.

Un exemple de gestionnaire de modes de marche, prototype basé sur l'utilisation de systèmes experts a été présenté. Ce dernier a permis de valider complètement la démarche, avec l'imbrication de deux moteurs d'inférence, l'un destiné à un raisonnement déductif, l'autre destiné à un raisonnement planificateur. Ce prototype de gestionnaire fait partie des "petits systèmes", avec un nombre de règles inférieur à 500. Il permet toutefois de valider l'approche proposée.

Nous avons de plus, préservé l'idée d'une séparation des fonctions de la partie commande de celles du niveau hiérarchique en gardant la possibilité de les associer. Les différentes fonctions de pilotage sont informatisées d'un point de vue CIM et conduisent à l'obtention d'une commande flexible, facilement configurable et donc adaptable. Cela justifie la décomposition en plusieurs niveaux, des ressources de commande. L'intérêt d'une telle approche est de pouvoir simultanément tenir compte du produit et des machines en cours d'exploitation :

- **la partie commande** assure la gestion individuelle des produits lors de la production,
- **le niveau hiérarchique** gère, quand à lui, le fonctionnement et le pilotage d'ensemble des machines. Dans notre démarche, nous n'avons donc pas considéré le produit comme contrainte supplémentaire dans la gestion des modes de marche. Ceci demeure difficile à résoudre de façon générale, du point de vue de la modélisation.

Des perspectives d'amélioration sont envisageables, d'une part, pour améliorer l'outil informatique, et d'autre part pour affiner le modèle.

De manière prospective, il nous apparait que l'emploi de techniques informatiques telles que les langages objets et l'usage de moteurs d'ordre 1, enrichiraient les capacités du gestionnaire de modes de marche, en facilitant le cadre de modélisation.

En effet, une approche objet est plus propice au prototypage, et semble mieux adaptée à la description naturelle d'un système complexe. Les différentes machines et composants de l'unité de production sont alors décrits par des objets possédant leurs propriétés de comportement : des actions ou opérations sont réalisables par les objets. Les contraintes de fonctionnement sont intégrées aux caractéristiques de comportement des objets. Une description objet offre de plus,

des possibilités intéressantes de simulation afin de vérifier le bon fonctionnement général. Le modèle est alors utilisable non seulement en vue de l'exploitation mais aussi à des fins de conception.

La réalisation de moteurs d'ordre 1 est également possible dans ce cadre. Le caractère générique des graphes de comportement et des arborescences autorise l'emploi de variables lors du passage aux règles. Leur introduction induit une réduction intéressante du nombre de règles.

Par ailleurs, l'enrichissement du modèle de base par la prise en compte du produit et de son état constitue une autre perspective de recherche intéressante. Une solution possible, dans le cas de processus manufacturiers, serait de considérer l'état du produit et de l'associer à celui de la machine, au même titre que ceux de ses composants. Les modes de marche d'une machine de fabrication seraient influencés par l'état de son produit en cours de traitement. En effet, les procédures de reprise après une panne en cours de fabrication, ne sont pas toujours identiques et dépendent fortement de l'état du produit présent sur la machine au moment de la reprise :

- si le produit a été endommagé ou altéré, il faut prendre des mesures spéciales, souvent manuelles, pour le retirer de la machine ou le réparer,
- si le produit est intact, une simple procédure de reprise à chaud est suffisante pour revenir dans un régime normal de production.

Ces apports constituent des axes intéressants et prospectifs de développement, dans le cadre de l'exploitation d'un système de production.

# REFERENCES

## CHAPITRE 1

[AND 89] ANDRE C., FANCELLI L.
"Etude d'une réalisation mixte (asynchrone/synchrone) d'un système temps-réel"
Rapport interne, RR 89-3, Lassy, Nice, Mai 1989.

[AFC 77] AFCET
"Normalisation de la représentation du cahier des charges d'un automatisme logique"
Rapport final de la commission AFCET, Août 1977.

[AFN 87] AFNOR
"Recommandation de plan qualité logiciel"
Z67-130, AFNOR, 1977.

[BEN 86] BENASSY J. ET ALL
"L'usine intégrée par ordinateur"
Ed. HERMES, collection gestion et productique, 1986.

[BEN 85] BENCHIMOL G.
"La conception des usines de demain"
Ed. HERMES, 1985.

[BER 86] BERRY G, COSSERAT L.
"The synchronous programming language Esterel and its mathematicals demantics"
Rapport I.N.R.I.A. 327,1986.

[BNI 84] Bureau d'orientation de la Normalisation en Informatique
SCEPTRE, 1984.

