N° d'ordre : 405

50376
1987
3

THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR INGENIEUR

Spécialité : Automatique

par

Rabah HACHEMANI

INGENIEUR IT ORAN

CONTRIBUTION A LA SURETE DES AUTOMATISMES SEQUENTIELS: MODELISATION, IDENTIFICATION ET TEST EN LIGNE DE LA PARTIE OPERATIVE

Soutenue le 15 janvier 1987 devant la commission d'examen

Membres du jury: MM.

P. VIDAL	Président
J.M. TOULOTTE	Rapporteur
J. DEFRENNE	Rapporteur
M. STAROSWIECKI	Examinateur
PRUNNET	Examinateur
DHOOGE	Examinateur

e mémoire est dédié

à mes parents,

à ma femme,

à mes enfants: Rachid, Samir et Farid

AVANT-PROPOS

Le travail présenté dans ce mémoire a été effectué au **Laboratoire d'Automatique** de l'Université des Sciences et Techniques de Lille Flandres Artois.

Que Monsieur **P. VIDAL**, Professeur à l'université de Lille Flandres Artois, trouve ici l'expression de ma reconnaissance pour m'avoir accueilli dans son laboratoire et pour avoir bien voulu accepté la présidence de mon jury de thèse.

Que Messieurs **J.M. TOULOTTE**, Professeur à l'université de Lille, et **J. DEFRENNE**, Maître de conférence, trouvent ici l'expression de mes plus vifs remerciements pour m'avoir guidé avec tant de clairvoyance tout au long de cette étude et pour la confiance témoignée.

Je remercie vivement Monsieur **M. STAROSWIECKI**, Professeur à l'E.U.D.I.Lille pour l'honneur qu'il me fait pour avoir bien voulu accepté de juger ce travail.

Je remercie également Monsieur **F. PRUNET**, Professeur à l'U.S.T.Languedoc et Monsieur **M. DHOOGE**, Ingénieur Télémécanique Electrique pour l'honneur qu'ils me font en voulant bien examiner ce travail.

Je tiens à remercier également les **gouvernements français et algérien** qui m'ont permis, par leur soutient financier, d'entreprendre ces travaux.

SOMMAIRE

CHAPITRE III: MISE EN OEUVRE ET IMPLANTATION DU MECANISME DE TEST EN LIGNE.

CHAPITRE IV: APPRENTISSAGE AUTOMATIQUE DU MODELE DE LA PO.

CHAPITRE V: IDENTIFICATION DE MACHINES SEQUENTIELLES

INTRODUCTION GENERALE

Devant l'évolution, ces dernières années, des systèmes de production automatisés de plus en plus complexes, les problèmes de sûreté de fonctionnement et en particulier de sécurité prennent une importance de plus en plus accrue.

Les implications socio-économiques des défaillances de tels systèmes peuvent être particulièrement néfastes. Une surveillance du fonctionnement du système permettant la détection des pannes est nécessaire afin de réagir et limiter les conséquences des défaillances.

Notre étude est une contribution à l'amélioration de la sûreté de fonctionnement en particulier de la sécurité des systèmes de production automatisés à évolution séquentielle. Une procédure de test en ligne de la partie opérative (PO) est proposée.

Le chapitre 1 reprend les définitions de base de la sûreté de fonctionnement des systèmes et les conclusions sur le choix d'une architecture avec dispositif de détection d'erreurs. L'étude des taux de pannes des automatismes industriels et des techniques généralement utilisées pour améliorer leur sûreté de fonctionnement, en particulier leur sécurité, fait apparaître la nécessité d'une surveillance des éléments de la partie opérative.

Le chapitre 2 étudie la modélisation du comportement de la partie opérative vis à vis de la commande. Ce modèle est ensuite utilisé pour réaliser le test en ligne de la partie opérative. Ce test prend en compte le temps normalement imparti à l'éxécution des actions élémentaires.

Le chapitre 3 présente une solution pour l'implantation du test. Une description par Grafcet est utilisée. Un language de spécification du comportement de la PO permet de générer la structure de données qui sera utilisée en temps réel.

Dans les chapitres 4 et 5 nous étudions les possibilités d'apprentissage automatique du modèle de la partie opérative en vue de générer automatiquement la structure de données. La classification des durées des actions élémentaires permet de retrouver les vitesses d'évolution ou commandes appliquées. Les résultats de la classification sont utilisés pour l'identification de la machine séquentielle constituée par la partie 'préactionneurs'.

CHAPITRE I:AUTOMATISMES INDUSTRIELS ET SURETE DE FONCTIONNEMENT

1. NOTION D'AUTOMATISME, ASPECT FONCTIONNEL

1.1. Constitution d'un automatisme.
1.2. La partie Opérative.
1.3. La partie commande.

2. SURETE DE FONCTIONNEMENT DES SYSTEMES: DEFINITIONS.

2.1. Comportement d'un automatisme en présence de fautes, terminologie.
2.2. Les paramètres d'évaluation de la sûreté de fonctionnement.
2.3. Les estimateurs de la sûreté de fonctionnement.
2.4. Systèmes à tolérance ou intolérance de pannes.
2.5. Classification des redondances.

3. EVALUATION DE LA SURETE D'UN SYSTEME REPARABLE.

3.1. Calcul des paramètres de la sûreté.
3.2. influence des différents facteurs.
3.3. Calcul des paramètres de la sûreté d'un système dans le cas où le taux de pannes du dispositif de détection d'erreurs n'est pas nul.
3.4. Remarques sur le choix d'une architecture.

4. PRINCIPALES CAUSES DE DEFAILLANCES DANS UN AUTOMATISME

5. DETECTION ET LOCALISATION DES ERREURS.

5.1. Domaine d'évolution d'un système.
5.2. Systèmes autotestables.
5.3. Principales structures autotestables.
5.4. Descriptions de quelques procédures de détection d'erreurs.

6. SECURITE DES AUTOMATISMES INDUSTRIELS.

6.1. Système sûr en présence de défaillances.
6.2. Sécurité statique et sécurité dynamique.
6.3. Description des principaux montages utilisés pour l'amélioration de la sécurité.

7. INFLUENCE DE LA SYNTHESE DE LA PC SUR LA SURETE.

7.1. Réceptivité et sensibilité d'un système.
7.2. Evaluation du comportement de la PC en présence de fautes non consistantes.
7.3. Susceptibilité par rapport à une entrée.
7.4. Validation du logiciel de commande.

CHAPITRE I

AUTOMATISMES INDUSTRIELS

ET SURETE DE FONCTIONNEMENT

1.AUTOMATISMES INDUSTRIELS, ASPECT FONCTIONNEL

1.1.constitution d'un automatisme:

Un système est conçu pour assurer une mission, visant à satisfaire un besoin. Dans le domaine de la production automatisée, cette mission porte sur un ensemble de transformations à opérer sur des objets. On peut distinguer deux classes de fonctions réalisées par deux types de sous systèmes:

- Les tâches de transformation du produit sont réalisées par la partie opérative (PO).
- La gestion de l'ordonnancement des actions ou tâches est confiée à la partie commande (PC).

Ces deux parties coopèrent par l'intermédiaire des ordres émis par la partie commande et des comptes rendus retournés par la partie opérative. Leur ensemble forme ce que nous appelons un automatisme.

Cet automatisme est placé dans un environnement avec lequel il coopère par un échange d'informations dites de conduite. L'opérateur humain pouvant assurer des tâches opératives ou de commande fait partie de cet environnement.

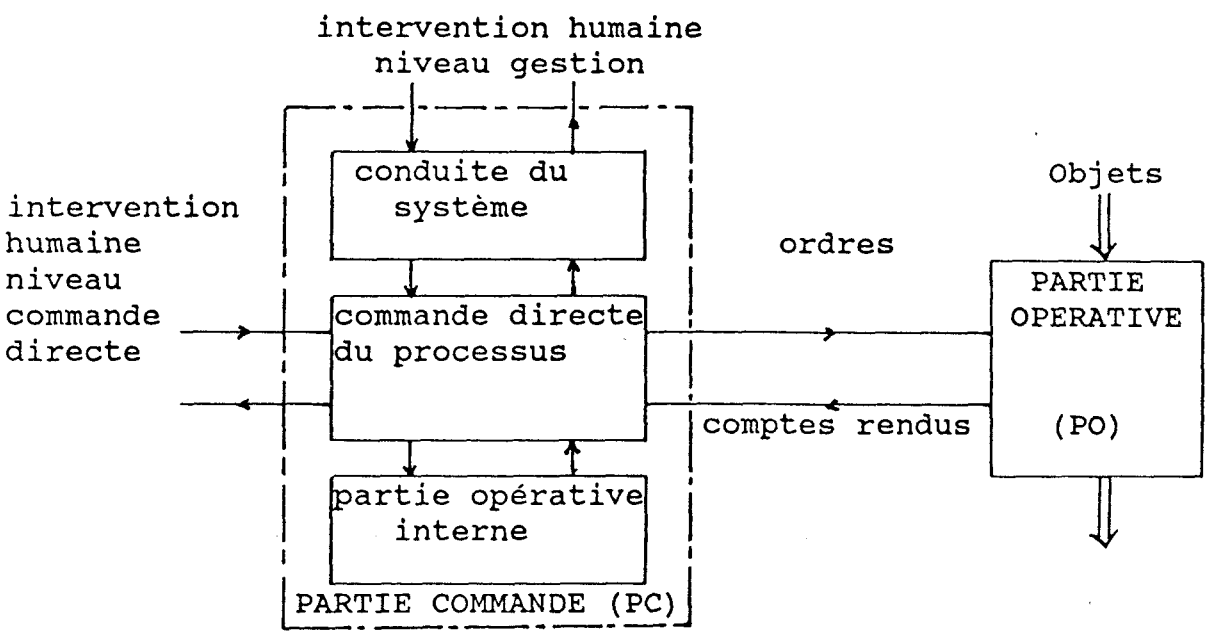


Figure I.1: Constitution d'un automatisme

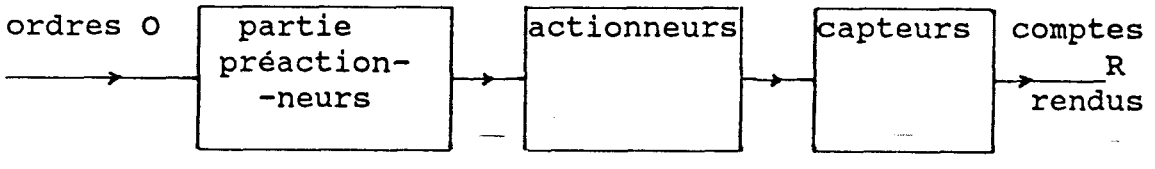
1.2.La partie opérative (fig.I.2)

Les ordres (O) élaborés par la partie commande, agissent sur des préactionneurs qui ont le rôle de moduler la puissance fournie aux actionneurs. Ils peuvent être de type tout ou rien (contacteurs, distributeurs...) ou de type analogique (redresseurs commandés, amplificateurs...). Les préactionneurs ont donc un rôle d'adaptation (technologique, puissance) mais peuvent aussi effectuer un traitement (mémorisation des ordres, priorité, etc...).

Un actionneur peut être défini comme un organe destiné à faire évoluer une grandeur physique selon un seul degré de liberté (moteurs, électro-aimant, vérins ...).

Exemple de grandeurs:
- position d'un arbre, d'une came;
- température d'une enceinte;
- ouverture d'une vanne;
- niveau d'un liquide dans un résevoir, etc...

Figure I.2: Constitution d'une Partie Opérative:



L'évolution des actionneurs est repérée par des capteurs, le plus souvent de type tout ou rien. Dans le cas de capteurs analogiques le signal est converti en valeurs numériques qui sont alors prises à l'intérieur de prédicats.

1.3.La partie commande (fig.I.3)

Elle est souvent constituée par un microcalculateur ou automate pragrammable industriel (API). On peut distinguer deux grandes parties:
- l'unité centrale ou unité de traitement comprenant le processeur de calcul et les mémoires. Elle éxécute le programme de séquencement des différentes tâches.
- Les interfaces d'entrées/sorties qui adaptent les différents signaux aux circuits externes.

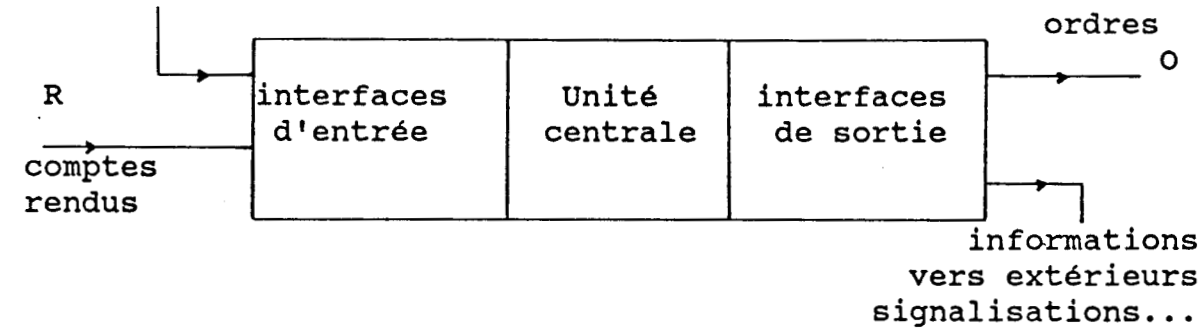


Figure I.3: La partie commande

2.SURETE DE FONCTIONNEMENT DES SYSTEMES: DEFINITIONS

2.1.Comportement d'un automatisme en présence de fautes, Terminologie: [58]

Au cours de l'exploitation d'un système, des évènements anormaux appelés pannes ou fautes peuvent survenir. Leur origine peut être liée au matériel, au logiciel ou à l'opérateur humain.

Dans le processus de dégradation d'un composant exprimé par son taux de pannes on distingue 3 étapes (figure I.4).

- La période de jeunesse où le taux de pannes va en décroissant, par l'élimination en particulier des erreurs de conception.
- La période de vie utile où le taux de pannes est pratiquement constant.
- dans la dernière période dite de vieillesse, le processus de dégradation ne cesse de s'agraver.

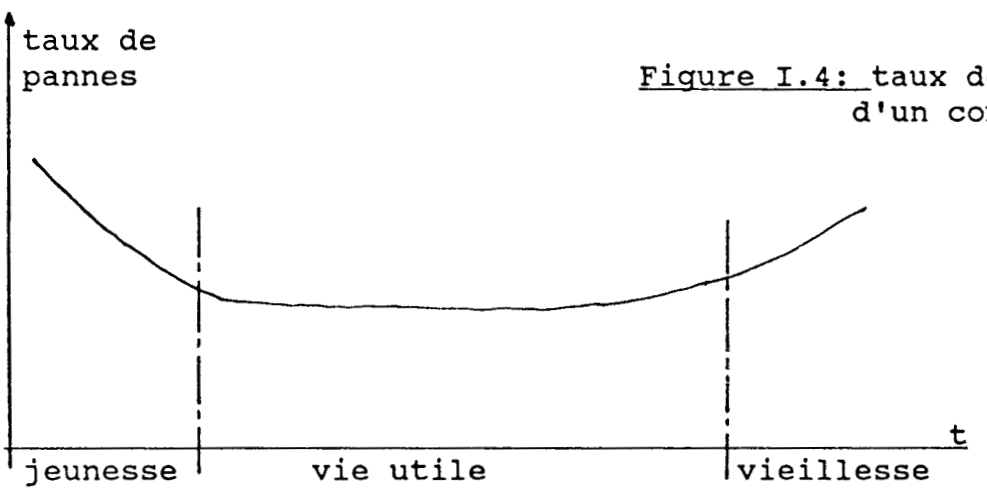


Figure I.4: taux de pannes d'un composant.

On considère en général qu'au cours de la vie utile d'un composant le taux de pannes est constant, ce qui revient à considérer une fonction de survie exponentielle.

La défaillance d'un composant peut être catalectique lorsqu'elle est soudaine et complète, ou progressive lorsque le composant perd peu à peu ses propriétés par dérive ou usure.

La faute altère les aptitudes du système dans l'accomplissement de sa mission. Il génère alors un résultat erronné ou erreur.

Exemple: le collage d'un capteur (faute) génère une information erronnée (erreur) sur la position de l'actionneur.

La prise en compte d'une erreur par un sous système, même sain, peut générer une nouvelle erreur ou même une panne. C'est le phénomène de propagation des erreurs qui peut être très néfaste.

Exemple: Le traitement par la partie commande d'une erreur sur la position d'un actionneur, suite au collage d'un capteur, génère des ordres erronnés (erreur). La prise en compte de ces ordres anormaux par la partie opérative peut provoquer une panne (rupture d'outils, etc...).

Une faute est acceptée par le système si elle ne fait décroître la probabilité de réalisation correcte d'aucune tâche. Elle est simplement tolérée si la probabilité de réalisation de toutes les tâches dans un temps acceptable est non nulle.

Une faute est dite latente ou non révélée à un instant t si l'utilisateur ignore encore son existance à cet instant. Elle est révélée, en général, par un mécanisme de détection qui constate l'erreur induite.

La réparation élimine une faute, alors que la correction ne fait que supprimer une erreur. Un mécanisme utilisant un code correcteur d'erreurs peut donc corriger une erreur mais ne supprime pas la panne. La correction peut donc éviter la propagation des erreurs.

On dit qu'une faute est consistante si sa manifestation est permanente jusqu'à ce qu'il y est réparation. Elle est dite fugitive ou non consistante dans le cas contraire.

2.2.Les paramètres d'évaluation de la sûreté de fonctionnement

On peut définir la sûreté de fonctionnement, d'une façon générale, comme l'aptitude d'un système et de son environnement à minimiser la probabilité d'apparition de défaillances et/ou à minimiser leurs effets.

Elle est caractérisée essentiellement par quatre grandeurs qui ne sont pas indépendantes les unes des autres. Ce sont les suivantes:

a) La fiabilité (Relaibility) R(t) est la probabilité de bon fonctionnement à tout instant $\sigma \in [0,t]$.

R(t) est une fonction non croissante variant de 1 à 0 sur l'intervalle $[0,\infty]$.

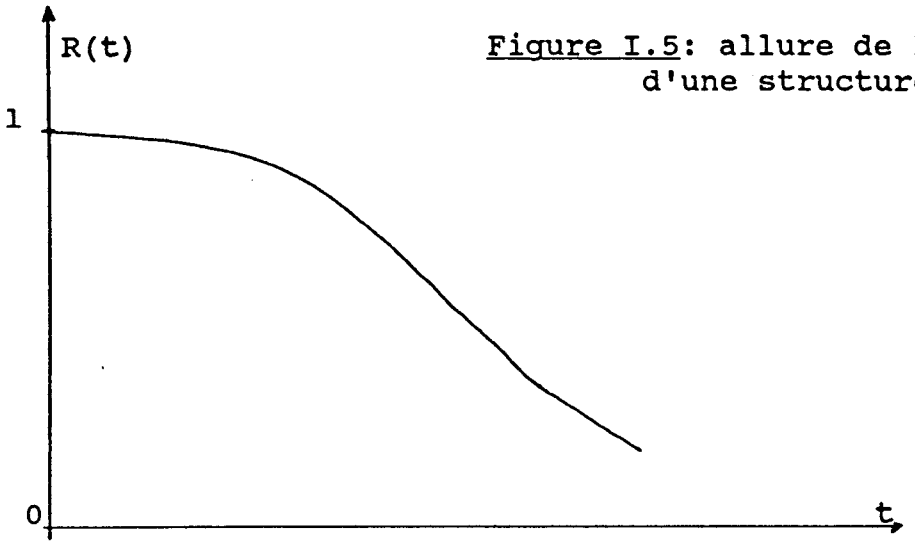


Figure I.5: allure de la fiabilité d'une structure.

b) La disponibilité (Availability) A(t) est la probabilité que le système fonctionne à l'instant t.

Dans le cas de systèmes non réparables, la définition de A(t) est équivalente à celle de la fiabilité.

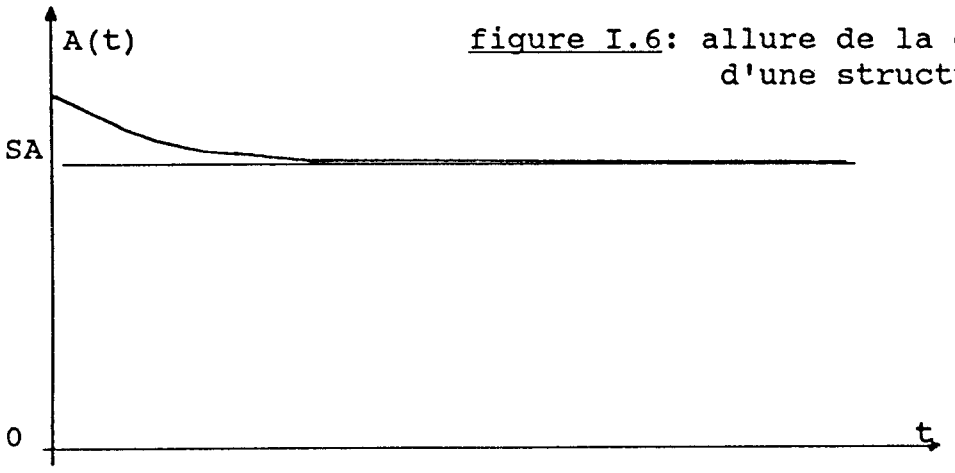


figure I.6: allure de la disponibilité d'une structure.

c) La maintenabilité: M(t) est la probabilité pour que le système fonctionne à l'instant t sachant qu'il était en panne à tout instant $\sigma \in [0,t]$.

Cette notion ne concerne que les systèmes réparables. M(t) est une fonction non décroissante variant de 0 à 1 dans l'intervalle $[0,\infty]$, figure I.7.

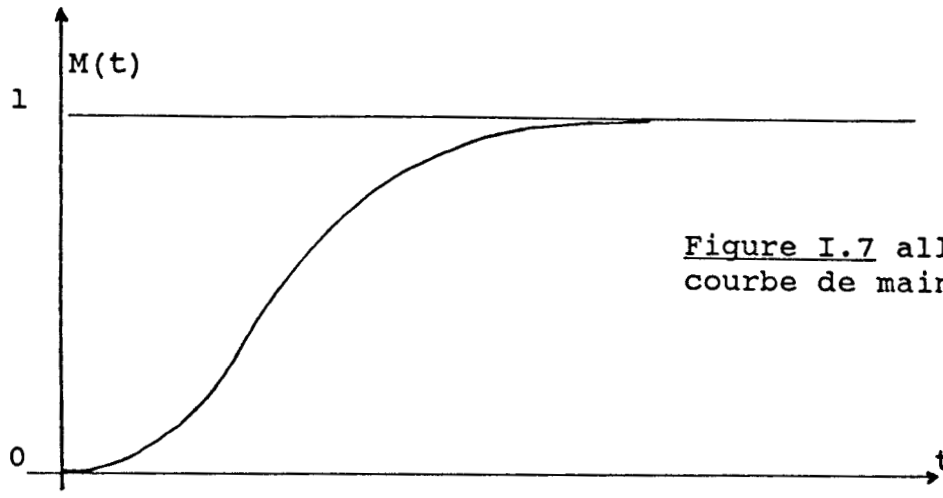


Figure I.7 allure de la courbe de maintenabilité

d) La sécurité: (Safety) S(t) est la probabilité de fonctionnement de sécurité à tout instant $\sigma \in [0,t]$.
Un fonctionnement de sécurité est tel que blessures, morts de personnes ou dégats matériels catastrophiques soient impossibles. Il est clair que la sécurité absolue n'existe pas.

D'autres notions sont parfois évoquées pour mieux caractériser un système, ce sont notamment la crédibilité et la survivabilité qui se définissent comme suit:

e) La crédibilité: (Credibility) C(t) est la probabilité pour que le temps moyen de latence d'une erreur soit inférieur à un temps t.
Les erreurs de conception n'y sont pas prises en compte, en général.

f) La survivabilité SU(t) est la probabilité de fonctionnement à l'instant t sachant qu'une défaillance s'est produite entre les instants 0 et t.

2.3. Les estimateurs de la sûreté de fonctionnement:

Toutes ces grandeurs sont donc des fonctions du temps t. leur calcul étant assez fastidieux, les industriels utilisent d'autres estimateurs déduits de ces fonctions par intégration ou moyennage. Les principaux sont les suivants:

a) La durée moyenne de bon fonctionnement avant la première défaillance (Mean Time to First Failure):

$$MTFF = \int_0^{\infty} R(t)\, dt$$

b) La disponibilité stationnaire (Stationnary Availability): SA est généralement proche de l'unité, on a couramment SA=0.99.

$$SA = \lim_{t \to \infty} A(t)$$

c) Le temps moyen jusqu'à la réparation (Mean Time To Repair):

$$MTTR = \int_0^{\infty} M(t)\, dt$$

d) Le temps moyen entre défaillances (Mean Time Between Failure) pour un système réparable:

$$MTBF = MTFF + MTTR$$

Ce facteur peut être calculé par deux autres estimateurs avec la formule: MTBF = MUT + MDT où :
- MUT (Mean Up Time) est le temps moyen de bon fonctionnement après réparation.
- MDT (Mean Down Time) est le temps moyen de défaillance .
MUT et MDT correspondent respectivement aux valeurs de MTFF et MTTR atteintes en régime stationnaire.

e) On rencontre aussi un autre estimateurs le MTFMF (Mean Time to First Maliguant Failure) ou temps moyen jusqu'à la première défaillance maligne lié à la sécurité par la relation:

$$MTFMF = \int_0^{\infty} S(t)\, dt$$

Ces différents paramètres, permettant d'estimer la sûreté de fonctionnement d'un système, n'ont pas la même importance suivant que celui-ci est ou non réparable. Alors que la fiabilité est primordiale pour un système non réparable, pour les systèmes réparables la disponibilité constitue un facteur plus significatif.

2.4.Système à tolérance ou intolérance de pannes:

Les différents estimateurs de la sûreté de fonctionnement ne sont pas indépendants. Le concepteur aura donc souvent à faire un compromis entre ces grandeurs pour respecter les critères de sûreté imposés par le cahier des charges.
Il dispose pour cela de deux moyens d'action:

1) Il peut diminuer la probabilité d'apparition de défaillances, on parle alors d'intolérance aux fautes;

2) Il peut limiter les effets des perturbations en augmentant la probabilité qu'une erreur soit acceptée ou tolérée. On parle dans ce cas de tolérance de pannes.

Suivant l'approche adoptée, tolérance ou intolérance aux fautes, les problèmes à résoudre, pour obtenir une meilleure sûreté de fonctionnement, sont différents.

a) Intolérance aux fautes:

Cette politique necessite une analyse très poussée en phase de conception pour une meilleure adéquation du système avec son environnement, ce qui implique en particulier:
- Un choix de composants très fiables;
- Une étude affinée des conditions de fonctionnement;
- Une validation à chaque niveau de la conception, et en particulier l'élimination des défauts de jeunesse.
- Une maintenance préventive efficace.

b) Tolérance aux fautes:

Cette approche est basée sur l'utilisation de redondances qui permettent au système de tolérer ou d'accepter certaines fautes ou erreurs. Ceci necessite:
- La détection et la localisation des erreurs;
- Un mécanisme de réaction permettant:
    - Le confinement des erreurs évitant ainsi leur propagation;
    - La correction de ces erreurs;
    - La reconfiguration du système afin qu'il puisse continuer à assurer sa mission malgré la panne.

Cette politique est souvent appliquée aux systèmes dont la mission est importante ou dont la réparation est impossible. Les méthodes de protection contre les fautes sont souvent relatives à certaines hypothèses de pannes. Les types de pannes le plus généralement prises en compte dans les automatismes sont les collages de capteurs à 0 ou 1.

L'introduction de redondances dans un système a l'inconvéniant d'augmenter le taux de défaillance. Les redondances ne constituent donc pas toujours la solution idéale. Dans la majorité des cas les deux stratégies de conception, tolérance ou intolérance de pannes, coexisteront pour assurer une sûreté de fonctionnement optimale.

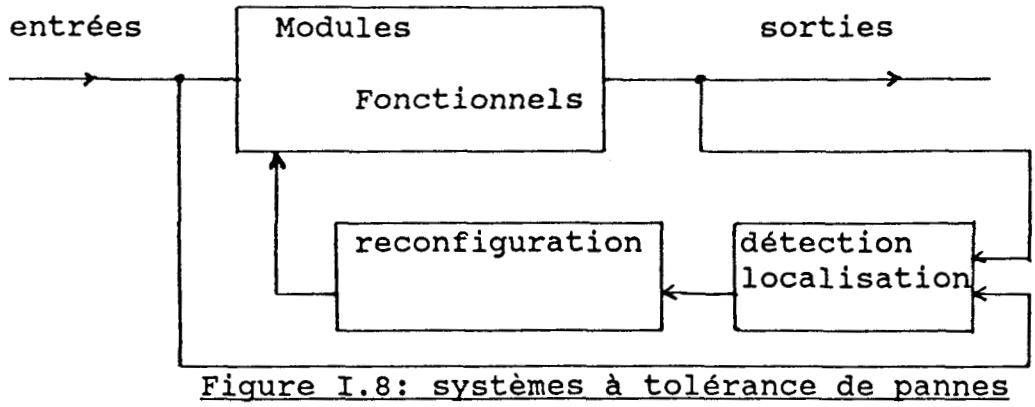


Figure I.8: systèmes à tolérance de pannes

2.5.Classification des redondances:

La redondance peut être envisagée au niveau d'un élément, d'un sous ensemble ou de toute la chaine. En fonction de l'état de l'élément redondant avant la détection d'une erreur et son comportement vis à vis des tâches on peut distinguer les types de redondances suivantes.

a) Dans une redondance statique l'élément redondant participe à la réalisation des tâches avant même l'apparition d'une erreur. Un cas particulier de redondance statique est constitué par un ensemble de modules effectuant le même traitement simultanément. Le résultat à prendre en compte est alors choisi par un vote à la majorité.

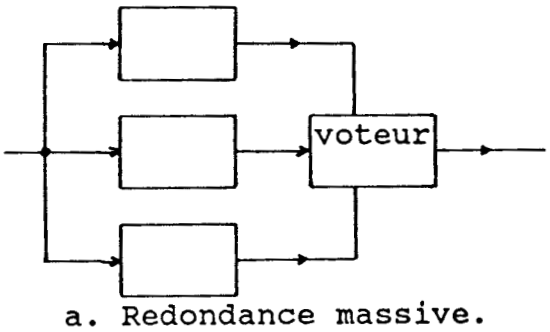
b) Si la redondance est dynamique, l'élément redondant ne participe à la réalisation des tâches qu'après détection d'une erreur.

c) Lorsque la redondance est temporelle, l'élément redondant se contente de mémoriser les informations avant erreur afin de les restituer après réaction pour la reconfiguration et la reprise. Un cas particulier de redondance temporelle est constituée par la réalisation de stocks tampons.

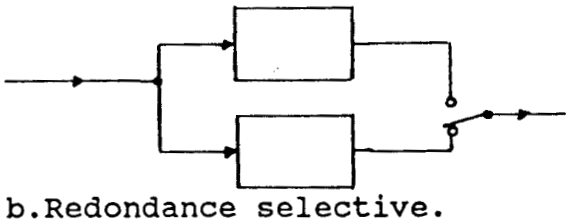
d) Dans une redondance sélective l'élément redondant remplace l'élément défaillant après détection d'une erreur.

e) La redondance est dite active si l'élément redondant est en service avant détection d'une erreur; elle est dite passive dans le cas contraire.

figure I.9: différentes redondances.



a. Redondance massive.



b.Redondance selective.

3.EVALUATION DE LA SURETE D'UN SYSTEME REPARABLE

L'évaluation des paramètres de sûreté de fonctionnement d'un système permet entre autre de mieux choisir la structure qui convienne en fonction des objectifs assignés. Ce calcul est en général assez complexe [46], [51], [52] et ne constitue pas l'objet de notre étude. Mais l'évaluation simplifiée qui sera faite ci-après va nous permettre de dégager des tendances et de justifier le cadre dans lequel s'inscrit la suite de notre étude.

3.1.Calcul des estimateurs de la sûreté:

On représente le système sous la forme d'une machine d'états stochastique. A chaque état E_i de la structure est associée une probabilité $P_i(t)$ d'être dans cet état. A chaque arc (E_i,E_j) est associé un taux de transition $P_{ij}(t)$ représentant le taux de défaillance et de réparation qui permettent de changer l'état de la structure.

Nous distinguons trois états principaux de la structure (figure I.10).

- E_1 : l'automatisme est sain.
- E_2 : l'automatisme est dans une position de sécurité suite à la détection d'une erreur.
- E_3 : l'automatisme contient une erreur non détectée. Il est considéré comme dangereux.

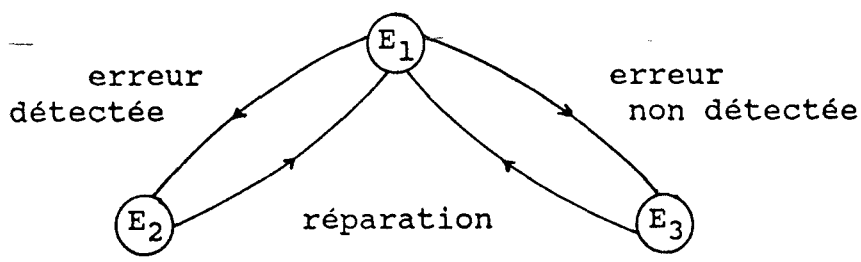


Figure I.10: Les états de la structure.

Le système est muni d'un dispositif de détection d'erreurs supposé infaillible pour simplifier dans une première étape. Il est caractérisé par son taux de couverture Pc ou probabilité que la panne soit signalée sachant qu'une panne existe.

Toute panne détectée est supposée mettre le système dans un état de sécurité. De plus nous supposons que le taux de pannes λ, le taux de réparation après une panne détectée μ et non détectée γ sont constants. Ce qui permet d'obtenir le graphe de la figure I.11.

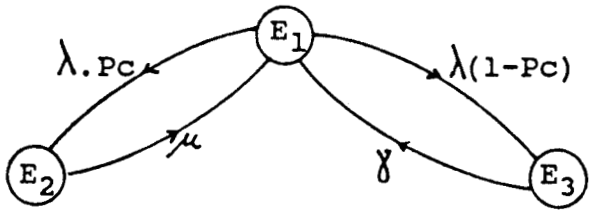


Figure I.11: taux de transition.

Si $P_{ij}(t,t+dt)$ représente la probabilité de transition de l'état E_i vers l'état E_j dans l'intervalle [t,t+dt], le théorème des probabilités totales permet de calculer la probabilité $P_j(t+dt)$ d'être dans l'état E_j à l'instant t+dt par la formule:

$$(I.1) \quad P_j(t+dt) = \sum_{i=1}^{n} P_{ij}(t,t+dt).P_i(t); \quad n = \text{nombre d'états}$$

Les taux de transition P_{ij} étant supposés constants, nous obtenons l'équation différentielle suivante lorsque dt tend vers zéro.

$$(I.2) \quad \frac{d(P_j)}{dt} = P'_j = \sum_{i=1}^{n} P_{ij}.P_i(t)$$

A partir du graphe de la figure I.11 nous obtenons le système d'équations suivant:

$$(I.3) \quad \begin{cases} P_1 = -\lambda P_1 + \mu P_2 + \gamma P_3 \\ P_2 = \lambda Pc P_1 - \mu P_2 \\ P_3 = \lambda(1-Pc) P_1 - \gamma P_3 \end{cases}$$

En utilisant la transformée de Laplace et en prenant comme conditions initiales que le système est sain $P_1(t_o)=1$, $P_2(t_o)=P_3(t_o)=0$, on obtient le graphe de fluence de la figure I.12 suivant:

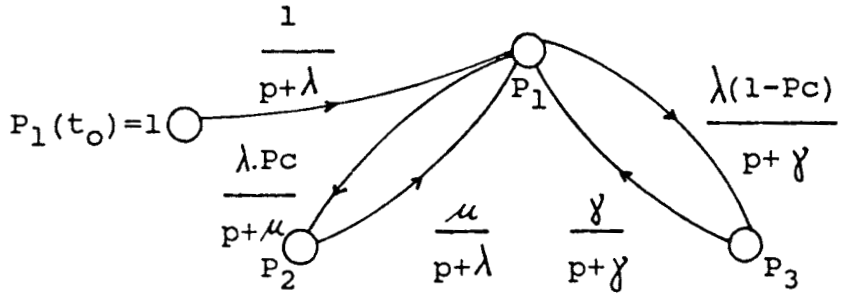


Figure I.12: (p désigne la variable de Laplace).

L'utilisation de la règle de Mason permet de tirer de ce système d'équations les relations suivantes.

a) Disponibilité:

$$(I.4) \qquad A(p) = \frac{1/(p+\lambda)}{1 - \frac{\mu Pc\,\lambda}{(p+\mu)(p+\lambda)} - \frac{\gamma\lambda(1-Pc)}{(p+\lambda)(p+\gamma)}}$$

ce qui donne la disponibilité asymptotique:

$$(I.5) \qquad SA = \frac{\mu\gamma}{\mu\gamma + \lambda Pc + \mu\lambda(1-Pc)}$$

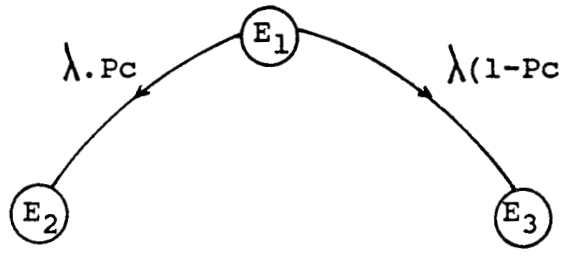
Si on suppose que $\mu=\gamma$ on obtient les relations simplifiées suivantes:

$$(I.6) \qquad A(t) = \frac{\mu}{\lambda+\mu}\left\{1 + \frac{\lambda}{\mu}\, e^{-(\lambda+\mu)t}\right\}$$

$$(I.7) \qquad SA = \lim_{t\to\infty} A(t) = \lim_{p\to 0} pA(p) = \frac{\mu}{\lambda+\mu}$$

b) Fiabilité: La mission n'étant plus remplie pendant les temps de réparation le graphe se réduit à:

Figure I.13: graphe de fiabilité.



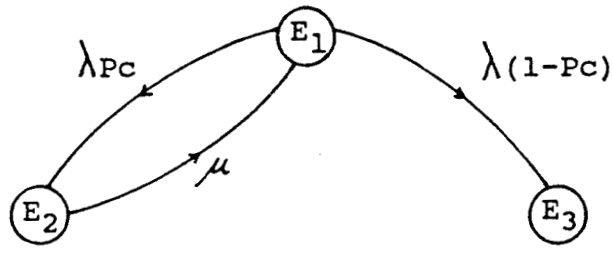
R(t) = Probabilité d'être dans l'état E_1 à tout instant $\sigma \in [0,t]$. Ce qui permet d'écrire:

$$(I.8) \qquad R(t) = e^{-\lambda t}$$

$$(I.9) \qquad MTFF = 1/\lambda$$

c) Sécurité: l'état E_3 est le seul inacceptable, en le rendant absorbant on obtient le graphe suivant:

Figure I.14: graphe de sécurité.



$S(t) = 1 - P_3(t)$ soit la probabilité de ne pas être dans l'état E_3 à tout instant $\sigma \in [0,t]$. Le calcul donne finalement la relation suivante pour l'expression de la sécurité:

$$(I.10) \qquad S(t) = \frac{k1 + \mu + \lambda Pc}{k1 - k2} e^{k1t} + \frac{k2 + \mu + \lambda Pc}{k2 - k1} e^{k2t}$$

où k1 et k2 sont donnés par:

$$k1,k2 = \frac{1}{2} \left(-(\mu+\lambda) \pm \sqrt{(\mu-\lambda)^2 + 4\mu\lambda Pc} \right)$$

La valeur de l'estimateur MTFMF est alors donnée par la ralation I.11 suivante:

$$(I.11) \qquad MTFMF = \frac{\mu + \lambda . Pc}{(1 - Pc) . \mu . \lambda}$$

3.2. Influence des différents facteurs sur la sûreté:

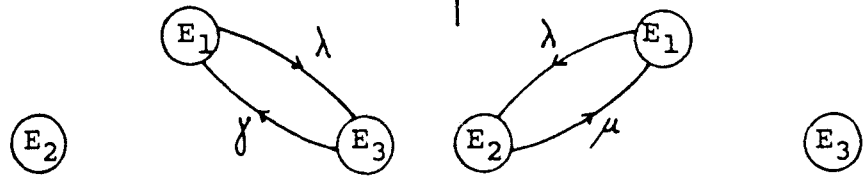
a) Influence du mécanisme de détection

L'évaluation qui a été faite n'est qu'approximative dans la mesure où le mécanisme de détection a été supposé avoir une probabilité de panne nulle. Néamoins, ce calcul va nous permettre de dégager des tendances et de justifier le cadre dans lequel s'inscrit la suite de notre étude.

Pour cela nous considérons les cas extrêmes avec Pc=0 (suppression du mécanisme de détection) et Pc=1 (toutes les pannes sont détectées). Les résultats sont consignés dans le tableau suivant:

	Pc=0	Pc=1
Fiabilité	$R(t)=e^{-\lambda t}$; $MTFF=1/\lambda$	$R(t)=e^{-\lambda t}$; $MTFF=1/\lambda$
Sécurité	$S(t)=e^{-\lambda t}$; $MTFMF=1/\lambda$	$S(t)=1$; $MTFMF \to \infty$
Disponibilité	$A(t)=\frac{\gamma}{\lambda+\gamma}\left(1-\frac{\lambda}{\gamma}e^{-(\lambda+\gamma)t}\right)$	$A(t)=\frac{\mu}{\lambda+\mu}\left(1+\frac{\lambda}{\mu}e^{-(\lambda+\mu)t}\right)$

Figure I.15



Malgré les résultats idéalisés en ce qui concerne la fiabilité et la sécurité, nous pouvons admettre néamoins une amélioration de la sécurité par l'utilisation d'un dispositif de détection d'erreurs.

Si celui-ci est capable d'aider à localiser les défauts $\mu < \gamma$ la disponibilité peut être améliorée. Celle-ci n'est pas modifiée si $\mu = \gamma$. Elle est, en fait, beaucoup plus sensible à l'efficacité de la maintenance.

b) Influence des taux de réparation:

Le temps de réparation dépend aussi d'autres conditions (disponibilité et efficacité de l'équipe de maintenance, gestion des stocks de pièces de rechange...). Dans la pratique, les temps de réparation sont beaucoup plus faibles que les temps entre défaillances. Un rapport $\beta = \lambda/\mu$ de 10^{-2} est facilement atteint même par des automatismes assez complexes. La perte de production en régime stationnaire est alors de 1% . Elle passe à 1 ‰ pour un rapport $\lambda/\mu = 10^{-3}$. Dans ces conditions, en tant que critère de choix d'une structure, la disponibilité passe au second plan.

En ce qui concerne l'influence des taux de réparation sur la sécurité, la relation (I.10) permet de tirer l'expression suivante lorsque $\beta = \lambda/\mu$ tend vers zéro.

$$S(t) = e^{-(1-Pc)\lambda t} \qquad (I.12)$$

La figure I.16 montre l'influence faible qu'a le rapport λ/μ sur la sécurité.

Figure I.16 évolution de S(t) en fonction de λ/μ pour des valeurs telles que $\lambda t=1$, $\lambda=0.0001$.

λ/μ	Pc=0.5	Pc=0.9
10^{-2}	0.60803	0.90564
10^{-3}	0.60668	0.90492
10^{-4}	0.60655	0.90485
10^{-5}	0.60653	0.90484

c) Influence du temps de latence:

Entre le moment où apparaît une panne détectable et le moment où l'automatisme passe en fonctionnement de sécurité, il s'écoule un certain temps appelé temps de latence t_L. La probabilité d'apparition d'autres pannes pendant ce temps n'est pas nulle. La structure ne peut plus être représentée par un graphe markovien puisque le taux de transition de l'état défaillant E_2 vers l'état conduisant à la réparation est fonction du temps.

Nous ferons donc un calcul de probabilité directe [51]. Le temps de latence t_L est supposé constant. Soit

P_{21} = Probabilité qu'une deuxième panne se produise pendant le temps de latence d'une panne détectable.

Si nous supposons que ces pannes ne sont pas corrélées, nous pouvons écrire.

P_{21} = Pr(une défaillance détectable à l'instant t_1) * Pr(une défaillance dans l'intervalle t_1, $t_1 + t_L$)

A partir de la structure complètement réparée nous avons:

$$(I.13) \qquad Pr(\text{une panne détectable à } t_1) = Pc(1 - e^{-\lambda t_L})$$

Si nous supposons que la première panne ne modifie ni le taux de couverture ni le taux de pannes alors:

$Pr(\text{une défaillance dans l'intervalle } t_1,t) = 1 - e^{-\lambda(t-t_1)}$

avec $t_1 \leq t \leq (t_1+t_L)$

A l'instant $t=t_1+t_L$ les relations précédentes donnent:

$$(I.14) \qquad P_{21} = Pc(1-e^{-\lambda t_1})(1-e^{-\lambda t_T})$$

Le développement limité de (I.12) au voisinage de $\lambda t_L=0$ pour t quelconque conduit à:

$$(I.15) \qquad P_{21} \# Pc.\lambda.t_L(1-e^{-\lambda t})$$
$$P_{21} \# \lambda t_L.P_2$$

Soient les probabilités:

P_{22}=Pr(1 panne détectable après une panne détectable latente)

P_{23}=Pr(1 panne non détectable après une panne détectable latente)

On trouve par un calcul analogue:

$$(I.16) \qquad P_{22}= \lambda t_L Pc.P_2$$
$$P_{23}= \lambda t_L(1-Pc)P_2$$

3.3.Calcul des paramètres de la sûreté dans le cas où le taux de panne du système de détection n'est pas nul

Dans le calcul précédent nous avons supposé que le mécanisme de détection est idéal du point de vue de la fiabilité (taux de panne nul). Cette hypothèse n'est pas réaliste. Le calcul qui suit va nous permettre de mieux analyser l'influence de la présence d'un dispositif de détection d'erreurs [51].