[BOI 88] BOISSONNAT J.D., FAVERJON B., MERLET J.P.
"Techniques de la robotique"
Ed. HERMES, tome 1, architectures et commandes, 1988.

[BON 85] BONETTO R.
"Les ateliers flexibles de production"
Ed. HERMES, 1985.

[BOU 88] BOUREY J.P.
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, 1988.

[BRA 83] BRAMS G.W.
"Réseaux de Petri : théorie et pratique"
Tome 2, modélisation et applications, Ed. MASSON,1983.

[CAS 87] CASTELAIN E.
"Modélisation et simulation interactive de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, Février 1987.

[CHA 87] CHANTREUIL ET AL
"Les automatismes programmables"
CEPADUES-édition, 1987.

[CHE 90] CHEN D., VALLESPIR B., ZANETTIN M.
"Architecture globale CIM : application dans le projet ESPRIT-IMPACS"
CIM 90, Bordeaux.

[CNR 88] CNRS
Rapport établi par le groupe de réflexion Temps réel du CNRS
TSI Volume 7 n°5, 1988.

[COL 91] COLLOMBIER V.
"Le concept CIM"
Document BULL, Mars 91.

[COM 89] COMPEX
Présentation des réseaux LAC/LAC2, Mars 89.

[COR 79] CORBEEL D.
"Schéma de cablage et schéma de contrôle. Application à la simulation et à la gestion des processus industriels"
Thèse de Doct. de Spécialité, LILLE, 1979.

[COR 80] CORBEEL D.
"Formal description of processes systems and exception handling"
Mini&Micro, Proc.pp.335 à 339, BUDAPEST, Septembre 1980.

[COR 85] CORBEEL D., VERCAUTER C., GENTINA J.C.
"Application of an extension of Petri nets to modelization of control and production processes"
Sixth European Workshop on application and theory of Petri nets, pp.53 à 74, Juin 1985.

[CRA 89] CRAYE E.
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, Janvier 1989.

[CRU 91] CRUETTE D.
"Méthodologie de conception des systèmes complexes à événements discrets : application à la conception et la validation hiérarchisée de la commande de cellules flexibles de production dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, Février 1991.

[DAV 89] DAVID R., ALLA H.
"Du Grafcet aux réseaux de Petri"
Ed. HERMES, Traité des Nouvelles Technologies, série Automatique, 1989.

[DOU 84] DOUMEINGTS G.
"Méthode Grai : méthode de conception des systèmes en productique"
Thèse de Doct. d'état ès sciences, Bordeaux, 1984.

[ENG 81] ENGELBERGER J.F.
"Les robots industriels"
Ed. HERMES, applications, gestion et pratique, 1981.

[FRO 84] FROMENT B., LESAGE J.L.
"Productique : les techniques de l'usinage flexible"
Ed. DUNOD, Série Génie Mécanique, 1984.

[GAC 90] GACHES R., QUERENET B., VIOLLET P., VERNADAT F.B.
"CIM-OSA : une architecture ouverte pour la productique"
Congrès CIM 90, Bordeaux, Juin 1990.

[GAS 89] GASNIER B.
"Sur un outil interactif de spécification de gammes opératoires en production flexible manufacturière"
Thèse de Doct. de l'Université, LILLE, Janvier 1989.

[GER 83] GERWIN D.
"A framework for analysing the flexibility of manufacturing processes"
School of Business and Administration
University of WISCONSIN, MILWAUKEE USA, 1983.

[GOR 87] GORISSE J.P.
"Automatisation du suivi de production dans un atelier de galvanisation"
Thèse de Doct. Ingénieur, spécialité : Automatique, LILLE, Novembre 1987.

[HAM 91] HAMMADI S., CASTELAIN E. , BORNE P.
"Efficient and feasible Job-Schop Scheduling in a Flexible Manufacturing System"
IMACS, Toulouse, Mars 1991.

[HAR 87] HAREL D.
"Statecharts : a visual formalism for complex systems"
Science of computer programming 8, p231, 274, 1987.

[ING 88] INGERSOLL
"L'usine intégrée"
Ed. HERMES, 1988.

[KAP 88] KAPUSTA M.
"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 Doct. de l'Université, LILLE, 1988.

[KOR 86] KOREN Y.
"Robotique pour Ingénieurs"
Ed MCGRAW-HILL, 1986.

[LEP 89] LEPAGE F., AFILAL F., ANTOINE P., BAJIC E., BRON J.Y., DIVOUX T.
"Les réseaux locaux industriels"
Ed. HERMES, Traité des Nouvelles Technologies, série Automatique, 1989.