Nous supposons que ce dispositif, appelé ensemble secondaire dans la suite, est distinct du module fonctionnel ou primaire.
Nous ferons de plus les hypothèses suivantes:

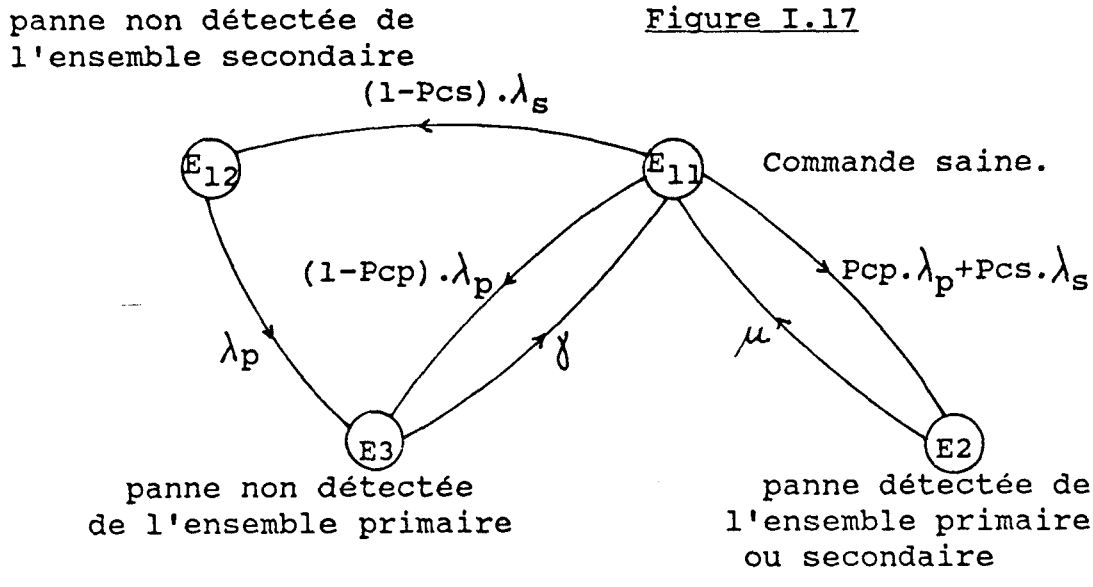
- Toute panne de l'ensemble secondaire le rend inefficace.

- la détection d'une panne provoque une interruption de service même si c'est l'ensemble secondaire qui est défaillant.

- On négligera le temps de latence des pannes détectables ainsi que les temps de réparation.

Dans ces conditions le graphe des états de la structure se présente comme suit (figure I.17).

Figure I.17



Dans les états E_2, E_3 défaillants la mission n'est plus assurée. Pcp et Pcs sont respectivement le taux de couverture de l'ensemble primaire et secondaire. Les termes λ_p et λ_s représentent le taux de pannes de l'ensemble primaire et de l'ensemble secondaire.

a) Fiabilité

Le graphe de fiabilité de cette structure est celui représenté à la figure I.18.

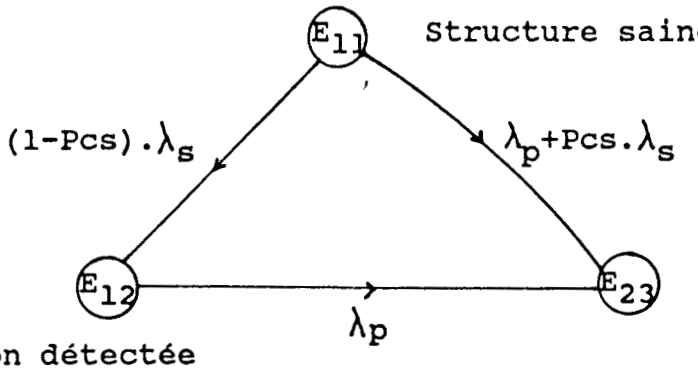


Figure I.18: graphe de fiabilité

On obtient pour l'expression de la fiabilité les relations suivantes:

$$(I.17) \qquad R(t) = e^{-\lambda_p . t} \{ 1 - Pcs(1 - e^{-\lambda_s . t}) \}$$

$$(I.18) \qquad MTFF = \frac{\lambda_p + (1 - Pcs).\lambda_s}{\lambda_p(\lambda_p + \lambda_s)}$$

Dans le cas où le taux de couverture de l'ensemble secondaire Pcs est proche de 1 on trouve les relations simplifiées suivantes:

$$(I.19) \qquad R(t) = e^{-(\lambda_p+\lambda_s)t} \qquad (I.20) \qquad MTFF = \frac{1}{\lambda_p + \lambda_s}$$

b) Sécurité:

Dans le cas de la sécurité on considère que seul l'état E_3 où il y a panne non détectée de l'ensemble primaire est dangereux. Ce qui donne le graphe de sécurité de la figure I.19.

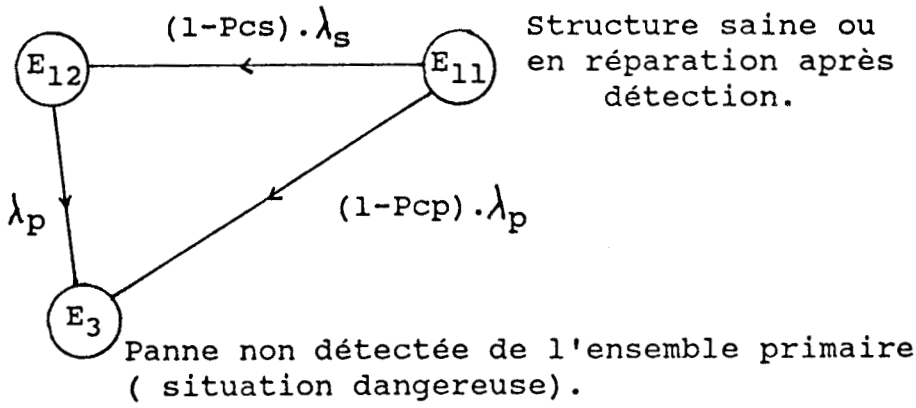


Figure I.19: graphe de sécurité.

Le calcul donne les relations suivantes pour la sécurité.

$$(I.21) \quad S(t) = \frac{1}{1-k} \{ e^{-(kp+ks)t} - k.e^{-\lambda p.t} \}$$

$$(I.22) \quad MTFMF = \frac{Pcp.\lambda_p}{(\lambda_p - ks.kp)ks.kp} - \frac{1 - Pcs}{\lambda_p - ks.kp} . \frac{\lambda_s}{\lambda_p}$$

avec

$$kp = (1 - Pcp)\lambda_p \; ; \qquad ks = (1 - Pcs)\lambda_s \; ; \qquad k = \frac{ks}{kp - \lambda_p}$$

Lorsque $Pcp.\lambda_p <> 0$ on peut écrire:

$$(I.23) \quad \lim_{ks \to 0} S(t) = e^{-kp.t}$$

Si Pcs est très peut différent de 1 on obtient la relation suivante pour la sécurité:

$$(I.24) \quad S(t) = e^{-kp.t}$$

$$(I.25) \quad MTFMF = \frac{1}{kp}$$

3.4.Remarques sur le choix d'une architecture:

Le choix de la structure d'un automatisme réparable résulte d'un compromis coût/performance. L'introduction de redondances destinées à améliorer la sécurité entraine une augmentation de la fréquence des pannes . Un compromis sécurité/fiabilité est donc à rechercher.

Les paramètres de la sûreté de fonctionnement sont relatifs à l'aptitude d'un système à remplir sa mission, et non aux caractéristiques de sa structure. Ceci est particulièrement vrai pour la fiabilité.

R(t) = Probabilité de fonctionnement à tout instant $\sigma \in [0,t]$

est en général interprétée comme la probabilité de remplir la mission. Ce qui tend à privilégier le choix de structures redondantes complexes au profit d'un dispositif de détection-réaction.

Pour mieux caractériser une structure on défini un estimateur appelé fiabilité intrinsèque (inherent relaibility) de la façon suivante:

IR(t) = Probabilité que la structure ne nécessite pas de réparation à tout instant $\sigma \in [0,t]$.

Celui-ci diffère donc de la fiabilité puisque toute panne même acceptée ou tolérée va nécessiter une réparation. Allonger le temps de réparation d'une faute acceptée ou tolérée conduit à un risque de pannes multiples.

Ce paramètre permet d'estimer le temps moyen entre interventions du service de maintenance et donne donc une estimation des coûts d'entretien.

Exemple

La structure à redondance massive d'ordre 3 de la figure I.9.a à vote majoritaire a une fiabilité qui tend vers 1 (MTFF ->∞) avec un voteur idéal (taux de pannes nul). Par contre:

$IR(t) = e^{-3\lambda t}$ si λ est le taux de panne d'un seul module.

Nous constatons que le rapport λ/μ est tel que la disponibilité d'un automatisme simple est souvent suffisante. Par contre la sécurité obtenu avec cet automatisme non tolérant peut être jugée insuffisante. Il faut alors agir pour améliorer ce paramètre. Le choix d'une architecture dépend au fait de la neccessité ou non d'assurer la continuité partielle ou totale de la mission pour assurer la sécurité. Si cette continuité est requise l'introduction de redondances est nécessaire.

L'introduction d'un dispositif de détection d'erreurs s'il permet d'améliorer notablement la sécurité peut avoir une influence néfaste sur la fiabilité du système.

Pour s'approcher au mieux de l'optimum nous avons la possibilité d'améliorer l'efficacité des autotests (Pcs -> 1) et/ou la fiabilité des organes secondaires (λs -> 0).

On peut remarquer (relation I.21) que lorsque le rapport

$$(I.26) \qquad Sc = \frac{(1 - Pcs)\lambda s}{Pcp.\lambda p}$$

tend vers 1 nous obtenons une sécurité $S(t) = e^{-\lambda p.t}$ qui est identique à celle de la structure sans dispositif de détection.

Pour obtenir un gain en sécurité il faut donc que ce rapport Sc soit inférieur à 1. Le taux de couverture des pannes de l'ensemble primaire Pcp reste sans influence sur la fiabilité (relation I.17).

Le dilemme sécurité/fiabilité est illustré ici par l'influence du taux de couverture Pcs. L'augmentation de Pcs améliore la sécurité (I.24) au détriment de la fiabilité (I.19) Lorsque Pcs # 1, les relations I.24 et I.25 montrent que la sécurité obtenue dans ce cas est effectivement celle d'une structure avec un système de détection totalement intolérant.

La réduction de λs ou l'augmentation de la fiabilité du dispositif secondaire de détection est bénéfique sur le plan de la sécurité et de la fiabilité de l'ensemble.

4. PRINCIPALES CAUSES DE DEFAILLANCE DANS UN AUTOMATISME

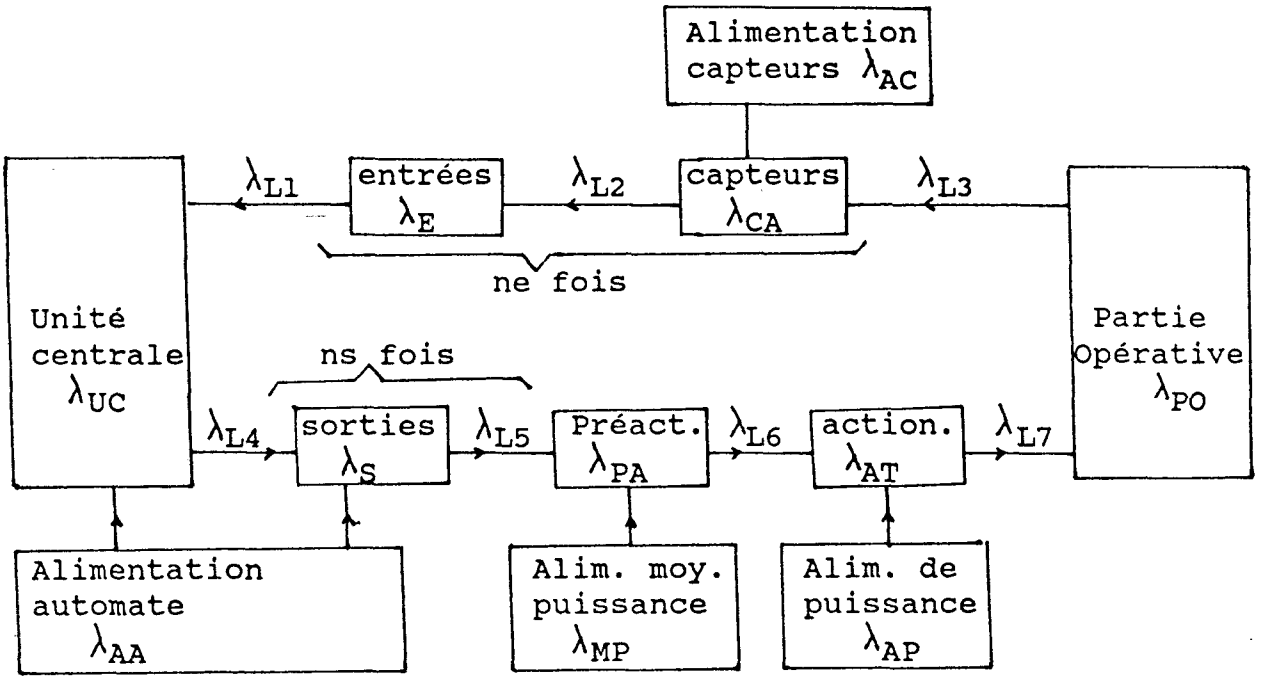
L'opinion actuellement développée par certains constructeurs d'automates programmables est que 95% environ des pannes survenant dans un automatisme ont des causes extérieures à l'automate. Ces pannes proviendraient donc de la PO, préactionneurs, actionneurs, capteurs. De plus, dans ce pourcentage une grande partie des fautes sont imputables aux interfaces d'entrée/sortie.

Dans ces conditions, les mécanismes de test utilisés le plus souvent pour surveiller le fonctionnement de l'automate ont une couverture de détection très restreinte vis à vis de l'ensemble de l'automatisme. Il serait alors plus judicieux de surveiller les parties les moins fiables pour améliorer le facteur de couverture Pc et par conséquent la sécurité.

Sans aller jusqu'à faire un calcul exaustif de fiabilité, nous allons, par des considérations simples, essayer de dégager les principales tendances dans ce domaine.

La figure I.20 montre les principaux modules constituant un automatisme qui doivent être pris en compte dans un calcul de fiabilité ainsi que leurs taux de panne respectifs.

Figure I.20 répartition des taux de panne dans un automatisme.



On suppose:

- que tous ces éléments sont en série pour la fiabilité: la défaillance de l'un des composants entraine la défaillance de l'ensemble;

- et que les défaillances affectant les différents éléments sont non corrélées et exponentiellement distribuées.

Dans ces conditions le taux de pannes de l'ensemble de l'automatisme est exprimé par la somme:

$$\lambda = \lambda_{UC} + \lambda_{L1} + \lambda_{L2} + ne(\lambda_E + \lambda_{L2} + \lambda_{CA} + \lambda_{L3}) + ns(\lambda_S + \lambda_{L5}) + npa.\lambda_{PA} + nac(\lambda_{L6} + \lambda_{AT} + \lambda_{L7}) + \lambda_{PO} + \lambda_{AA} + \lambda_{AC} + \lambda_{MP} + \lambda_{AP}$$

Dans cette expression:
- ne est le nombre d'entrées,
- ns est le nombre de sorties,
- npa est le nombre de préactionneurs,
- nac est le nombre d'actionneurs

Un exemple de valeurs théoriques de taux de pannes est donné dans le tableau de la figure I.21, concernant les A.P.I de la série PB400 de Merlin Gérin.

Figure I.21: Taux de pannes des composants d'un API.

carte	$\lambda.10^{-6}$ panne/h	$\lambda_{nc}.10^{-6}$ panne/h	Pc %
16 E 24 Vc	37.8	26.5	29.7
16 E 110 ou 220 V	41.3	29.1	29.5
16 E 24/48 v AC/DC	41.9	30.0	24.4
16 S Relais	54.0	26.6	50.7
16 S 24/48 v 0.5 A	45.9	26.4	42.5
8 S 2 A	37.8	23.4	38.2
8 S 110/220 v	39.9	25.4	36.3
12 E analogiques	78.2	14.85	81.0
UC 4k RAM	124.7	64.5	48.0
UC 4k EEPROM	120.7	36.4	69.8
carte surveillance	30.7	1.5	95.0
Alimentation	80.0	1.0	98.8

λ_{nc} représente le taux de pannes non couvertes par des mécanismes internes de détection implantés dans cet automate.

Pour une configuration moyenne comprenant 60 entrées et 48 sorties, on trouve un taux de pannes de l'automate et de ses interfaces de l'ordre de:

$\lambda_{AUT} = 377.10^{-6}$ (E/S) + $268.10^{-6} = 645.10^{-6}$ panne/h

Les dispositifs d'entrée/sortie constituent ainsi l'origine d'environ 25% des pannes de l'automate. Ce pourcentage passe à 76 pour une configuration à 160 entrées/sorties.

Les éléments de la partie opérative ou hors automate ont des taux de panne fonction du nombre de manoeuvres f. Un ordre de grandeur est donné ci-après:

- capteurs type fin de course: $0.3\ 10^{-6} * f$
- distributeurs pneumatiques : $0.4\ 10^{-6} * f$
- vérin : $0.5\ 10^{-6} * f * h$

Le facteur h est la course du vérin.

En supposant que l'automate précédent est utilisé avec une PO comprenant:

- un capteur pour chaque entrée,
- un vérin associé à un distributeur pour deux sorties (hypothèse de préactionneurs double pilotage) avec une course moyenne pour les vérins de 0.2 m.

on trouve un taux de panne pour les préactionneurs, actionneurs, capteurs de l'ordre de:

$$\lambda_{PO} = (60*0.3\ 10^{-6} + 24*0.4\ 10^{-6}).\ f$$

Ce qui donne $276 \cdot 10^{-6}$ panne/h pour une cadence de f=10 sollicitations/heure. Ce chiffre passe à 960.10^{-6} pour une configuration à 160 entrées/sorties avec les mêmes suppositions. Le tableau figure I.22 donne les pourcentages de pannes des différents sous-ensembles.

Figure I.22: contribution des différents sous ensembles au taux de panne d'un automatisme.

	Automate		Préactionneurs, actionneurs, capteurs
	Unité centrale	Interfaces d'E/S	
64E/48S	29%	41%	30%
160E/160S	12%	44%	44%

On constate avec ces tendances que pour des automatismes ayant peu d'entrées/sorties la fiabilité de l'unité de traitement et de ses interfaces est prépondérante. Lorsque le nombre d'entrées/sorties est important avec beaucoup de capteurs et d'actionneurs la fiabilité de l'ensemble sera beaucoup plus tributaire des éléments hors automate et sera fonction de la cadence (f) de la partie opérative.

5. DETECTION ET LOCALISATION DES ERREURS

La détection et la localisation des erreurs constituent une phase très importante pour les systèmes à tolérance de pannes dont c'est souvent le cas pour les systèmes de production automatisés. Cette détection permet en particulier de réaliser la commutation nécessaire pour reconfigurer le système.

Le problème de la conception de systèmes autotestables a donné lieu à une abondante littérature [43], [45]. Nous empreintons à Geffroy [45] les définitions qui suivent.

5.1. Domaines d'évolution d'un système séquentiel:

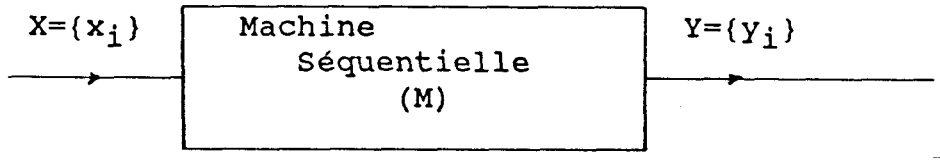
Soit un sytème M à évolution séquentielle où:

X = {x1,x2,...xn} constitue l'alphabet d'entrée ou l'ensemble des configurations possibles binaires prises par les entrées d'un système logique.

Y = {y1,y2,...,ym} est l'alphabet de sortie.

X_N^* et Y_N^* désignent respectivement l'ensemble des séquences d'entrée effectivement appliquées au système et l'ensemble des séquences de sortie en fonctionnement normal.

Figure I.23:domaines d'évolution d'un système



Pour un ensemble de n variables essentielles ou alphabet $Z=\{z_i\}$ d'un système, on définie les domaines d'évolution suivants:

a) Le domaine statique d'évolution, noté D_{SZ} est constitué par les éléments de l'alphabet Z. Dans le cas de variables logiques D_{SZ} représente l'ensemble des configurations binaires ayant une probabilité non nulle d'apparition au cours du fonctionnement normal. Ce domaine est alors défini sur $\{0,1\}^n$ pour n variables binaires.

b) Le domaine dynamique d'évolution, noté D_{DZ} est l'ensemble des séquences de longueur finie formées par les éléments de l'alphabet et observables en fonctionnement normal. Ce domaine est défini sur l'ensemble U de toutes les séquences de longueur finie que l'on peut former. On peut également faire intervenir explicitement le temps pour mieux caractériser cet évolution dynamique du système.

Exemple: Soit un chariot dont la position est repérée à l'aide de trois capteurs a, b, c de type tout ou rien:

$D_{SZ} = \{a\bar{b}\bar{c}, \bar{a}\bar{b}\bar{c}, \bar{a}bc, \bar{a}\bar{b}c\} = \{r1, r2, r3, r4\}$

$U_{SZ} = \{\bar{a}\bar{b}\bar{c}, a\bar{b}\bar{c}, \bar{a}b\bar{c}, \bar{a}\bar{b}c, \bar{a}bc, a\bar{b}c, ab\bar{c}, abc\}$

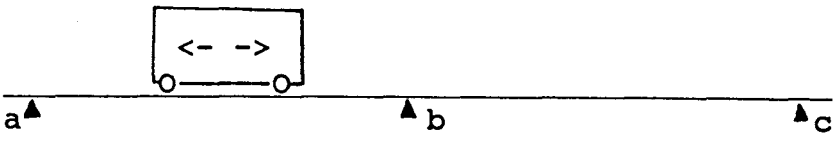


Figure I.24

$D_{DZ} = \{ (r1,r2,r2,r2,r3,r3,r3),(...) ...\}$; $ri \in D_{SZ}$

U_{DZ} = ensemble des séquences $\underline{s} = (U_{SZ})^*$

Il serait possible de faire intervenir explicitement la probabilité d'apparition de chaque combinaison ou élément de l'alphabet ou séquence en associant une fonction de probabilité au domaine d'évolution.

Un domaine d'évolution est redondant s'il est plus petit que son domaine de définition. Ceci est souvent le cas en pratique:

$U_Z \cap D_Z \neq \emptyset$

5.2.Systèmes autotestables:

La détection des erreurs nécessite une observabilité accrue des modules fonctionnels par l'organe de détection. Différentes techniques et redondances peuvent être mises en oeuvre.

a) Test hors ligne:

Il s'agit d'appliquer des séquences de test au système soustrait de son environnement après arrêt total de son fonctionnement. cette technique est généralement appliquée aux systèmes à conséquences de pannes non graves.

b) Test en ligne:

Le test en ligne est appliqué lorsque les conséquences de pannes peuvent devenir graves. Dans ce cas l'observabilité des pannes doit être prévue lors de la conception. Ce test peut être réalisé d'une façon continue ou discontinue:

- Dans le test en ligne discontinu les séquences de test sont appliquées aux modules fonctionnels depuis l'organe de détection. La surveillance est donc intégrée au fonctionnement normal.
- Dans le cas du test en ligne continu l'organe de détection ne fait qu'observer un ensemble de variables provenant des sous systèmes fonctionnels (observation permanente).

Si la détection des erreurs s'effectue par observation du domaine statique D_{DZ} l'observateur est sans mémoire ou d'ordre zéro. C'est le cas le plus simple et le plus fréquemment rencontré en pratique.

Si la détection est basée sur l'observation du domaine dynamique D_{DZ} l'observateur possède alors une mémoire. Il est d'ordre n pour des séquences observées de longueur maximale n.

c) Test de vraissemblance:

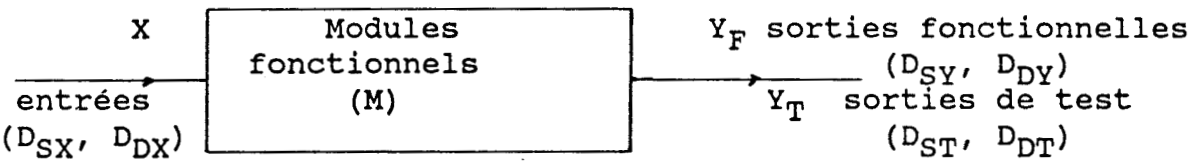
Il s'agit de la vérification de l'appartenance à une fourchette de valeurs d'une grandeur, de sa dérivée ou de son accélération.

d) Autotest:

La détection est ici permanente (pannes transitoires ou non consistantes...).

Une détection permanente nécessite le choix de structures dites autotestables utilisant par exemple un code détecteur d'erreurs.

Figure I.25 systèmes autotestables



Les sorties fonctionnelles et de test ne sont pas toujours séparées. Suivant le code détecteur d'erreurs et la structure retenue on peut avoir les cas suivants:

- Y_T et Y_F définies sur les mêmes variables, (code k parmi n)
- Y_T et Y_F définies sur des variables différentes (parité...)
- les variables définissant Y_T sont partiellement ou totalement incluses dans celles définissant Y_F.

Définition I.1: Un système est **autotestable** pour un ensemble de pannes F si et seulement si:

$$\forall f \in F \quad \text{il existe } \underline{x}_i \in D_{DX} \text{ tel que } M(\underline{x}_i, f) = Y_T \notin D_{DT}$$

Y_T est la séquence résultant de l'application de $\underline{x}_i$ au système ayant la panne f. Cette séquence ne doit donc pas appartenir au code détecteur d'erreurs.

Pour une machine séquentielle $M = (X, Q, Y, \delta, \lambda)$ cette condition est exprimée de la façon suivante:

$$\forall f \in F, \text{ il existe } \underline{x}_i \in D_{DX} \text{ et } q \in Q \text{ tel que } \lambda^f(\delta^f(\underline{x}_i, q)) \notin D_{DT}$$

δ^f et λ^f représentent respectivement la fonction état suivant et la fonction de sortie lorsque la faute survient.

Une condition nécessaire de réalisation d'un système autotestable est que D_{DZ} et D_{DE} soient disjoints dans l'ensemble de définition U. D_{DE} est défini comme étant l'ensemble des séquences de fonctionnement prises par Y_T pour toutes les pannes de F.

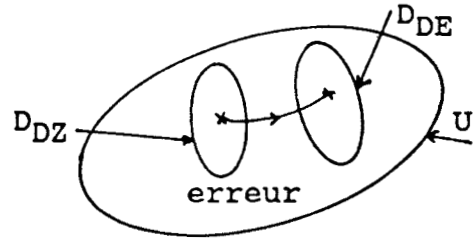


Figure I.26

Définition I.2 Un système est sûr en présence de panne pour un ensemble donné F de pannes si et seulement si:

$\forall f \in F, \forall \underline{x}_i \in D_{DX}$ on a:

soit $M(\underline{x}_i, f) = Y_T \in M(\underline{x}_i, \emptyset)$

soit $M(\underline{x}_i, f) = Y_T \neq M(\underline{x}_i, \emptyset)$ et $Y_T \notin D_{DT}$

$M(\underline{x}_i, \emptyset)$ correspond à un fonctionnement sans panne.
Pour une machine séquentielle $M=(X,Q,Y,\delta,\lambda)$ cette condition est exprimée de la façon suivante:

$\forall f \in F, \forall \underline{x}_i \in D_{DX}, \forall q \in Q,$

soit $\lambda^f(\delta^f(\underline{x}_i, q)) = \lambda(\delta(\underline{x}_i, q)$

soit $\lambda^f(\delta^f(\underline{x}_i, q)) \notin D_{DT}$

Définition I.3: Un système est totalement autotestable pour un ensemble de pannes F s'il est autotestable et sûr en présence d'une panne de F.

Cette définition assure que lorsque la panne agit elle est détectée, et lorsqu'elle n'agit pas le fonctionnement est correct.

Définition I.4: Un système est à auto-surveillance pour un ensemble de pannes F si et seulement si:
- il est autotestable pour F;
- il est à détection immédiate d'erreur lors de la première activation sur les sorties Y_T d'une panne de F.

Cette dernière définition est moins restrictive que la précédente car après la première activation, le système peut

évoluer de façon arbitraire et notamment la configuration suivante peut survenir:

$M(\underline{x}_i, f) = Y_T \in D_{DT}$ (retour au code)

$M(\underline{x}_i, f) = Y_F <> M(\underline{x}_i, \emptyset)$ (fonctionnement erronné)

Un système totalement autotestable satisfait donc à la définition de l'auto-surveillance avec la contrainte supplémentaire de ne jamais avoir de comportement erronné non détectable en sortie Y_T.

5.3.Principales structures totalement autotestables:

Ces structures peuvent être réparties en deux classes suivant que la redondance est ou non séparable.

a) Structures à redondance séparable:

Figure I.27: Observation des entrées-sorties

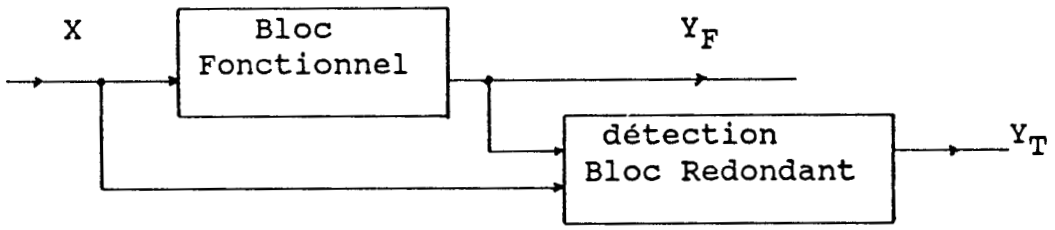


Figure I.28: Structure à duplication

Toute panne n'affectant pas de la même manière les deux blocs est détectée. Le coût en matériel est important.

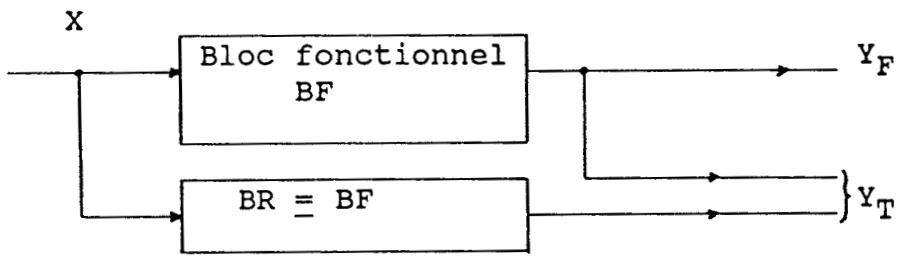
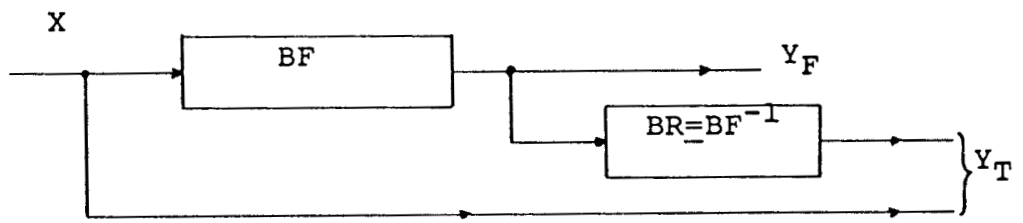


Figure I.29: Structure avec inverse:



Un système ne possède pas toujours d'inverse. La conception est souvent compliquée. La vitesse de fonctionnement est ralentie.

b) Structures à redondance non séparable:

Dans ces structures la redondance est totalement intégrée au bloc fonctionnel et apparaît comme un codage non séparable des informations caractéristiques du système (entrée, variables internes, sorties). Le code le plus utilisé est celui à pondération constante k parmi n. La distance de Hamming entre deux mots du code est égale à 2 afin d'éviter le masquage des erreurs simples.

Ces structures necessitent plus de contraintes à la conception des circuits fonctionnels et peuvent impliquer une adaptation du monde extérieur. La réduction en matériel est contreballancée par une couverture de panne souvent plus restreinte.

Un cas particulier de structure à redondance non séparable est celui d'une réalisation "double rails". Les méthodes basées sur la notion de poids associé à un réseau de Pétri font également partie de cette catégorie [37].

5.4.Description de quelques procédures mises en oeuvre pour la détection en ligne des erreurs dans les automatismes:

On peut classer ces procédures de diagnostic en deux groupes suivant qu'on surveille la partie commande ou la partie opérative. Au niveau de la PC, en tant que calculateur industriel, les méthodes de détection propres aux systèmes informatiques sont appliquées [36]. Parmi celles-ci on peut citer:

- le circuit "chien de garde" qui vérifie le fonctionnement du processeur ou le déroulement du programme,
- l'utilisation de codes détecteurs d'erreurs (parité...) ou sommes de contrôle pour les données et programmes,
- redondance temporelle dans les transmissions,
- contrôle de l'évolution de la PC en particulier à l'aide des réseaux de Pétri pondérés,
- contrôle de cohérence des informations.

5.4.1 Utilisation des réseaux de Pétri Pondérés: [57]

Un réseau de pétri pondéré est obtenu en attribuant à chaque place P_i un nombre entier strictement positif p_i appelé poids de la place. Ce nombre est choisi de façon à ce que pour toute transition t_j du réseau la somme des poids des places

antécédentes soit égale à celle des places subséquentes. Le poids total marqué d'un réseau représenté par la quantité:

$$\sum_i m_i p_i$$

où m_i est le nombre de marqueurs, est donc un invariant. Toute évolution anormale du graphe qui se manifeste par une diminution ou augmentation du poids total marqué est détectable. Les réseaux de Pétri pondérés permettent ainsi de détecter des modifications du marquage dues à l'action directe de parasites sur les mémoires représentant les places.

Parmi les méthodes de surveillance de la partie opérative on peut citer:

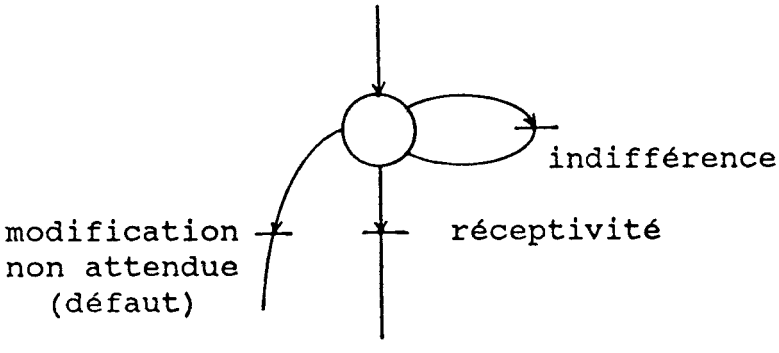
- détection des configurations inacceptables des capteurs,
- détection des variations non admissibles des capteurs en fonction de l'état de la commande,
- utilisation des temps de couverture des actions,
- comparaison de la séquence de comptes rendus à une séquence préenregistrée.
- test des sorties par rebouclage sortie/entrée.

5.4.2.Détection des configurations aberrantes des capteurs:

Certaines configurations des capteurs étant impossibles en fonctionnement normal, de tels évènements sont un indice de l'existence de défaillances. Il s'agit ainsi de surveiller le domaine statique d'évolution du système. Ceci n'est donc possible que si ce domaine d'évolution est plus petit que le domaine de définition constitué içi par toutes les combinaisons possibles des capteurs.

5.4.3. Surveillance de la variation des entrées: [37]

Il s'agit de vérifier que la variation d'une entrée est normale ou acceptable pour l'état actuel de la partie commande ou marquage du graphe de commande.



5.4.4 Temps de couverture des actions:

Chaque fois qu'une action est lancée, on enclenche une temporisation. celle-ci est réglée à une valeur légèrement supérieure au temps maximum nécessaire au déroulement de l'action en fonctionnement normal. La temporisation est désactivée dès que le compte rendu de fin d'action est reçu.

Ainsi la variable de temporisation ne doit jamais évoluer en fonctionnement normal. Tout blocage de l'évolution sera donc détecté.

5.4.5.séquence enregistrée:

On commence par enregistrer une séquence de comptes rendus R générée par la PO saine en réponse à une séquence d'entrée O. Pour éffectuer un test en ligne il suffit que la séquence O corresponde au cycle complet de la machine en cours d'exploitation. Le diagnostic consiste donc à comparer la séquence de sortie R' observée à celle R enregistrée.

Cette méthode ne peut être appliquée que si le cycle de fonctionnement du processus est strictement répétitif. En cas d'action simultannées la séquence n'est pas strictement répétitive. La fluctuation des durées des actions doit être prise en compte.

6. SECURITE DES AUTOMATISMES INDUSTRIELS

La sécurité tient une place particulière dans la sûreté de fonctionnement d'un système. Elle est régie par des textes législatifs visant essentiellement la protection des personnes.

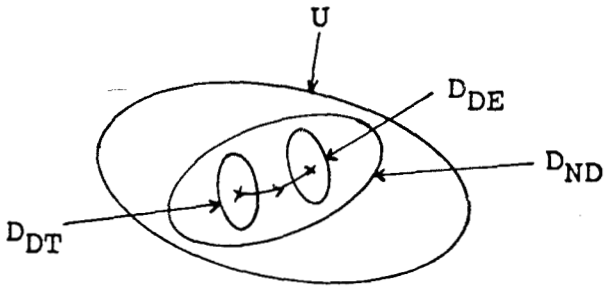
Un certain nombre de règles doivent être appliquées dès le début de la conception afin d'éviter les situations dangereuses que ce soit en exploitation normale ou en cas de défaillance.

Les accidents occasionnés par les machines dangereuses sont généralement dûs à la présence d'une partie du corps du travailleur dans la zone dangereuse durant la phase de travail. Malheureusement le danger peut englober une partie de l'espace bien plus grande encore (Tchernobyl).

6.1. Système sûr en présence de défaillance:

Certains comportements des systèmes en présence de pannes peuvent être dangereux . Une analyse détaillée des modes de défaillance doit être effectuée. En présence de panne activée, le comportement erronné (D_{DE}) du système doit appartenir à un ensemble D_{ND} ayant la propriété de "non danger" (figure I.30).

Figure I.30: comportement d'un système sûr en présence de faute.



On défini la notion de fonction à panne non dangereuse à 0 (ou 1) pour un système logique, lorsque, en présence de panne, la sortie ne peut prendre que la valeur 0 (respectivement 1) ou bien la valeur correcte du fonctionnement normal.

Définition I.5 Soit (0,1,N) un système ternaire, une machine séquentielle est à pannes non dangereuses sans décision pour un ensemble de pannes F si:

$$\forall f \in F, \forall \underline{x}_i \in D_{DX}, \forall q \in Q$$

soit $\lambda^f(\delta^f(\underline{x}_i,q)) = \lambda(\delta(\underline{x}_i,q))$ (fonctionnement correct)

soit $\lambda^f(\delta^f(\underline{x}_i,q)) = N$ (non danger)

6.2. Sécurité statique et sécurité dynamique:

Deux types d'actions peuvent être envisagés pour augmenter la sécurité.

a) La sécurité statique: Les éléments de la structure et leur cablage sont choisis de façon à ce que toute défaillance amène naturellement l'automatisme dans un état de fonctionnement de sécurité.

Cette méthode est appliquée en particulier en cas de coupure d'alimentation ou de liaisons pour éviter une évolution arbitraire des dispositifs (frein actif par coupure d'alimentation ...). Son principal avantage est de ne pas nécessiter de composants supplémentaires, mais l'état de sécurité ne peut pas toujours être atteint directement et naturellement. Il est alors nécessaire de faire dérouler une séquence de fonctionnement particulière.

b) La sécurité dynamique est obtenue par l'utilisation de mécanismes de détection et de réaction. La sécurité obtenue avec un système à tolérance de pannes relève de cette catégorie.

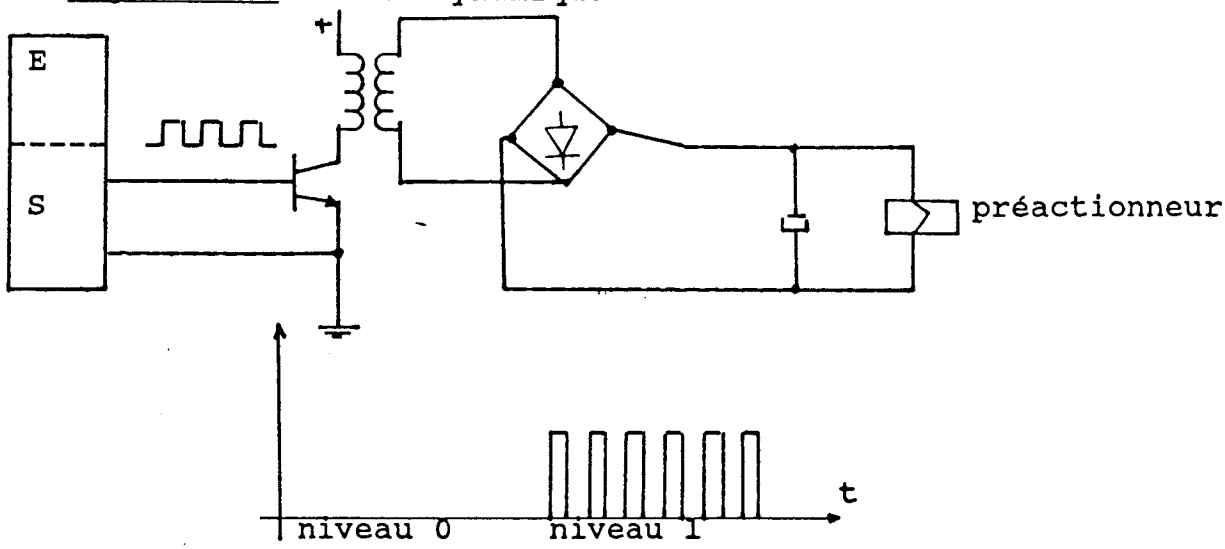
Le mécanisme de détection et de réaction permet de prendre des mesures d'exception afin d'amener l'automatisme dans un état de sécurité. Ceci nécessite donc des redondances afin d'assurer la continuité de service partielle ou totale, continuité qui peut être impérative.

6.3. Description de quelques montages destinés à améliorer la sécurité des automatismes:

6.3.1. Utilisation de sorties dynamiques [44]

La figure I.31 montre le montage utilisé. L'automate doit donc fournir un signal alternatif sur la sortie dynamique au lieu d'un niveau logique. Ceci est réalisé en complémentant l'état de la sortie à chaque cycle si celle-ci est à 1. Ce montage a un effet de filtrage et de chien de garde. Il permet la mise en sécurité en cas de défaut de l'unité centrale. La diminution de fiabilité occasionnée par l'adjonction de ce montage reste très faible.

Figure I.31 Sortie dynamique

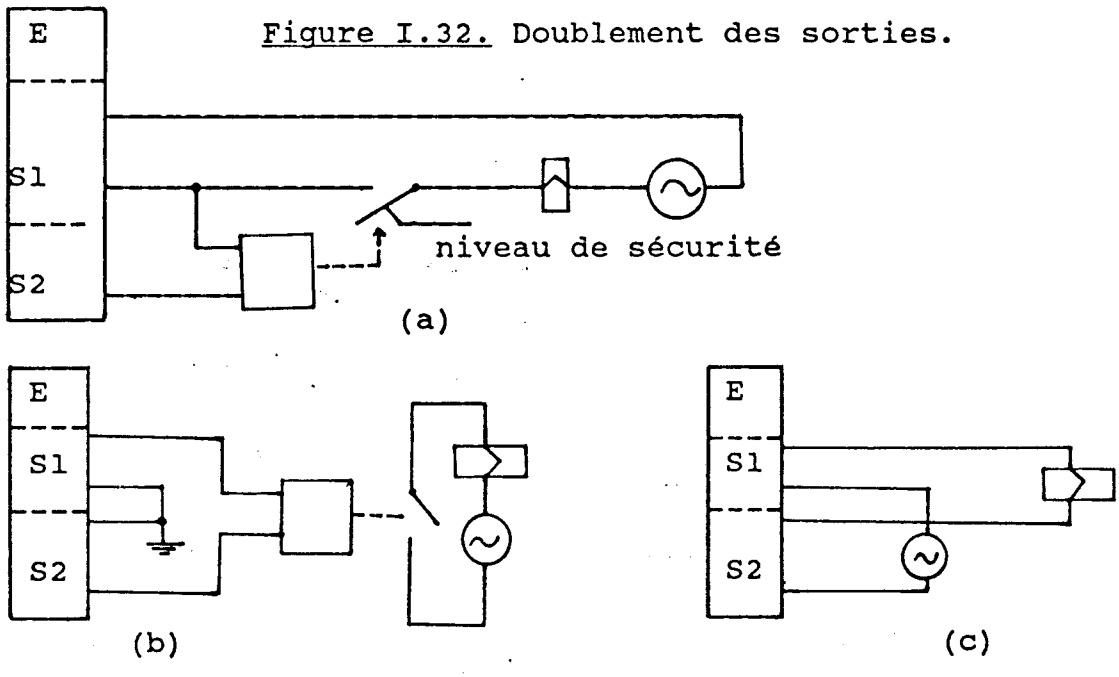


6.3.2 Doublement des sorties (figure I.32)

Il s'agit ici de l'utilisation de deux sorties de l'automate pour la commande du préactionneur. S'il y a discordance entre les valeurs fournies par ces deux sorties le circuit de contrôle commute alors pour la mise en sécurité. Le circuit de discordance peut être réalisé par un circuit OU ou ET. En pratique les montages des figures I.32.b, et c sont les plus utilisés.

Ce montage n'élimine que partiellement l'insécurité liée aux sorties. Il peut être amélioré par l'adjonction d'une signalisation en cas de discordonce. Le montage de la figure I.32.c permet la mise en sécurité en cas de coupure d'alimentation ou de liaison.

Figure I.32. Doublement des sorties.



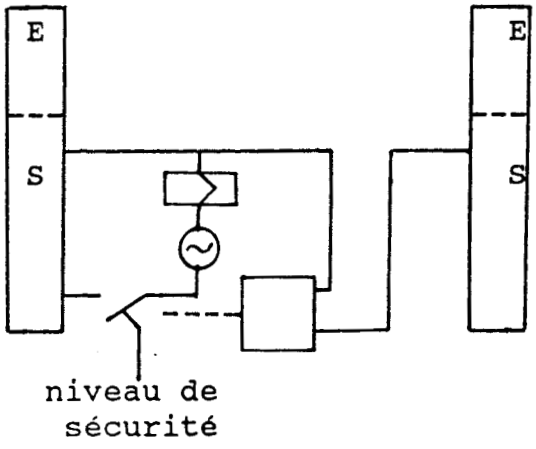
6.3.3. Utilisation de deux automates

Deux montages sont utilisés suivant que la continuité de service est ou non requise pour la sécurité.

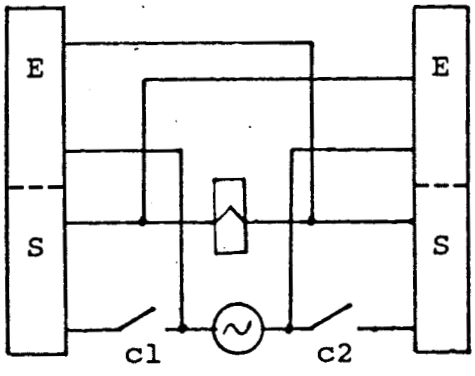
6.3.3.a. Placement en sécurité en cas de discordance (figure I.33)

Les deux automates fonctionnent simultanément ce qui suppose la résolution des problèmes de synchronisation. Ce montage utilise donc une redondance de type statique. Le principe du circuit ajouté reste le même que dans la méthode précédente (figure I.32) sauf qu'ici les deux sorties utilisées proviennent de deux automates différents. Suivant le niveau de sécurité on peut réaliser le forçage à "0" (figure I.33.b) ou à "1" avec le circuit de la figure I.33.c. Dans ce dernier cas, un automate constatant une erreur peut déconnecter l'autre ce qui necessite une surveillance mutuelle. Ceci peut être réalisé en particulier par un rebouclage Sortie-Entrée croisé.