[MOA 85] MOALA M.
"Réseaux de Petri interprétés et Grafcet"
TSI, VOL 4, N°1 - 1985, Ed. DUNOD.

[MOR 90] MOREL G., LHOSTE P., ROESCH M.
"Automatisation intégrée d'un ilôt de fabrication manufacturière"
Congrès CIM 90, Bordeaux, Juin 1990.

[NUS 88] NUSSBAUMER H.
"Informatique Industrielle IV"
PRESSES POLYTECHNIQUES ROMANDES, 1988.

[ROS 77] ROSS D.T.
"Structured Analysis : A language for communicating ideas"
IEEE Transactions on software engineering, vol. 3, n°1, 1977.

[SIM 72] SIMON W.
"Amélioration de la productivité"
Ed. EYROLLES,1972.

[TIX 89] TIXADOR J.M.
"Une contribution au Génie Automatique : la Spécification EXécutable des Machines et Systèmes Automatisés de Production"
Thèse de l'université de NANCY I, 1989.

[VER 91] VERON M.
"Méthodes et outils d'intégration. Expériences du CRAN-LACN"
32ème CIRP, NANCY, Juin 1991.

[VEK 89] VEKEMANS E.
Rapport de stage chez BULL, IDN, 1989.

## CHAPITRE 2

[ADE 79] ADEPA
Le GEMMA "guide des modes de marche et d'arrêt", 1979.

[BER 86] BERRY G., COSSERAT L.
"The synchronous programming language Esterel and its mathematicals demantics"
Rapport I.N.R.I.A. 327, 1986.

[BOI 91] BOIS S., CRAYE E., GENTINA J.C.
"Gestionnaire de mode de marche"
32ème CIRP, NANCY, Juin 1991.

[CHA 89] CHAILLOUX J.
"Le_Lisp, Manuel de référence"
I.N.R.I.A, 1989.

[CHA 87] CHANTREUIL ET AL
"Les automatismes programmables"
CEPADUES-édition, 1987.

[CNR 88] CNRS
Rapport établi par le groupe de réflexion Temps réel du CNRS
TSI Volume 7 n°5, 1988.

[COM 89] COMPEX
Présentation des réseaux LAC/LAC2, Mars 89.

[FAR 87] FARRENY H., GHALLAB M.
"Eléments d'intelligence artificielle"
Ed. HERMES, 1987.

[GEN 90] GENTINA J.C., BOIS S., DANCOISNE D., TOGUYENI A.K.A.
"Contrat d'études IBM/IDN, Rapport final", Octobre 1990.

[LEM 77] LEMOIGNE J.L.
"Théorie du système général", 1977.

[LEP 89] LEPAGE F., AFILAL F., ANTOINE P., BAJIC E., BRON J.Y., DIVOUX T.
"Les réseaux locaux industriels"
Ed. HERMES, Traité des Nouvelles Technologies, série Automatique, 1989.

[LHO 85] LHOSTE P.
"EX.A.O ou Exploitation Assistée par ordinateur"
Thèse de l'université de NANCY I, 1985.

[OBS 89] OBSERVATOIRE FRANÇAIS DES TECHNIQUES AVANCEES
"Systèmes experts et conduite de processus"
Ed MASSON, 1989.

[PAR 91] PARAYRE T., SALLEZ Y., SOENEN R.
"Etude des cycles de vie des S.A.P. vers une méthodologie de l'exploitation"
32ème CIRP, NANCY, Juin 1991.

[RIC 87] RICH E.
"Intelligence artificielle"
Ed. MASSON, 1987.

[ROD 89] RODDE G.
"Les systèmes de production ; Modélisation et Performances"
Ed. HERMES, 1989.

[TEL 87] TELEMECANIQUE
"Modes de marches et d'arrêts"
Manuel TELEMECANIQUE, 1987.

## CHAPITRE 3

[AMA 90] AMAR S., CASTELAIN E., GENTINA J.C.
"Modélisation des moyens de production par langages orientés objet en vue de la conception de la commande d'un système de production flexible"
Congrès CIM 90, Bordeaux, Juin 1990.

[BEN 86] BENCHIMOL G., LEVINE P., POMEROL J.C.
"Systèmes-experts dans l'entreprise"
Ed. HERMES, 1986.

[BOI 90] BOIS S., CRAYE E., GENTINA J.C.
"Aided design and implementation of the scheduling level of control and the hierarchical level of a flexible manufacturing system"
ISATA, 22nd Symposium, Florence, Italie, Mai 1990.

[BON 86] BONNET A., HATON J.P., TRUONG-NGOC J.M.
"Systèmes-experts : vers la maîtrise technique"
InterEditions, 1986.