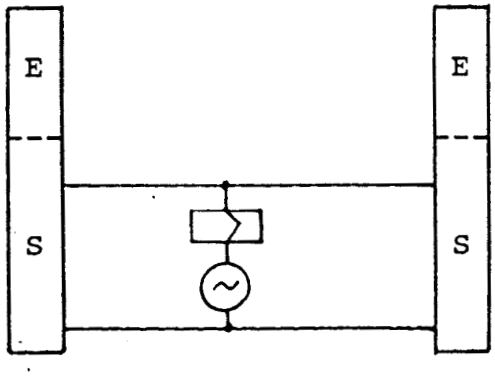
Figure I.33.Utilisation de deux automates.
la continuité de service n'est pas requise.



(a)



(b) forçage à 0.

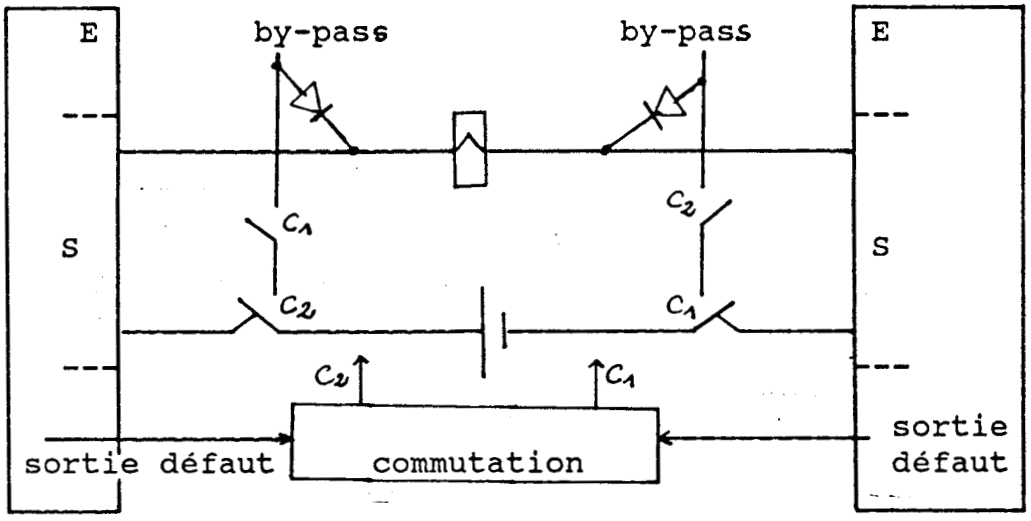


(c) forçage à 1.

6.3.3.b. Cas où la continuité de service est requise.
(figure I.34)

On utilise deux automates avec autotest. Ils participent tous les deux à la mission lorsqu'ils sont sains. Toute défaillance d'un automate provoque sa déconnexion. Nous sommes donc en présence d'une redondance sélective active. Le circuit de commutation constitue ici un point dur. La sécurité obtenue reste très bonne mais l'efficacité des autotests doit être excellente.

Figure I.34: Utilisation de deux automates en redondance sélective active.



6.3.4 : Utilisation de 3 automates en redondance massive:
(figure I.35)

La sécurité et la fiabilité ne sont limités pratiquement que par les performances du voteur et le circuit de commutation associé. Une détection de pannes et une localisation de l'automate défaillant est nécessaire.

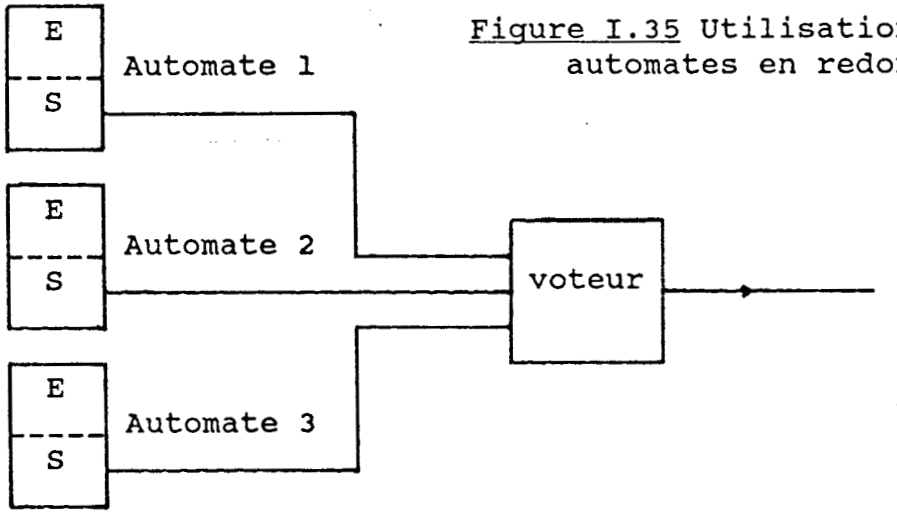
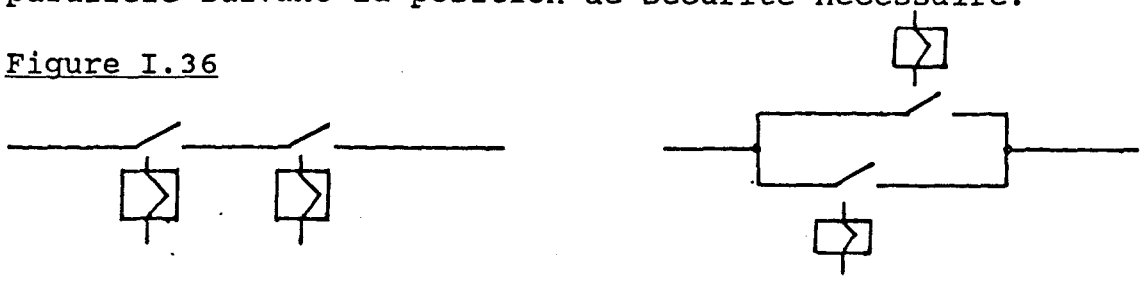


Figure I.35 Utilisation de trois automates en redondance massive

6.3.5. Amélioration de la sécurité au niveau des éléments de la partie opérative:

a) Au niveau des préactionneurs les montages les plus utilisés consistent à dupliquer ceux-ci et à réaliser un circuit série ou parallèle suivant la position de sécurité nécessaire.

Figure I.36



b) Au niveau des actionneurs il n'y a pas en général de redondance à cause du coût ou de l'encombrement. Chaque fois que cela est possible les préactionneurs et actionneurs doivent être choisis de façon à ce qu'en cas de coupure d'alimentation le dispositif prennne une position de sécurité.

c) Au niveau des capteurs et leurs liaisons, la sécurité se situe à deux niveaux: sûreté de l'information sur l'état du processus et protection de zone.

La redondance des capteurs permet de réaliser un codage adéquat. C'est en général l'automate qui teste l'appartenance au code (sécurité dynamique).

Lorsqu'il s'agit de capteurs analogiques, un filtrage des bruits de mesure est nécessaire avant comparaison.

7. SYNTHESE DE LA PARTIE COMMANDE ET SON INFLUENCE SUR LA SURETE DE FONCTIONNEMENT

7.1.Réceptivité et sensibilité d'un système:

Le cahier des charges d'un automatisme défini le séquencement des actions ou tâches en fonction d'évènements dits normaux qu'ils soient internes (comptes rendus de la PO) ou externes (consignes, ...). Ce séquencement prend en compte les différents modes de marche et certains fonctionnements d'exeption ou de sécurité.

Le comportement de la partie commande ainsi défini est presque toujours incomplètement spécifié. Certaines combinaisons des entrées ou séquence d'entrée sont en effet inaccessibles en l'absence de défaillances.

Cette spécification incomplète de la machine permet souvent une réduction du nombre d'états internes et une simplification des fonctions combinatoires. Mais cette pratique a un inconvéniant majeur: l'absence de prévision de l'évolution du système suite à l'apparition d'un évènement anormal pouvant être lié à une défaillance ou fausse manoeuvre.

Le comportement de la PC vis à vis des évènements peut être analysé par les notions de réceptivité et de sensibilité qui sont définies comme suit.

a) Réceptivité:
Un système placé dans un état déterminé est réceptif à une entrée si la variation de celle-ci peut provoquer un changement d'état interne.

b) Sensibilité:
Un système placé dans un état interne donné est dit sensible à une entrée si toute modification de cette entrée peut provoquer la variation d'au moins une sortie sans que le système change d'état interne.

c) Indifférence:
Si le système n'est ni réceptif, ni sensible à une entrée à un moment donné il sera dit indifférent à cette entrée.

La sensibilité caractérise donc la machine combinatoire (combinatoire local, général) alors que la réceptivité est relative à la partie séquentielle du système.

L'utilisation du GRAFCET et des réseaux de Pétri pour la spécification de la partie commande rend naturellement celle-ci réceptive et sensible aux seuls évènements normaux. La commande obtenue alors sera indifférente aux comptes rendus anormaux, ce

qui peut être bénéfique pour la sécurité par le fait que certaines erreurs ne sont pas prises en compte.

Exemple:

Soient les deux GRAFCETs de la figure I.37 représentant un même cahier des charges relatif à la commande d'un chariot.

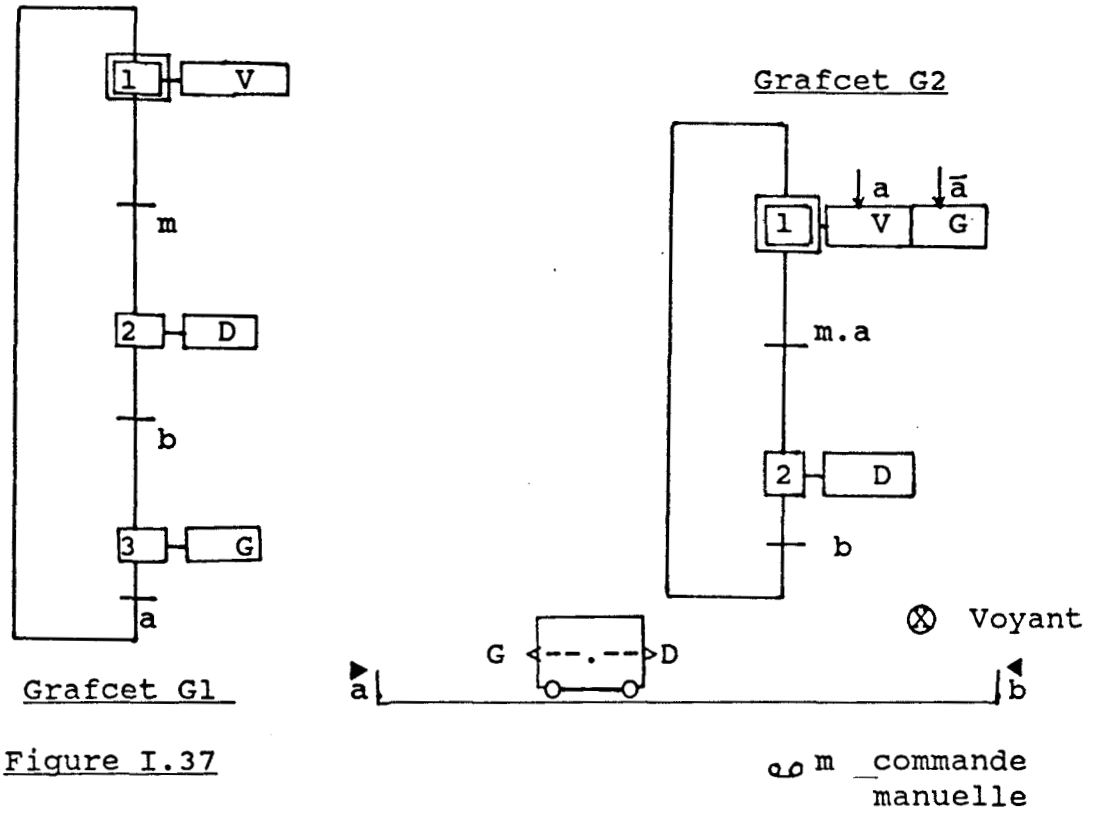


Figure I.37

Dans le Grafcet G1, si le système est dans l'état (1) (étape 1 active), il devient réceptif à l'entrée m puisque celle-ci peut faire évoluer le grafcet vers l'étape 2. Le grafcet G1 est ainsi successivement réceptif à b dans la situation (2) et à la variable a dans la situation (3). Ce graphe n'est sensible à aucune variable.

Le grafcet G2 dans la situation (1) est sensible à la variable a puisque celle-ci a une influence sur les sorties V et G sans qu'il y est changement d'état du grafcet.

On peut se poser la question de savoir s'il faut privilégier l'aspect réceptif ou l'aspect sensible en fonction de leur influence sur la sécurité.

On considère généralement que la réceptivité à une faute fugitive est plus gênante que la sensibilité à cette même anomalie. L'idée est qu'à la disparition de l'erreur, en cas de sensibilité, le système reprend son état normal alors qu'en cas de réceptivité, la modification de l'état interne est irréversible.

Cette affirmation est d'autant plus juste si on admet que la modification d'une sortie pendant un court instant est admissible parce que filtrée par le temps de réaction des préactionneurs ou de l'actionneur. Ceci n'est pas toujours le cas. Pour notre part nous considérerons que toute variation intempestive d'une sortie est contraire à la sécurité. Il appartient cependant au concepteur d'évaluer les risques encourus pour chaque application particulière.

On peut évaluer algébriquement l'importance de la réceptivité et de la sensibilité d'une machine par rapport aux variables d'entrée [51] afin d'aider le concepteur dans le choix d'une solution assurant une meilleure sécurité.

7.2.Estimation du comportement de la PC en présence de fautes non consistantes:

Soit t_i les durées pendant lesquelles le système est réceptif ou sensible à une entrée x_i pendant une période d'observation T. On suppose qu'au cours de cette période il existe une faute non consistante de durée t_f sur x_i. La probabilité pour qu'il existe une date $t \in [0,T]$ où le système est à la fois réceptif ou sensible à x_i et la faute est présente est donnée par:

$$P(f_t) = \frac{t_f}{T} \cdot \frac{t_i}{T}$$

Les deux évènements considérés sont en effet indépendants. $P(f_t)$ représente donc la probabilité qu'une panne non consistante de durée t_f sur l'entrée x_i soit prise en compte ou propagée par la PC.

Pour diminuer les risques de propagation de ces erreurs fugitives, il faudrait donc minimiser le rapport t_i/T. Pour caractériser le comportement de la PC à prendre en compte ce type de fautes on pourrait prendre ce rapport t_i/T mais son calcul est trop compliqué en pratique. On définira alors un estimateur permettant de comparer le comportement de différents grafcets de commande représentant le même cahier des charges en prenant les conventions suivantes.

a) On considère un cycle fictif tel que tout état atteignable est atteint en une seule fois et une seule.

b) Tous les états atteignables (non instables) sont supposés avoir la même durée σ.

c) Si un système est potentiellement réceptif (respectivement sensible) à une variable x_i pour un état donné, il sera considéré comme réceptif (resp. sensible) à cette variable.

Les conditions a) et b) permettent d'avoir l'équiprobabilité des états atteignables. La probabilité d'être dans un état instable est supposée nulle.

Pour un système à n états internes, soit m le nombre d'états pour lesquels le système est réceptif à une entrée x_i nous pouvons écrire d'après les 3 propositions ci-dessus.

$$T = n.\sigma \quad ; \quad t_i = m.\sigma \quad ; \text{ d'où } \quad R(x_i) = m/n$$

De même, si h est le nombre d'états pour lesquels le système est sensible à une entrée x_i, la sensibilité $S(x_i)$ est donnée par:

$$S(x_i) = h/n$$

7.3. Calcul de la susceptibilité par rapport à une entrée:

On définit un facteur de susceptibilité de la PC vis à vis des fautes non consistantes par:

$$C(x_i) = \frac{t_i}{T} = \frac{m + h - k}{n}$$

où k représente le nombre d'états pour lesquels le système est simultanément réceptif et sensible à x_i.

Exemples de calcul:

On considère les 2 grafcets de la figure I.37 précédente. Les Grafcets G1 et G2 donnent les résultats suivants où on constate que le grafcet G2 est globalement plus susceptible.

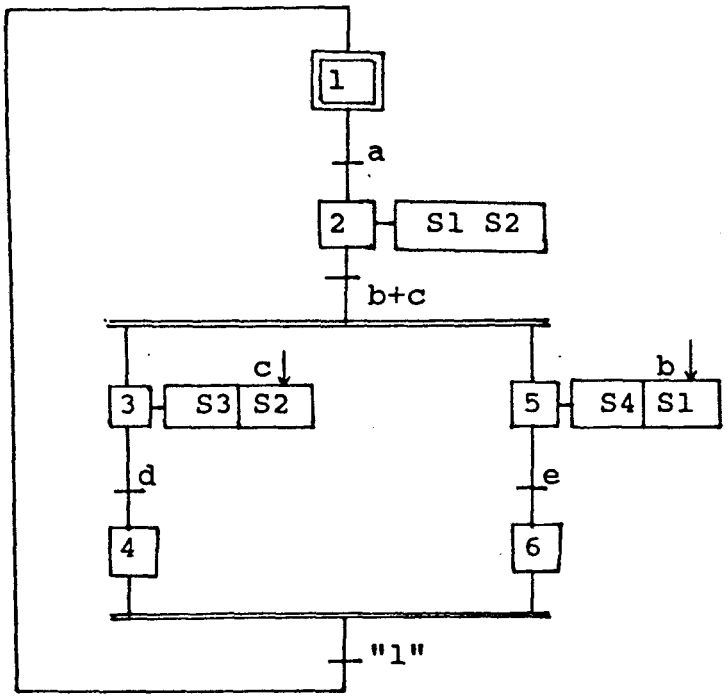
	grafcet G1			grafcet G2		
	a	b	m	a	b	m
R(x) Réceptivité	1/3	1/3	1/3	1/2	1/2	1/2
S(x) Sensibilité	0	0	0	1/2	0	0
C(x) susCeptibilité	1/3	1/3	1/3	1/2	1/2	1/2

Les situations du grafcet de la figure I.38 sont au nombre de 6 au total: {1}, {2}, {3,5}, {3,6}, {4,5}, {4,6}. La situation {4,6} instable éliminée, on trouve n = 5.

On trouve alors les différentes valeurs de la susceptibilité pour les différentes variables d'entrée:

	a	b	c	d	e
R(x)	1	1	1	2	2
S(x)	0	2	2	0	0
C(x)	1/5	3/5	3/5	2/5	2/5

Figure I.38: Exemple de calcul de susceptibilité



Le calcul des estimateurs R, S et C pouvant être fastidieux, un certain nombre de règles générales peuvent être mises en évidence [51] dont l'application permet d'obtenir naturellement un grafcet proche de l'optimum en ce qui concerne la susceptibilité. Ces règles sont les suivantes:

Règle 1:
Toute exploitation des spécifications technologiques visant à réduire le nombre d'états en rendant la commande moins réceptive mais plus sensible, est néfaste pour la sécurité.

Ceci est vérifié par la transformation du grafcet éffectuée à la figure I.37.

Règle 2:
La sensibilité doit être utilisée uniquement pour introduire des conditions d'inhibition qui sont fonctionnellement combinatoires.

Règle 3:
Toute modification attendue d'une entrée constituant un évènement normal, doit être prise en compte le plus possible dans une réceptivité. Dans la suite de l'évolution, il est préférable d'utiliser ce changement d'état interne plutôt que de reprendre l'état de la variable d'entrée.

7.3.Validation des commandes

L'utilisation d'outils de description tels que les réseaux de Pétri [33] et le Grafcet [30] permet une meilleure spécification de la commande. Devant la complexité toujours croissante des automatismes et malgré différentes améliorations de ces outils notamment par une approche modulaire [31] et une analyse descendante proche de la programmation structurée, une phase de validation des logiciels de commande est nécessaire. Cette phase permet de:

- vérifier que l'automatisme répond bien aux spécifications du cahier des charges,
- s'assurer qu'aucune évolution ne puisse aboutir à un fonctionnement dangereux y compris éventuellement en présence de défaillances les plus probables,
- affiner les stratégies de conduite.

On distingue différents types de validations:
- par dialogue,
- par preuve formelle,
- par analyse,
- par simulation.

a) Dans la validation par dialogue, utilisateur (spécialiste du processus) et automaticien (concepteur de la commande) analysent en commun le comportement de l'automatisme.

b) La validation par preuve formelle consiste à utiliser deux outils différents pour la spécification et comparer les résultats des deux descriptions.

c) La validation par analyse est utilisée principalement pour les descriptions par réseaux de Pétri. Il s'agit de vérifier un certain nombre de propriétés caractéristiques du "bon" fonctionnement du système.
Parmi ces propriétés on peut citer: [33]

- la vivacité: elle exprime le fait qu'au cours du fonctionnement du système tout évènement interne peut se reproduire; en particulier à partir de tout état successeur de l'état initial.

- la propriété de borné: elle se traduit pour les réseaux de Pétri par le fait que chaque place ne peut avoir qu'un nombre fini de marques pour toute évolution à partir du marquage initial.

- propriétés relatives aux partages de ressources: La mise en évidence des situations où il y a conflit constitue un des

problèmes les plus cruciaux dans l'analyse des systèmes à évolutions parallèles.

- propriété d'interdépendance entre évènements internes: Un système peut être vivant et admettre une séquence infinie de marquages pour lesquels une de ses transitions n'est jamais validée. De tels cas traduisent des situations de coalition ou de famine.

La vérification de toutes ces propriétés sur un réseau de Pétri peut être réalisée par:
- énumération des marquages,
- analyse structurelle,
- analyse par transformation.

Les procédures d'analyse par transformation [53], consistent à transformer le réseau initial (R,Mo) en un réseau (R',Mo') tel que:
- si (R',Mo') satisfait la propriété Pr à vérifier, alors (R,Mo) la satisfait nécessairement;
- la propriété Pr est plus facilement vérifiable sur (R',Mo') que sur le réseau initial.

d) La validation par simulation permet d'intégrer l'interprétation liée au réseau. Elle s'avère nécessaire en pratique pour l'évaluation du comportement du système. Elle permet d'acquérir certaines assurances sur la façon dont la conception réalise les objectifs assignés, sinon de révéler les lacunes ou situations mal résolues ou dangereuses y compris en présence de défaillances.
Cette simulation peut être réalisée de différentes manières:

- Dans la procédure interactive, l'opérateur par le forçage des entrées simule les réactions de la partie opérative et observe l'évolution des commandes. Mais devant le grand nombre de situations à passer en revue et l'absence d'un modèle de comportement réel de la PO, cette simulation est souvent très imcomplète.

- Dans une simulation automatique il est nécessaire d'avoir un modèle de la PO. On utilise souvent dans ce cas un graphe dual de celui de la commande. Ce qui présente un inconvéniant majeur puisqu'il représente la PO dans le contexte de la commande. Les véritables valeurs des entrées étant inconnues, il devient impossible de simuler l'arrivée d'évènements anormaux.
Un modèle de comportement réel de la PO vis à vis de la commande tel que nous le présentons dans la suite de cette étude pourrait améliorer notablement l'efficacité de cette simulation.

La création et la validation des logiciels de commande passent par:

- l'utilisation d'outils de spécification et de description (type Grafcet) et une méthode d'analyse qui reste encore à affiner.
- la création de simulateurs utilisant un véritable modèle de comportement de la partie opérative et de ses interfaces avec la commande.
- la génération automatique des programmes d'application à partir des descriptions ainsi validées.

CONCLUSION SUR LE PREMIER CHAPITRE

Dans ce premier chapitre nous avons rappelé ce qu'est la sûreté de fonctionnement. L'évaluation de ses différentes composantes dans le cas des systèmes réparables permet d'étudier l'influence des différents facteurs et de mieux choisir une architecture.

Dans le cadre des automatismes industriels il est souvent plus avantageux d'augmenter la disponibilité en agissant sur le taux de réparation par la mise en oeuvre de dispositifs de test par exemple que d'utiliser des redondances fonctionnelles. L'utilisation de redondances ne s'impose vraiment que lorsque la continuité de service est requise pour assurer la sécurité. La détection de pannes est nécessaire dans tous les cas, pour assurer la reconfiguration et/ou la mise en sécurité.

Les mécanismes d'autotest mis en oeuvre au niveau de l'unité de traitement sont relativement classiques, par contre pour les éléments hors automate (interfaces, partie opérative) les solutions généralement proposées consistent à dupliquer ceux-ci. Or il s'avère que ces éléments sont les plus pénalisants pour la fiabilité de part leur nombre et leur taux de pannes.

Dans la conception du logiciel de commande, l'analyse de la susceptibilité du système permet d'obtenir des solutions adéquates minimisant l'influence des pannes non consistantes.

L'utilisation d'un modèle de comportement de la partie opérative et de ses interfaces vis à vis de la commande dans la simulation en phase de validation peut améliorer notablement l'efficacité de celle-ci.

La suite de notre étude porte sur la définition d'un modèle de comportement de la partie opérative et son utilisation pour la réalisation d'un mécanisme de test en ligne. Dans la deuxième partie de l'étude une procédure d'apprentissage automatique du modèle de la partie opérative sera proposée.

CHAPITRE II: MODELISATION ET DIAGNOSTIC DE LA PARTIE OPERATIVE

1. GRAMMAIRE ET LANGAGE

1.1. Alphabet
1.2. grammaire:définition
1.3. Automate d'état fini

2. ANALYSE ET MODELISATION DU COMPORTEMENT DE LA PO

2.1. Grandeurs mesurées et trajectoires
2.2. Comptes rendus associés aux points singuliers
2.3. Domaine d'évolution dynamique d'une grandeur mesurée
2.4. Prise en compte de la commande dans le comportement dynamique de la partie opérative
2.5. obtention du modèle localisable de la PO
2.6. Interactions entre trajectoires des grandeurs mesurées
2.7. Prise en compte de la durée des actions
2.8. Modélisation de la partie "préactionneurs".

3. TEST EN LIGNE ET DIAGNOSTIC DE LA PARTIE OPERATIVE

3.1. Différentes possibilités de test en ligne.
3.2. Analyse syntaxique des CR en dehors du contexte de la commande.
3.3. Analyse syntaxique des CR avec prise en compte des commandes appliquées.
3.4. Analyse syntaxique des CR avec prise en compte de la durée de maintien dans un état.
3.5. Analyse syntaxique des CR avec prise en compte de la durée et des commandes appliquées.
3.6. Etude des cas où il y a interactions entre trajectoires.

CHAPITRE II

MODELISATION ET DIAGNOSTIC DE LA PARTIE OPERATIVE

Dans ce deuxième chapitre nous analysons le comportement de la partie opérative et nous proposons l'élaboration d'un modèle de comportement entrée/sortie vue par la partie commande. Il existe en effet des modèles de description de la PO vue par la partie gestion concernant les objets qui y circulent. On peut citer l'utilisation en particulier des réseaux de Pétri colorés qui permettent d'inclure la fonction de gestion des objets dans la commande [54].

La modélisation du comportement de la partie opérative est basée sur l'observation de l'évolution d'un certain nombre de grandeurs physiques à travers les capteurs. L'identification des trajectoires suivies par ces grandeurs sous l'action des commandes appliquées et les comptes rendus associés, permettent d'obtenir un modèle d'évolution dynamique de la PO. Ce modèle est ensuite complété par l'introduction du temps de réponse de la PO ou durées des actions, pour mieux caractériser cette évolution dynamique.

Ce modèle d'évolution est à la base de la procédure de test en ligne proposée dans ce chapitre. Nous étudions différents modèles de détection des défaillances de la partie opérative. On commence par une simple analyse syntaxique des comptes rendus. On prend ensuite en compte les commandes appliquées. La dernière procédure introduit la durée des actions.

1.LANGAGE ET GRAMMAIRE [23],[24]

1.1 Alphabet

On appelle **alphabet** un ensemble fini, noté X de symboles ou lettres. Ces symboles sont notés par des lettres minuscules.

Une **phrase** sur l'alphabet X, notée $\underline{x}$ est une suite ordonnée d'éléments de X (concaténation des éléments de X).

X = {a,b,c,d}; $\underline{x}$ = abbac

La **longueur** d'une phrase $\underline{x}$ notée $|\underline{x}|$ est le nombre d'éléments qu'elle comporte: $\underline{x}$ = abbaabc ; $|\underline{x}|$= 7. La phrase vide notée Λ a une longueur nulle.

La phrase $\underline{x}$ = aaaa...a, (n fois) est notée a^n par convention. L'ensemble des phrases sur l'alphabet X est noté X^*. La notation X^+ désigne l'ensemble des phrases non vides formées des éléments de X.

On appelle **langage** sur un alphabet X toute partie de X^*.

L = {a,ab,aab, ...}; X = {a,b}

1.2.Grammaire

Une **grammaire** est définie par un quadruplet:

G = (V_N, V_T, R, S) où:

V_T est l'alphabet **terminal** sur lequel sont écrites les phrases du langage.

V_N est un ensemble fini de symboles non terminaux ou alphabet **auxiliaire**. $V_N \cap V_T = \emptyset$; $V_N \cup V_T = V$.

S est un élément de V_N appelé symbole initial ou **axiome** à partir duquel toute génération commence.

R est un ensemble fini de **règles de production** ou de réécriture qui sont notées sous la forme: $\alpha \rightarrow \beta$ où α et β sont des phrases écrites sur $V_N \cup V_T$ soit $\alpha,\beta \in (V_N \cup V_T)^*$.

Une règle de production permet donc de transformer certaines phrases écrites sur V^* en d'autres. Le procédé génératif associé à la grammaire consiste à partir de l'axiome S à le réécrire en une phrase $\alpha 1$ par une des règles de R. Puis à réécrire $\alpha 1$ en $\alpha 2$ par une des règles de R, etc...

Si une phrase αk n'est composée que de symboles de V_T, on ne peut plus la réécrire.

On définit le langage L(G) généré par une grammaire G comme:

$$L(G) = \{ \underline{x} \mid \underline{x} \in (V_T)^* \text{ et } S \overset{*}{\underset{G}{\Longrightarrow}} \underline{x} \}$$

L'expression $\alpha \overset{*}{\underset{G}{==>}} \beta$ signifie que β est dérivable à partir de la chaine α par application des règles de R de la grammaire G.

Exemple:

Soit la grammaire $G = (V_N, V_T, R, S)$ avec

$V_T = \{a,b\}$; $V_N = \{S,A\}$

$$R = \begin{cases} S \rightarrow aS & (1) \\ S \rightarrow bA & (2) \\ bA \rightarrow bbA & (3) \\ A \rightarrow a & (4) \end{cases}$$

En partant de l'axiome S on peut écrire:

$$S \overset{(1)}{==>} aS \overset{(1)}{==>} aaS \overset{(2)}{==>} aabA \overset{(3)}{==>} aabbA \overset{(4)}{==>} aabbba$$

d'où aabbba $\in$ L(G)

Une grammaire est dite **régulière** lorsque ses règles de production sont de la forme:

$\alpha \rightarrow \beta$ avec $\alpha \in (V^* x V_N x V^*)$ et $\beta \in V^*$

1.3.automate fini

Un automate fini est défini par un quintuplet

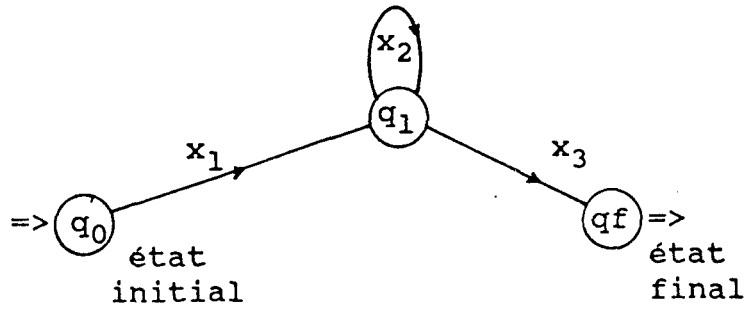
$A = (X, Q, \delta, q_0, Qf)$ où

- X est un alphabet (d'éléments terminaux);
- Q est l'ensemble fini des états de l'automate;
- $q_0 \in Q$ est l'état initial;
- $Qf \subset Q$ est l'ensemble des états finals;
- δ: $Q x X \rightarrow Q$ est la fonction de transition.

Représentation graphique:

Chaque élément $q \in Q$ est représenté par un cercle et chaque transition notée (q_i, x, q_j) avec $q_i, q_j \in Q$ et $x \in X$, est symbolisée par un arc orienté.

Exemple



Un automate fini sert à définir un langage sur X. L'état initial de l'automate correspond à l'axiome de la grammaire. Les deux alphabets X et V_T sont identiques. Les règles de production correspondent à la fonction de transition de la façon suivante.

Exemple

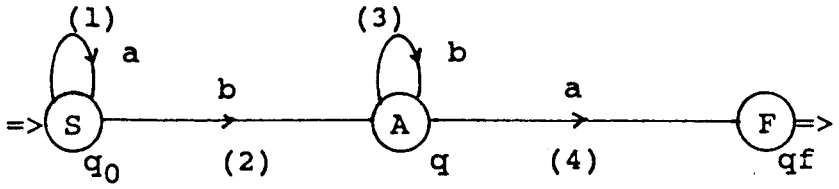
Soit $G = (V_N, V_T, R, S)$ avec $V_N = \{S,A\}$ et $V_T = X = \{a,b\}$

Les règles de production sont les suivantes:

(1) S -> aS (3) A -> bA

(2) S -> bA (4) A -> a

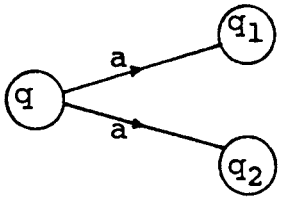
L'automate correspondant est représenté par le graphe suivant:



Automate non déterministe

Un automate est dit **non déterministe** lorsque sa fonction de transition est une application multivoque: à un couple $(q,x) \in Q \times X$ peut correspondre un ensemble d'états.

$\delta(q,a) = \{q_1, q_2\}$



De plus, la fonction peut ne pas être définie pour tout couple (q,x), l'automate est alors dit **incomplètement spécifié** ou **incomplet.**

Automate dérivé

Soit **A** = (X, Q, q_0, δ, Qf) un automate fini et P =(P1,...,Pr) une partition de l'ensemble Q. On appelle **automate dérivé** de **A** pour la partition P l'automate fini **A'** = (X, P, δ', P0, R) tel que:

a) P0 $\in$ P est tel que $q_0 \in$ P0

b) R = {Pi $\in$ P | $\exists$ $q_j \in$ Qf, $q_j \in$ Pi}

c) δ'(Pi,x) = Pj si $\exists$ $q_k \in$ Pi et $q_e \in$ Pj tels que $\delta(q_k,x)=q_e$

L'automate dérivé est donc construit en confondant certains états de l'automate initial. Cette dérivation est à la base des méthodes d'inférence. On a en particulier: L(**A**) $\subseteq$ L(**A'**).

Analyse syntaxique

L'**analyse syntaxique** d'une phrase $\underline{x} \in X^*$ est l'opération qui consiste à vérifier si celle-çi appartient ou non au langage L(G) engendré par la grammaire G.

Dans l'automate d'état fini **A**, une phrase $\underline{x} = a_1 a_2 \ldots a_n$ appartient au langage associé **L(A)** s'il existe un chemin d'origine q_0 et d'extrémité qf $\in$ Qf tel que ses arcs portent les étiquettes $a_1, a_2, \ldots, a_n$ dans cet ordre.

2.ANALYSE ET MODELISATION DU COMPORTEMENT DE LA PARTIE OPERATIVE

2.1.grandeurs mesurées et trajectoires

Dans un processus industriel automatisé (figure II.1), l'évolution de certaines grandeurs physiques est repérée par des capteurs qui sont en général de type tout-ou-rien dans le cas de processus à évolution séquentielle. Cette évolution est provoquée par les actionneurs.

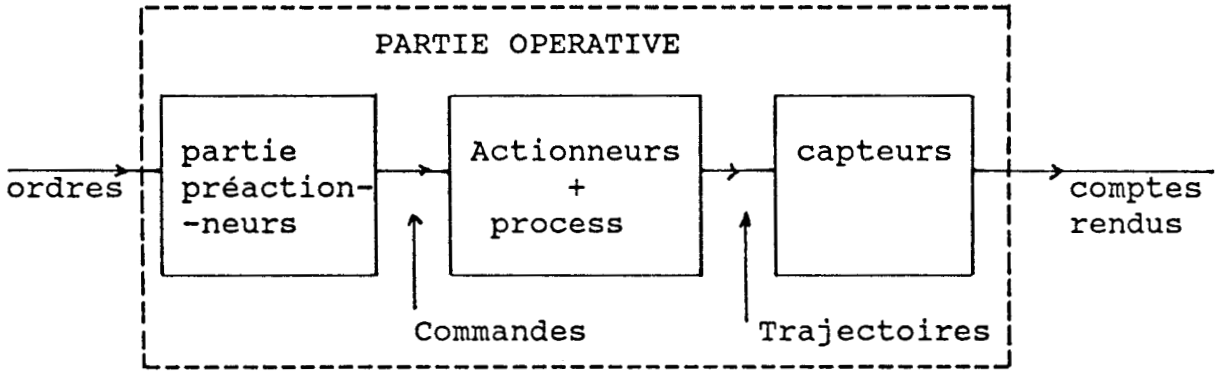


Figure II.1: grandeurs mesurées et trajectoires.

Trajectoire: Définition

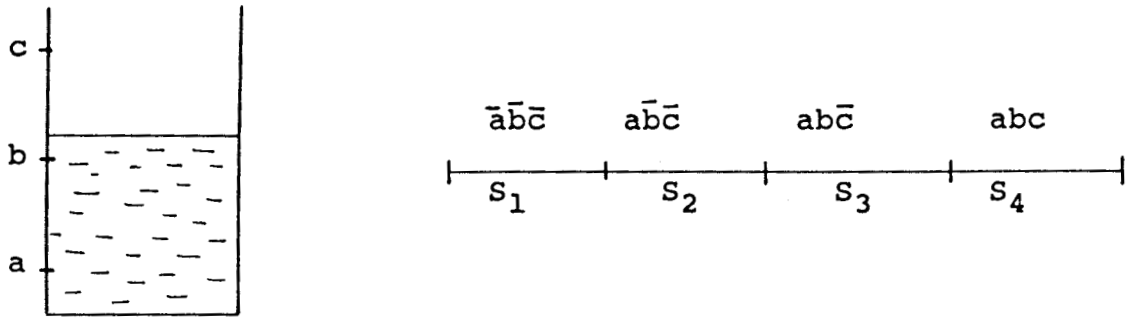
On appelle **trajectoire** associée à une grandeur mesurée, l'ensemble ordonné des valeurs que peut prendre cette grandeur physique lors d'une évolution monotone entre deux extrémums et selon un seul degré de liberté.

Les grandeurs mesurées sont en général discrétisées soit par les capteurs soit à l'aide de convertisseurs A/D.

Exemple (figure II.2)

L'utilisation de 3 capteurs de type tout-ou-rien (TOR) permet de repérer le niveau d'un liquide dans un réservoir en 4 valeurs distinctes. Ce niveau peut être mesuré par un capteur analogique. L'utilisation d'un convertisseur A/D 4 bits permet alors de discrétiser la trajectoire en 16 valeurs distinctes.

Figure II.2 discrétisation de la trajectoire d'une grandeur mesurée par les capteurs.



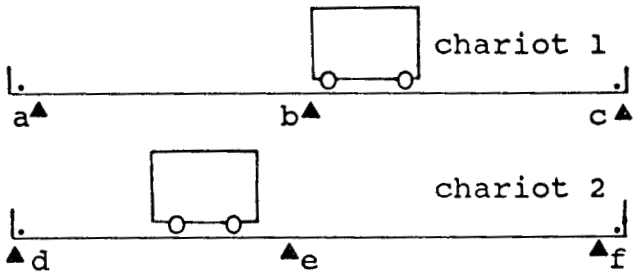
Point singulier de trajectoire

On appelle **point singulier** (PS) d'une trajectoire l'ensemble des points atteignables au cours de l'évolution d'une grandeur mesurée sans modification de compte rendu.

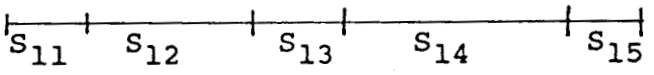
Dans le cas de la figure II.2, on peut distinguer 4 points singuliers notés $S_1,\ldots, S_4$ avec leurs comptes rendus associés.

L'ensemble ordonné des points singuliers constitue alors une trajectoire singulière qui est la discrétisation de la trajectoire réelle. Celle-ci a une dimension finie en général.

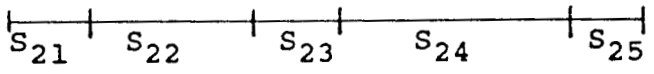
Figure II.3: Evolution de deux chariots sur deux voies ferrées distinctes.



trajectoire associée à la position du chariot 1



Trajectoire associée à la position du chariot 2

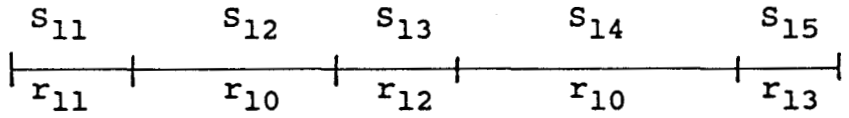


2.2.comptes rendus associés aux points singuliers

Une partie opérative peut comporter plusieurs grandeurs mesurées. Soit k ce nombre. Pour chaque trajectoire T_i, caractérisant l'évolution d'une grandeur G_i, on défini l'**alphabet de compte rendu** R_i comme l'ensemble des configurations possibles des capteurs modifiables par cette grandeur.

A la trajectoire du chariot 1 de la figure II.3 est associé l'alphabet $R_1 = \{r_{10}=\bar{a}\bar{b}\bar{c}, r_{11}=a\bar{b}\bar{c}, r_{12}=\bar{a}b\bar{c}, r_{13}=\bar{a}\bar{b}c\}$. Cet alphabet représente donc le domaine statique d'évolution du sous-système et est défini sur l'ensemble des combinaisons binaires des capteurs a, b, c.

Un codage c: R_i -> S_i établit une correspondance entres comptes rendus et points singuliers, chaque point singulier ayant un seul compte rendu associé par définition. Cette correspondance n'est pas biunivoque, en général: un même compte rendu peut être associé à plusieurs points singuliers.
L'exemple de la figure II.3 donne la correspondance suivante:



Les points singuliers S_{12} et S_{14} ont même compte rendu associé.

L'ensemble de la partie opérative a un alphabet de trajectoire **S** qui est le produit cartésien des alphabets de trajectoire S_i associée à chaque grandeur.

$$\mathbf{S} = \bigtimes_{i=1}^{k} S_i$$

De même, l'alphabet de compte rendu **R** d'une partie opérative comportant k grandeurs mesurées sera la produit cartésien des alphabets R_i de compte rendu associés au trajectoires:

$$\mathbf{R} = \bigtimes_{i=1}^{k} R_i$$

Exemple:

La partie opérative décrite à la figure II.3 possède l'alphabet de compte rendu suivant:

- alphabet associé à T_1:
  $R_1 = \{r_{10}=\bar{a}\bar{b}\bar{c}, r_{11}=a\bar{b}\bar{c}, r_{12}=\bar{a}b\bar{c}, r_{13}=\bar{a}\bar{b}c\}$
- alphabet associé à T_2:
  $R_2 = \{r_{20}=\bar{d}\bar{e}\bar{f}, r_{21}=d\bar{e}\bar{f}, r_{22}=\bar{d}e\bar{f}, r_{23}=\bar{d}\bar{e}f\}$

L'alphabet associé à l'ensemble de la PO est donc:
$R=R_1 x R_2 = \{r_{10}xr_{20}=\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f},\ r_{10}xr_{21}=\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f},\ \ldots,\ r_{13}xr_{23}=\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f}\}$

On peut remarquer içi l'explosion combinatoire occasionnée par le produit cartésien traduisant l'évolution en parallèle des deux sous-processus. Pour éviter ceci, nous maintiendrons la décomposition en sous parties opératives ou grandeurs mesurées et nous envisagerons leurs interactions éventuelles (partage de trajectoire, modification d'une trajectoire par une autre).

2.3.domaine d'évolution dynamique d'une grandeur mesurée

La trajectoire discrétisée représente donc les évolutions possibles d'une grandeur mesurée. Cette évolution peut être représentée par un graphe de transition ou automate (fig.II.5).

Exemple:
L'évolution du chariot 1 de la figure II.3 peut être représentée par le graphe de transition de la fig.II.5 suivant:

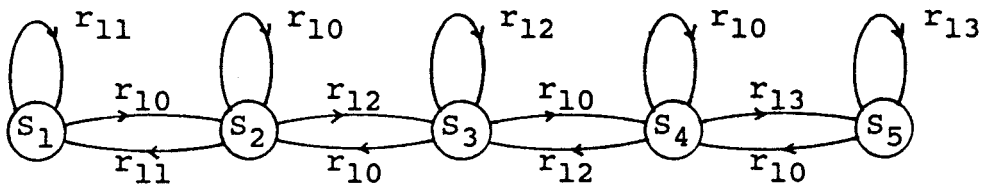


Figure II.5: Evolution dynamique d'une grandeur.

L'état de la grandeur mesurée est représenté par le sommet du graphe correspondant au point singulier atteint. Ce graphe est souvent non déterministe. Pour le rendre déterministe et pouvoir prévoir l'évolution de la grandeur, la prise en compte de la commande appliquée est nécessaire.

2.4.prise en compte de la commande dans le comportement dynamique de la partie opérative

Le sens d'évolution d'une grandeur mesurée est déterminé par la commande appliquée aux actionneurs. L'ensemble des commandes C agissant sur une grandeur peuvent être réparties en trois classes:

- C_0 : les commandes d'arrêt ou à vitesse nulle;
- C_P : les commandes faisant évoluer la grandeur dans un sens positif (ce sens peut être choisi arbitrairement);
- C_N : les commandes ou vitesses de sens négatif.

Le nombre de commandes dans chaque sens est en général limité. La connaissance de l'état de la grandeur mesurée et la commande appliquée permet de déterminer sans ambiguité l'état suivant et donc le compte rendu attendu.