[BOU 88] BOUREY J.P.
"Structuration de la partie procédurale du système de commande de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, 1988.

[CHA 89] CHAILLOUX J.
"Le_Lisp, Manuel de référence"
I.N.R.I.A, 1989.

[CRA 89] CRAYE E.
"De la modélisation à l'implantation automatisée de la commande hiérarchisée de cellules de production flexibles dans l'industrie manufacturière"
Thèse de Doct. de l'Université, LILLE, Janvier 1989.

[DAN 91 ] DANCOISNE D.
Mémoire Ing. CNAM., Centre Régional Associé de Lille, Juin 1991.

[FAR 87] FARRENY H., GHALLAB M.
"Eléments d'intelligence artificielle"
Ed. HERMES, 1987.

[GEN 90] GENTINA J.C., BOIS S., DANCOISNE D., TOGUYENI A.K.A.
"Contrat d'études IBM/IDN, Rapport final", Octobre 1990.

[GIA 89] GIANNOULIS Y.
"Communications TSX47-PS/2 et PS/2-controleur d'un robot IBM, par liaison RS232"
Rapport de stage, IDN, 1989.

[HUV 90] HUVENOIT B.
"Réalisation de la communication entre une station de travail GPX et un automate programmable TSX"
DEA de Productique, IDN, 1990.

[IPP 91] IPPOLITO S., PHILIPPE H.
"Aide à la conduite d'un processus industriel au moyen d'un système expert"
APII-1991-24-189-214, vol 24 N°3, 1990.

[MAY 89] MAYET J.
"Sur la conception d'un atelier flexible de fabrication mécanique"
Mémoire Ing. CNAM., Centre Régional Associé de Lille, Juin 1987.

[MAN 87] BUS UNITELWAY
"Manuel de références"
05.1989, TSX D24 0004F.

[OBS 89] OBSERVATOIRE FRANÇAIS DES TECHNIQUES AVANCEES
"Systèmes experts et conduite de processus"
Ed. MASSON, 1989.

[RES 85] RESEAU TELWAY
"Utilisation, mise en oeuvre"
Manuel TELEMECANIQUE, Juin 1985.

[TOG 90] TOGUYENI A.K.A., CRAYE E., GENTINA J.C.
"A method of temporal analysis to perform on-line diagnosis in the context of flexible manufacturing system"
Proceedings of IECON' 90, Vol. 1, pp. 445-450, Pacific Grove-California, November 90.

[VOY 87] VOYER R.
"Moteurs de systèmes experts"
Ed. EYROLLES, 1987.

## RESUME

La difficulté d'exploitation des systèmes automatisés de production résulte des possibilités de plus en plus riches d'évolution de leurs différentes entités : machines, produit...

Nous nous sommes ainsi intéressés à la conduite effective des machines en gérant rigoureusement leurs différents modes de marche. L'intérêt est de prendre en compte toutes les contraintes apparaissant lors de l'exploitation d'une unité de fabrication.

La démarche générale nous a permis, dans un premier temps, de construire un modèle du procédé, par une analyse ascendante des contraintes de fonctionnement entre machines. La liste exhaustive des contraintes, sur état ou sur changement d'état, a été spécifiée et introduite par la construction de tables. Cette technique de modélisation permet systématiquement, d'aboutir à un modèle arborescent du système de production avec la mise en évidence de ses différents niveaux de commande. Par ailleurs, le comportement individuel de chaque machine, effective ou virtuelle, est décrit à l'aide de graphes d'état.

Dans un second temps, nous avons montré que l'intégration des tables de contraintes aboutit à une décomposition structurelle et fonctionnelle : elle est obtenue sous la forme d'un schéma arborescent de recherche déductive des modes de marche des différentes machines du procédé. Ce modèle est facilement exploitable pour être utilisé par la suite, en vue du pilotage temps réel de l'unité de production par un raisonnement sur la logique de dépendance ou d'enchaînement de ses modes de marche.

L'exploitation de la modélisation nous a amené à étudier le pilotage d'une unité de production flexible. Nous en avons décrit les différentes fonctionnalités et les solutions adoptées pour résoudre tous les problèmes de fonctionnement, qui se répercutent à tous les niveaux de commande. La mise en oeuvre d'un logiciel de pilotage, chargé de la gestion automatique des modes de marche, a permis de valider complètement la démarche, avec l'utilisation de techniques d'intelligence artificielle.



## MOTS-CLEFS

CIM

SYSTEME AUTOMATISE DE PRODUCTION

MODES DE MARCHE

MODELISATION

GRAPHE D'ETAT

PILOTAGE

NIVEAU HIERARCHIQUE

SYSTEME EXPERT