Exemple

L'introduction de la commande dans le graphe de la fig.II.5 précédent permet d'obtenir le graphe déterministe suivant:

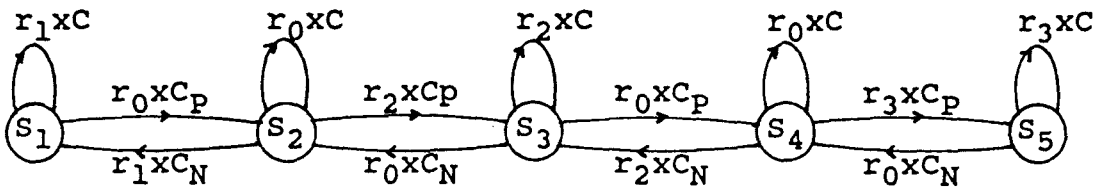


Figure II.6: prise en compte de la commande.

2.5.Obtention du modèle localisable de la partie opérative

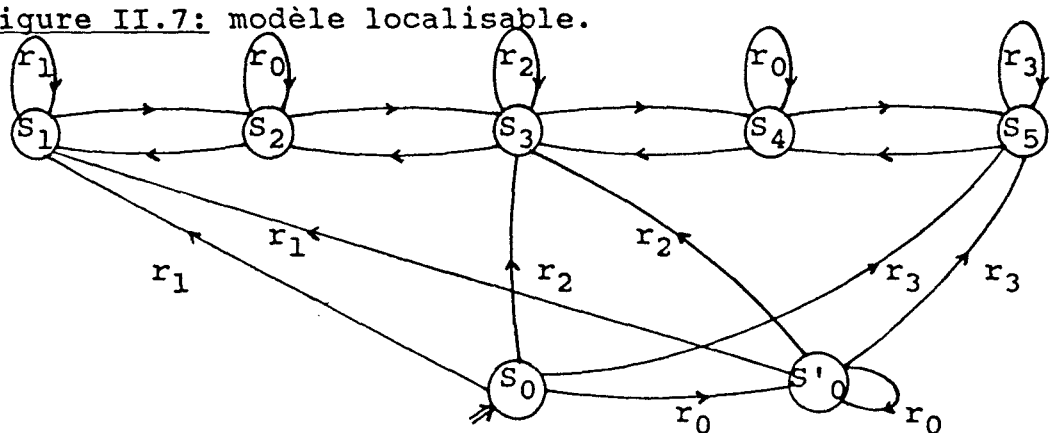
Le suivi de l'évolution de la partie opérative nécessite d'avoir un modèle localisable. On crée pour cela un état initial noté S_0 représentant le point d'entrée dans le modèle (à la mise sous tension par exemple).

Pour localiser l'état de la grandeur mesurée ou état de l'automate il existe des méthodes de recherche des séquences de synchronisation (homing sequences) [1] et notamment par utilisation des matrices de transitions d'ordre supérieur (annexe 1).

Exemple

Au graphe de la fig.II.6 on peut associer le modèle localisable de la fig.II.7. Les sommets S_1, S_3, S_5 ont des séquences de synchronisation de longueur 1. Les sommets S_2 et S_4 ayant les mêmes comptes rendus associés ne peuvent être caractérisés que par des séquences de longueur 2.

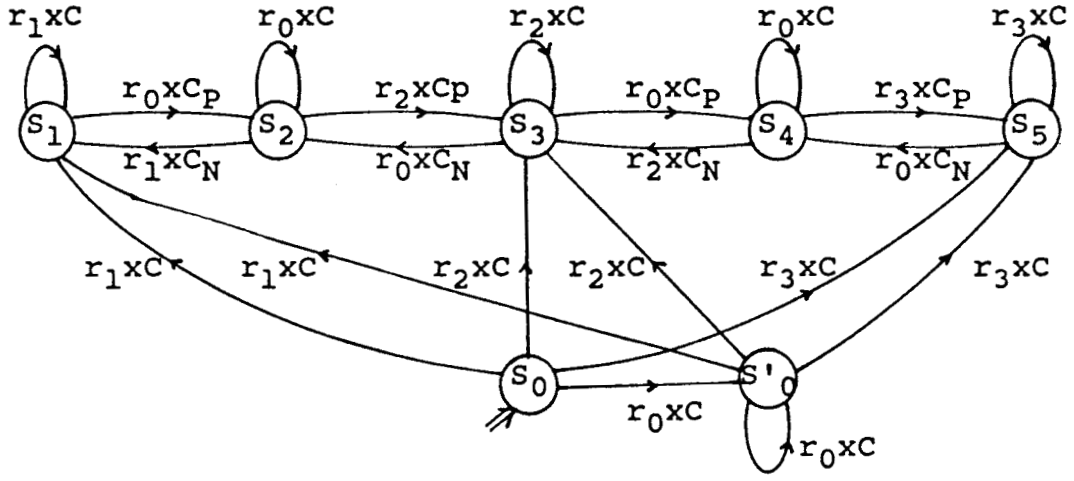
Figure II.7: modèle localisable.



A ce niveau l'état S'_0 ajouté représentant le modèle dans l'état S_2 ou S_4 n'améliore guère la localisation mais peut être utile pour le test.

Le modèle prenant en compte la commande appliquée peut être localisé de la même manière (fig.II.8). La connaissance de la commande peut éventuellement aider à localiser l'état présent plus rapidement.

Figure II.8: modèle localisable avec prise en compte de la commande.



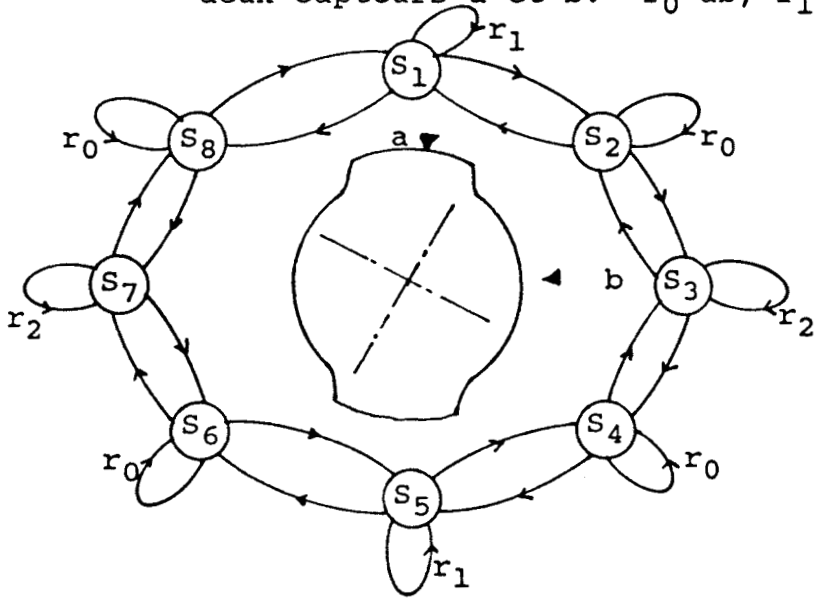
Cas particuliers:

Certains modèles ne sont pas localisables, c'est le cas de l'exemple de la figure II.9 où l'on est en présence d'une trajectoire fermée pour toute commande non nulle.

Une solution consiste à choisir arbitrairement un des états parmi S_1, S_5 si on a r_1 ou parmi S_3, S_4 si le compte rendu est r_2. Ce choix arbitraire n'est pas néfaste s'il y a symétrie de la came.

Une condition nécessaire pour qu'un modèle soit localisable est que la trajectoire soit ouverte.

Figure II.9: la position d'une came en rotation est repérée par deux capteurs a et b. $r_0=\bar{a}\bar{b}$, $r_1=a\bar{b}$, $r_2=\bar{a}b$



2.6.Interactions entre trajectoires des grandeurs mesurées

Nous avons supposé jusqu'à présent que l'ensemble de la partie opérative peut être décomposé en sous-processus "indépendants" évoluant en parallèle. Des interactions entre sous processus peuvent exister, se manifestant par:

- un partage de trajectoire,
- une modification de la trajectoire d'une grandeur par une autre.

a) Partage de trajectoire:

Exemple La figure II.10 représente une partie opérative avec deux chariots évoluant sur deux voies ferrées comportant un tronçon commun avec aiguillage. La position du chariot 1 est repérée par les capteurs a, b, c et d. Celle du chariot 2 est repérée par les capteurs a, b, e et f. La position de l'aiguillage est repéré par un capteur g.

Les alphabets de compte rendu étant définis sur des ensembles de capteurs différents, les trajectoires suivies par les deux grandeurs peuvent être différentiées.

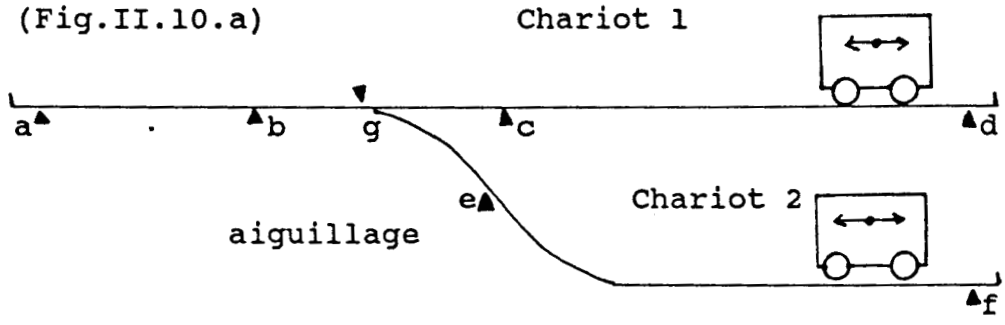
Mais un problème de codage des points singuliers se pose. Ce codage peut être réalisé de deux manières différentes suivant que la partie commune est ou non accessible simultanément aux deux grandeurs. La figure II.10 montre le codage adopté dans le cas où un seul chariot peut se trouver sur la partie commune.

Lorsque la partie commune est accessible simultanément aux deux grandeurs on supposera que celles-ci ne peuvent évoluer parallèlement vers le même point de façon à produire le même évènement. Dans ce cas le codage pour la partie commune est réalisé de façon à détecter les seuls évènements attendus pour le changement de point singulier. La figure II.10.e donne un exemple d'un tel codage.

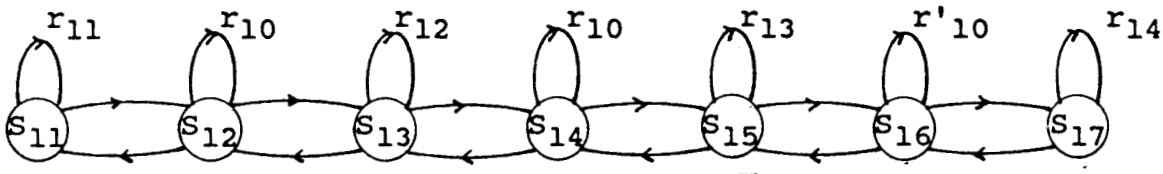
Les séquences de synchronisation dans le graphe d'initialisation de chaque sous-processus doivent tenir compte des capteurs partagés. Dans le cas extrême où l'ensemble de la trajectoire est partagé, il peut y avoir une impossibilité de localisation. On doit alors faire intervenir d'autres informations pour résoudre ce problème.

Figure II.10: partage de trajectoire.

(Fig.II.10.a)

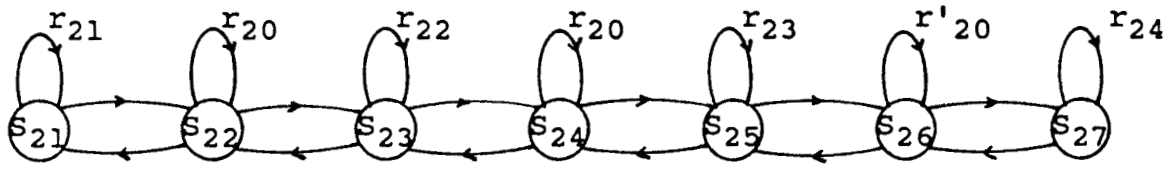


Trajectoire position chariot 1: (Fig.II.10.b)



$r_{10}=\bar{a}\bar{b}\bar{c}\bar{d}$, $r'_{10}=\bar{c}\bar{d}$, $r_{11}=a\bar{b}\bar{c}\bar{d}$, $r_{12}=\bar{a}b\bar{c}\bar{d}$, $r_{13}=c\bar{d}$, $r_{14}=\bar{c}d$

Trajectoire position chariot 2: (fig.II.10.c)



$r_{20}=\bar{a}\bar{b}\bar{e}\bar{f}$, $r'_{20}=\bar{e}\bar{f}$, $r_{21}=a\bar{b}\bar{e}\bar{f}$, $r_{22}=\bar{a}b\bar{e}\bar{f}$, $r_{23}=e\bar{f}$, $r_{24}=\bar{e}f$

Trajectoire position de l'aiguillage: (Fig.II.10.d)

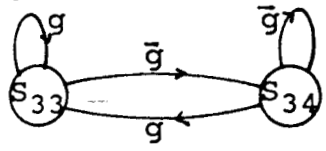
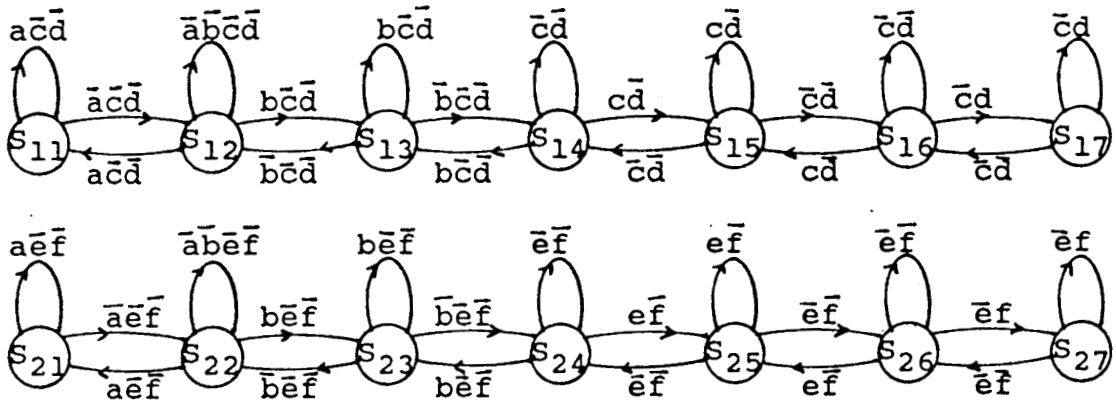


Figure II.10.e: Codage dans le cas où l'accès simultané au tronçon commun est possible.

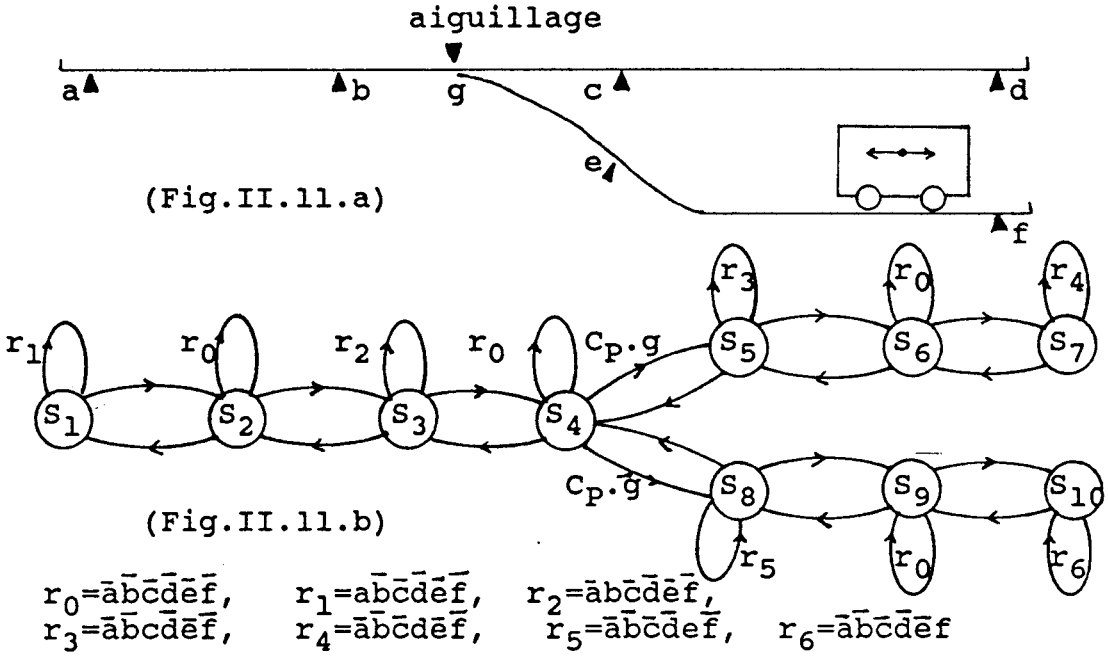


b) Modification de la trajectoire d'une grandeur par une autre:

Soit la PO de la figure II.11 représentant l'évolution d'un chariot sur une voie ferrée avec bifurcation commandée par un aiguillage. La position de ce dernier à donc une influence sur l'évolution du chariot à partir de S_4.

L'introduction d'une condition supplémentaire en fonction de l'état de l'automate modélisant le comportement de l'aiguillage permet de lever l'indétermination sur l'évolution du chariot. Cette condition peut être exprimée à partir de la position des capteurs repérant la grandeur modificatrice (aiguillage) si le codage de ses points singuliers est en bijection avec ses états. C'est le cas de l'exemple de la figure II.11.

Figure II.11: Modification de trajectoire.



$r_0=\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}\bar{f}$, $r_1=a\bar{b}\bar{c}\bar{d}\bar{e}\bar{f}$, $r_2=\bar{a}b\bar{c}\bar{d}\bar{e}\bar{f}$,
$r_3=\bar{a}\bar{b}cd\bar{e}\bar{f}$, $r_4=\bar{a}\bar{b}\bar{c}d\bar{e}\bar{f}$, $r_5=\bar{a}\bar{b}\bar{c}de\bar{f}$, $r_6=\bar{a}\bar{b}\bar{c}\bar{d}\bar{e}f$

La prise en compte de l'état du capteur g repérant la position de l'aiguillage peut aider à prévoir le compte rendu attendu pour un déplacement du chariot à partir de S_4 avec une vitesse positive.

2.7.Prise en compte de la durée des actions

Pour mieux caractériser le comportement dynamique de la PO on peut faire intervenir la durée des actions. Pour cela on suppose que la commande appliquée est maintenue pendant tout le temps d'évolution de la grandeur mesurée à l'intérieur d'un point singulier et que toute modification de commande n'intervient qu'au début d'un point singulier.

Cette hypothèse est cohérente avec le comportement normal de la partie commande: les ordres sont effectivement modifiés à l'issue d'un changement de compte rendu.

Action élémentaire: Définition

Une action élémentaire représente l'évolution d'une grandeur mesurée à l'intérieur d'un point singulier sous l'influence d'une commande maintenue.

Cette notion d'action élémentaire est donc associée aux arrêtes du graphe représentant l'évolution des grandeurs mesurées.

Chaque action élémentaire peut être caractérisée par:
- un compte rendu initial r_I (CR associé au PS);
- un compte rendu final r_F (CR associé au PS suivant);
- la commande appliquée Ci (non nulle).

Proposition

A un facteur d'incertitude près, la durée d'activation d'une action élémentaire (dans les mêmes conditions de parcours) est un invariant.

La partie opérative est en effet influencée dans sa vitesse d'éxécution, pour une commande donnée, par des perturbations non mesurées (charges, usure, inertie...). La durée d'une action n'est donc pas strictement constante.

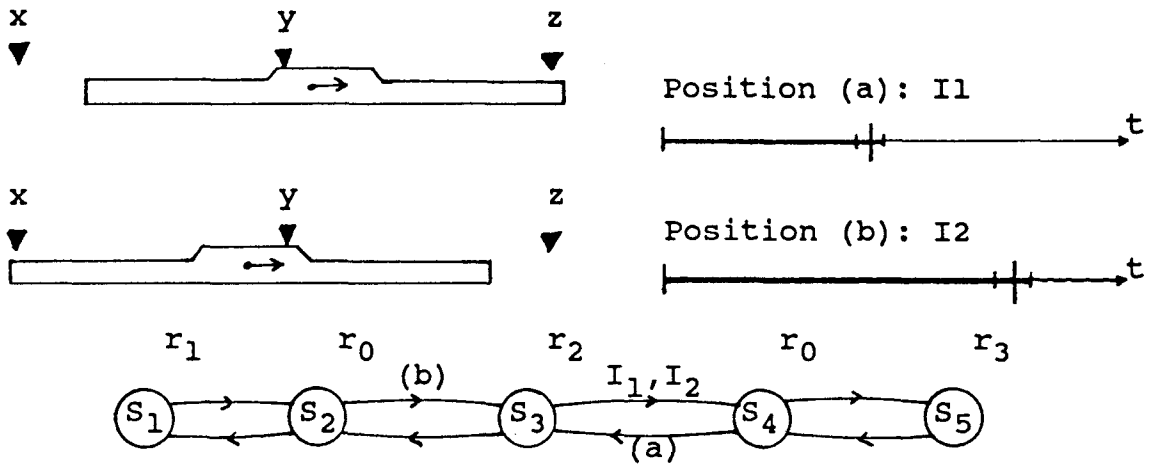
La date d'application de la commande (à vitesse non nulle) est une origine de la mesure des temps. Elle est donc fondamentale. Toute interruption (et à plus forte raison toute modification) de la commande appliquée empêche la mesure de la durée (problème d'inertie).

Pour tenir compte de la géométrie des capteurs et de la "came" nous distinguons deux intervalles de temps possibles selon qu'il y a ou non changement de sens d'évolution à l'entrée du point singulier.

Exemple

Figure II.12: Durée des actions avec ou sans changement de sens

Soit une came non ponctuelle se déplaçant devant 3 capteurs x, y, et z.



La position de la came n'étant pas la même à l'entrée du PS représenté ici par le compte rendu $r_2=\bar{x}y\bar{z}$, les temps mis dans les deux cas (a) et (b) pour traverser ce point singulier avec la même vitesse positive sont différents.

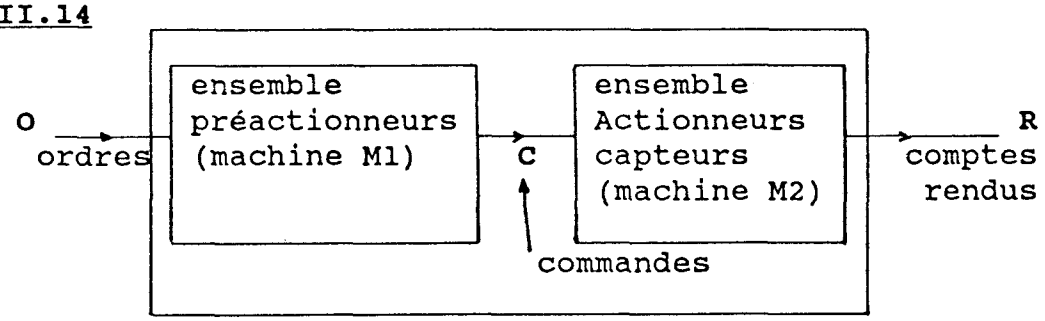
L'exploitation du modèle temporisé nécessite de déterminer pour chaque action élémentaire les deux intervalles de temps, avec et sans changement de sens. Ceci peut être réalisé pratiquement par une campagne de mesures des temps au cours d'une phase d'apprentissage.

Le choix de l'intervalle de temps adéquat à prendre en compte pour une évolution donnée, est fonction de la suite de commandes appliquées et de l'arc (qi,qj) empreinté précédemment.

2.8.Modélisation de la partie préactionneurs

Les commandes relatives à l'évolution des grandeurs mesurées sont générées par la partie "préactionneurs" en fonction des ordres appliqués (Fig.II.14).

Figure II.14



Les commandes n'étant pas directement observables, leur détermination nécessite d'avoir un modèle de comportement de ces préactionneurs en fonction des ordres qui leurs sont appliqués.

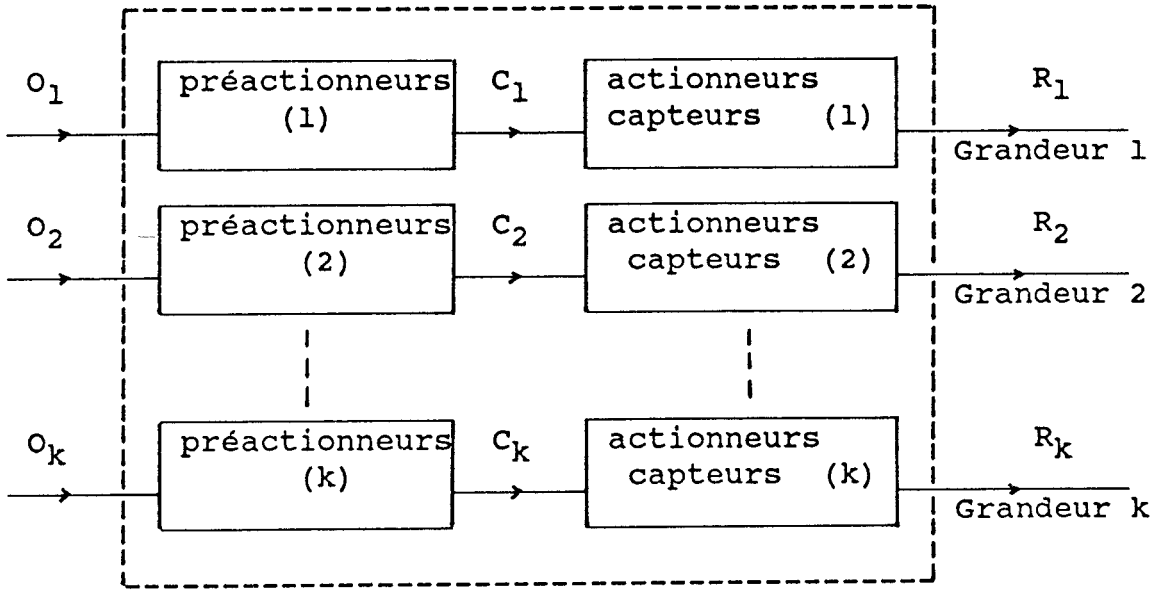
La décomposition en sous parties opératives opérée précédemment en fonction des grandeurs mesurées a fait ressortir l'association d'un alphabet de commande C_i à chaque grandeur i.

Nous conserverons cette décomposition pour la partie préactionneurs et nous tiendrons compte des interactions éventuelles (Fig.II.15). L'espace d'ordre O est donc formé par le produit cartésien:

$$O = \underset{i}{X}\ O_i$$

La partie préactionneurs constituée par l'association de contacteurs, distributeurs... de type monostable, bistable... peut être modélisée, d'une façon générale, par une machine séquentielle sur chaque couple d'alphabets O_i/C_i. D'un point de vue pratique, nous verrons dans le chapitre 3 différentes solutions pour la description de cette partie préactionneurs.

Figure II.15: Décomposition de la PO en sous processus.



3.TEST EN LIGNE ET DIAGNOSTIC DE LA PARTIE OPERATIVE

3.1. Différentes possibilités de test en ligne

La détection et la localisation des défaillances en ligne de la partie opérative peuvent être réalisées à différents niveaux (voir parag. I.5 sur les systèmes autotestables):
- Surveillance des domaines statiques (ordres, comptes rendus)
- Surveillance des domaines dynamiques,
- surveillance des domaines dynamiques avec prise en compte du temps ou durées des actions élémentaires.

Notre but dans cette étude étant la détection des erreurs générées par la PO nous supposerons que la partie commande est saine. Nous ne tiendrons donc pas compte d'éventuels ordres erronés.

Nous étudions différentes procédures de test en ligne basées sur la surveillance des domaines dynamiques d'évolution des grandeurs mesurées.

3.2. Analyse syntaxique des comptes rendus en dehors du contexte de la commande

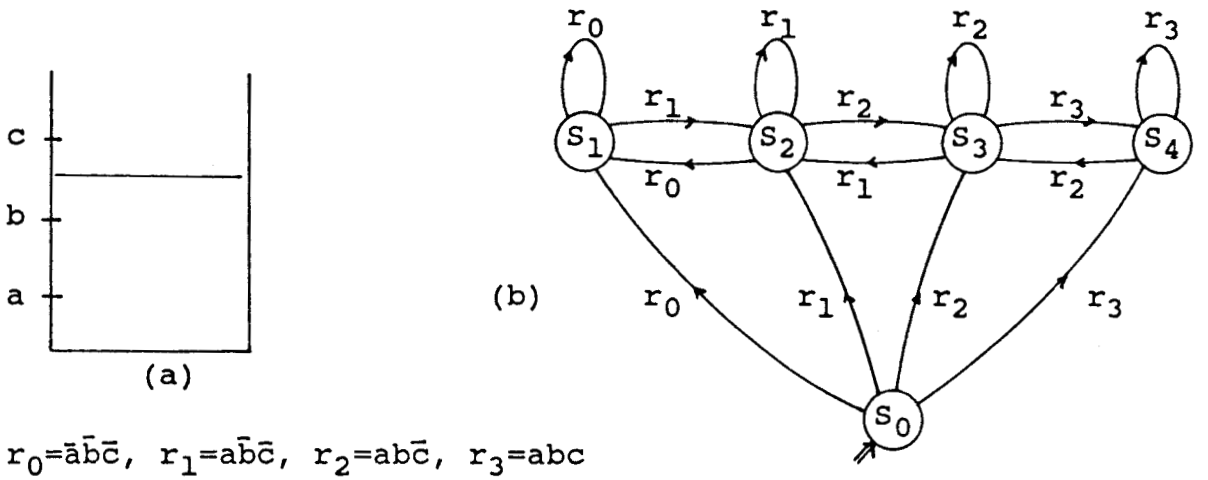
3.2.1.Construction du modèle de diagnostic

On considère le graphe associé à une grandeur mesurée représentant son évolution normale (fig.II.16.b) ou en d'autres termes l'ensemble des séquences de CR acceptables. Ce graphe peut être considéré comme un automate d'état fini si on ajoute un état final qf atteignable à la suite d'une défaillance (fig.II.16.c).

D'après la théorie des langages, toute phrase ou séquence de CR acceptée par l'automate est donc révélatrice d'une défaillance!

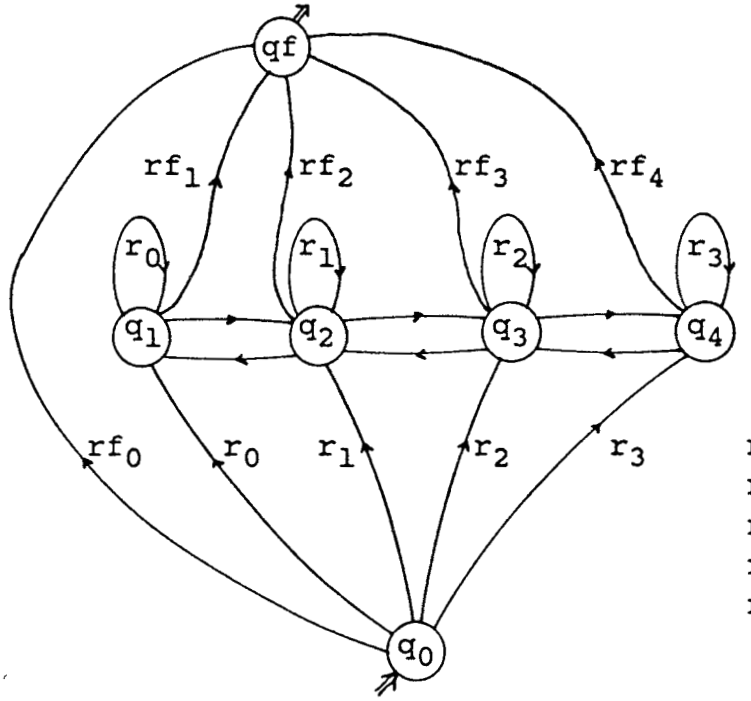
Les étiquettes associées aux arcs (qi,qf) menant vers l'état final sont formées par les comptes rendus non acceptables à partir de l'état qi y compris les combinaisons aberrantes des capteurs dénotées rf_0. Ces étiquettes tiennent compte éventuellement des partages de trajectoire et modification de trajectoire lorsqu'il y a interaction.

Figure II.16: Obtention de l'automate de diagnostic.



$r_0=\bar{a}\bar{b}\bar{c}$, $r_1=a\bar{b}\bar{c}$, $r_2=ab\bar{c}$, $r_3=abc$

Figure II.16.c



rf_0= {$\bar{a}b\bar{c}$, $\bar{a}bc$, $\bar{a}\bar{b}c$, $a\bar{b}c$}
rf_1= {rf_0, r_2, r_3}
rf_2= {rf_0, r_3}
rf_3= {rf_0, r_0}
rf_4= {rf_0, r_0, r_1}

Si l'automate ainsi obtenu n'est pas déterministe une transformation est nécessaire. L'analyse syntaxique d'une phrase par un automate non déterministe est en effet plus complexe.

3.2.2.Obtention de l'automate déterministe

Pour rendre un automate déterministe, on dispose d'un important théorème [24]:

théorème

Soit un language L, tel que L=L(**A**) où **A** est un automate fini non déterministe. Il existe alors **A'**, automate fini déterministe tel que L=L(**A'**) acceptant donc exactement le même langage.

L'automate **A'** = (X, Q', δ', q_0, Q'f) est défini de la façon suivante:

a) Q' = P(Q) ensemble des parties de Q
b) $q'_0 = q_0$
c) Q'f est l'ensemble des parties S de Q tel que $S \cap Q' = \emptyset$
d) δ' est défini pour $R \in P(Q)$

$$\delta'(R,x) = \bigcup_{q \in R} \delta(q,x)$$

L'algorithme pratique de détermination d'un automate fini consiste simplement à engendrer la fonction de transition δ' à partir de la définition de **A'**.

Algorithme pour rendre un automate déterministe:

On note:
q'_j le j^{eme} élément de Q'
q_i le i^{eme} élément de Q
$\delta_{i,xp}$ c δ' l'ensemble des arcs de δ' d'origine q_i et de même étiquette xp
Soit U un ensemble de partie de Q

a) On initialise la procédure en introduisant q_0 dans **U**.
b1) Prendre un élément $q'_j \in U$, l'introduire dans Q' et l'éliminer de **U**.
b2) Pour tout élément q_i représenté dans q'_j et pour tout ensemble d'arcs $\delta_{i,xp}$ de même étiquette xp faire:
c1) créer une partie de Q, notée q'_n, regroupant toutes les extrémités q_k des arcs tels que $(q_i,xp,q_k) \in \delta_{i,xp}$.

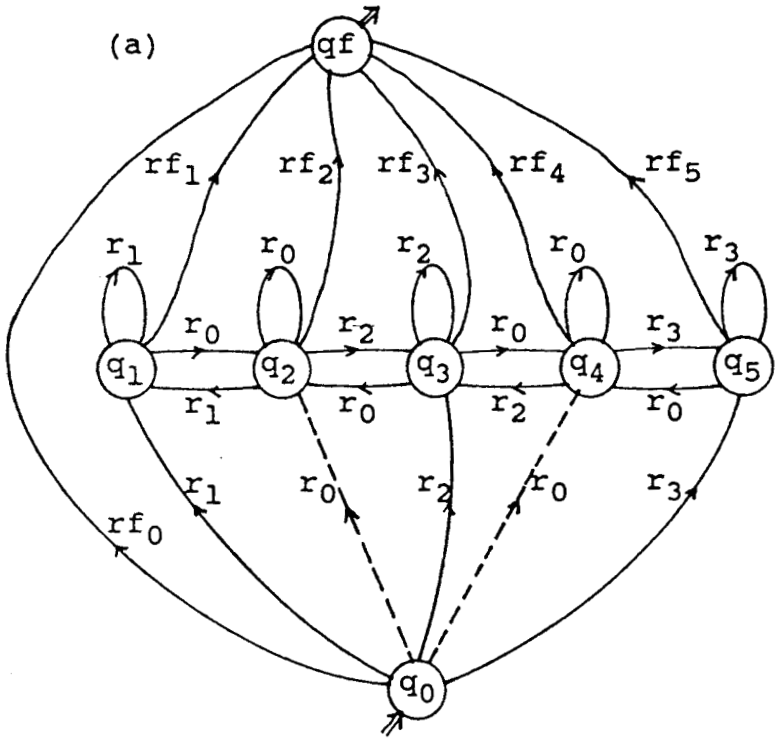
c2) introduire l'arc (q'_j, xp, q'_n) dans δ'.
c3) si q'_n n'est ni dans U ni dans Q' alors introduire q'_n dans U.
d) Reprendre en b1 jusqu'à ce que U soit vide.

A titre d'exemple la figure II.17.b donne l'automate déterministe correspondant à celui de la figure II.17.a.

L'indéterminisme a l'inconvéniant de diminuer l'efficacité du test. Dans l'exemple, les séquences r_2, r_0, r_1 et r_2, r_0, r_3 à partir de q3 ne peuvent plus mener vers l'état final.

Lorsqu'il y a partage de trajectoire, l'élaboration du modèle de diagnostic doit tenir compte de cette contrainte supplémentaire.

Figure II.17: Obtention de l'automate déterministe.

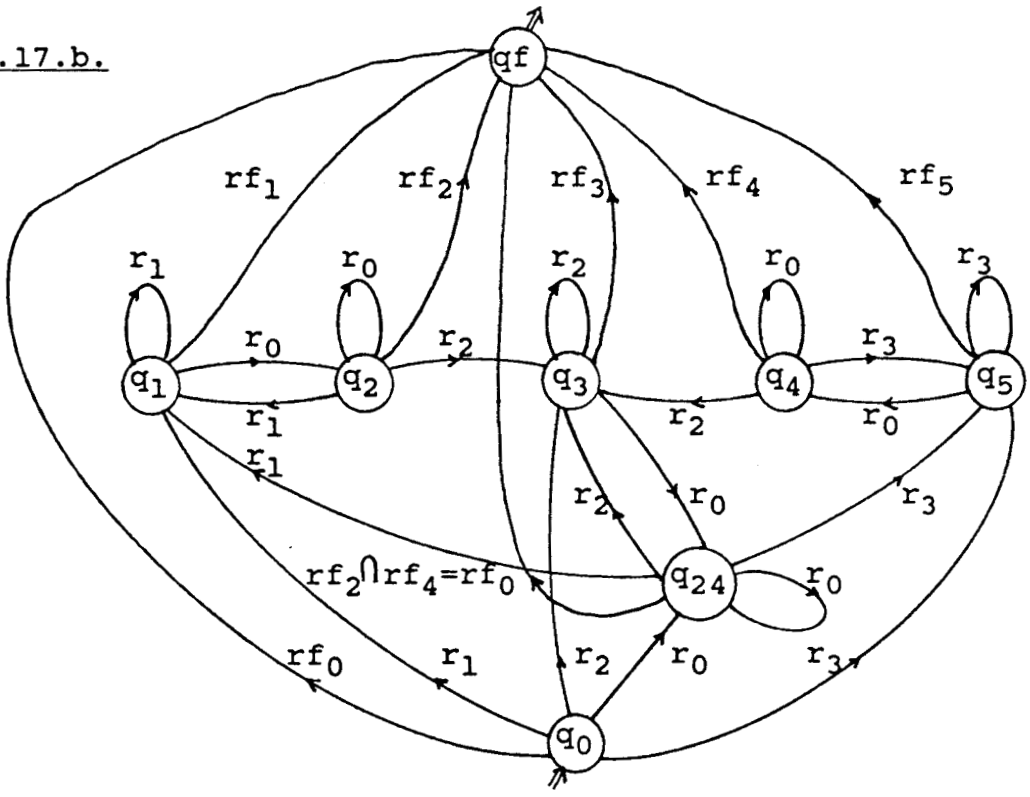


$r_0=\bar{a}\bar{b}\bar{c}$, $r_1=a\bar{b}\bar{c}$,
$r_2=\bar{a}b\bar{c}$, $r_3=\bar{a}\bar{b}c$

$rf_0=\{ab\bar{c},\ a\bar{b}c,\ \bar{a}bc,\ abc\}$

$rf_1=\{\ rf_0\ ,\ r_2\ ,\ r_3\}$
$rf_2=\{\ rf_0\ ,\ r_3\}$
$rf_3=\{\ rf_0\ ,\ r_1\ ,\ r_3\}$
$rf_4=\{\ rf_0\ ,\ r_1\}$
$rf_5=\{\ rf_0\ ,\ r_1\ ,\ r_2\}$

Figure II.17.b.



3.2.3. Défaillances détectables par le modèle:

Dans l'automate de diagnostic, l'arrivée dans un état de Qf correspond à une défaillance détectée. La mission de surveillance est alors terminée. Il faut faire intervenir le mécanisme de réaction-configuration et/ou alerter le réparateur.

Pour identifier l'origine probable de la défaillance, il faut conserver l'information sur l'état de la grandeur mesurée (compte rendu présent avant défaillance) et le dernier CR révélateur de la défaillance. Ceci peut être réalisé au niveau de l'automate de diagnostic en évitant le fusionnement des états finaux. L'état final atteint peut alors aider à localiser la panne.

En fait, pour qu'une défaillance soit détectable par le modèle il faut qu'elle engendre:

- soit un compte rendu aberrant, la détection est alors immédiate,
- soit une substitution de CR dans une séquence telle que celle-ci devienne acceptable par l'automate, menant donc vers un état final.

Si aucune de ces deux conditions n'est satisfaite en présence d'une panne, celle-ci est alors masquée. Mais l'évolution future de la PO peut mettre en évidence la défaillance. Ce temps de latence peut être plus ou moins important.

Les types de pannes détectables par le modèle sont essentiellement les collages de capteurs. L'avantage par rapport à la surveillance du seul domaine statique d'évolution (représenté ici par l'arc q_0, rf_0, qf) est qu'ici, tenant compte des séquences de CR possibles, tout collage de capteur est détectable.

Certains types de défaillances ne sont pas détectables par ce genre de modèle. Ce sont en particulier les substitutions de commandes pouvant se traduire par une variation notable de la vitesse d'évolution, changement de sens, blocage, départ intempestif... Ceci était en effet prévisible puisqu'il n'a pas été tenu compte dans ce modèle des commandes appliquées.

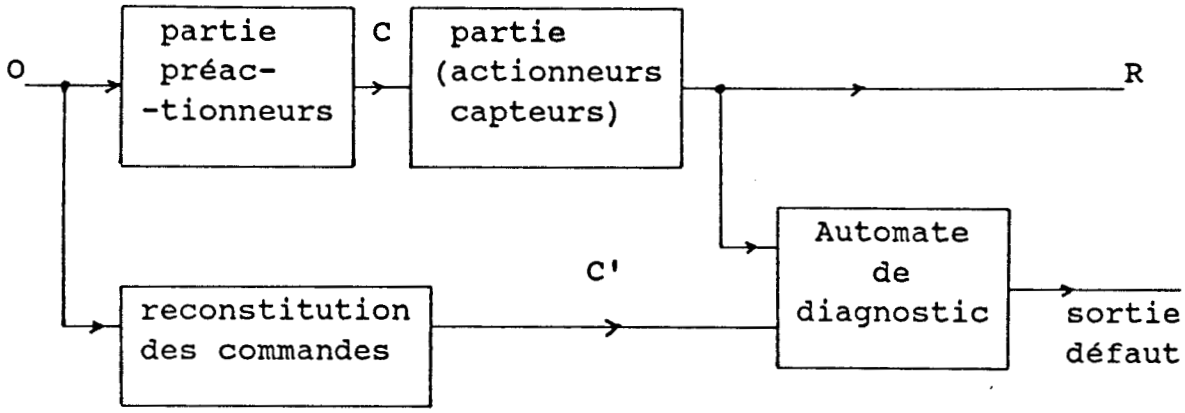
3.3.Analyse syntaxique des comptes rendus avec prise en compte des commandes appliquées

Le modèle de diagnostic tenant compte des commandes appliquées est obtenu suivant le même principe que précédemment, mais à partir du modèle de comportement de la PO défini sur l'alphabet RxC (fig.II.18).

La figure II.19.a représente l'automate de diagnostic de la PO décrite à la figure II.2. La figure II.19.b montre l'indétermination résolue par la prise en compte du sens d'évolution.

Il faut remarquer ici qu'il n'a pas été tenu compte d'éventuelles commandes incompatibles avec l'état présent de la partie opérative, en particulier aux extrémités de la trajectoire. Dans l'automate de la fig.II.19.a, les arcs $(q_4, r_{0-3}xCp, --)$ et $(q_1, r_{0-3}xC_N, --)$ ne mènent pas vers un état final.

Figure II.18: surveillance des séquences commande/compte rendu.

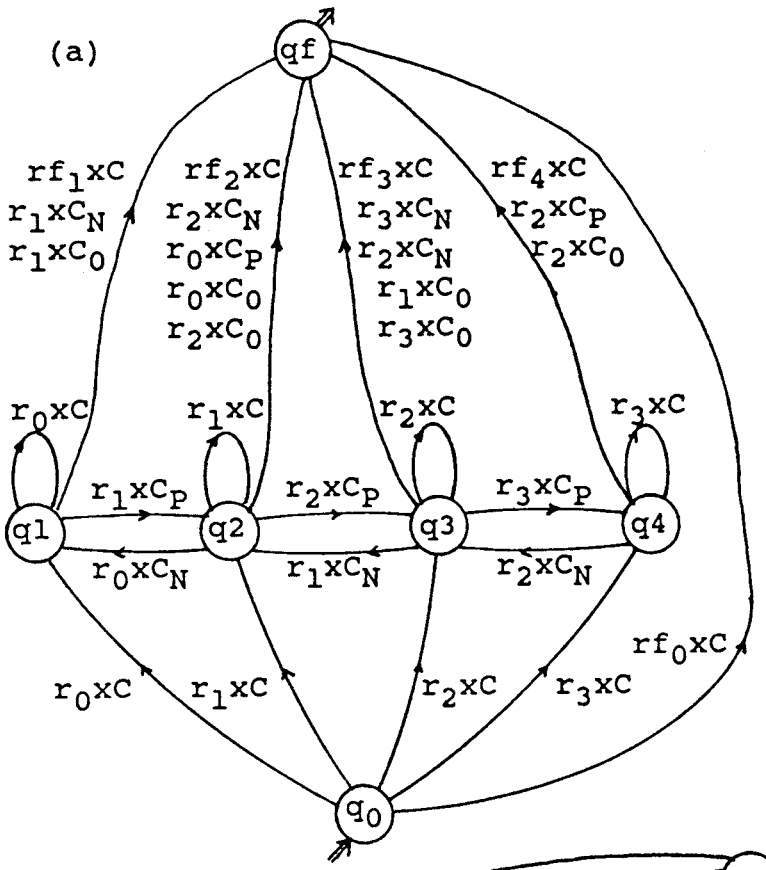


Le problème d'indétermination évoqué précédemment est résolu ici, en dehors de la phase de localisation, par la connaissance du sens d'évolution. Ce modèle inclu donc le test statique et dynamique de l'évolution des CR.

La prise en compte de la commande permet de mettre en évidence en plus des pannes détectables par le modèle précédent, essentiellement les substitutions de commandes se traduisant par un changement de sens d'évolution ou des évolutions intempestives (ou non commandées).

On ne peut pas détecter les blocages ou variations de vitesse avec cet automate.

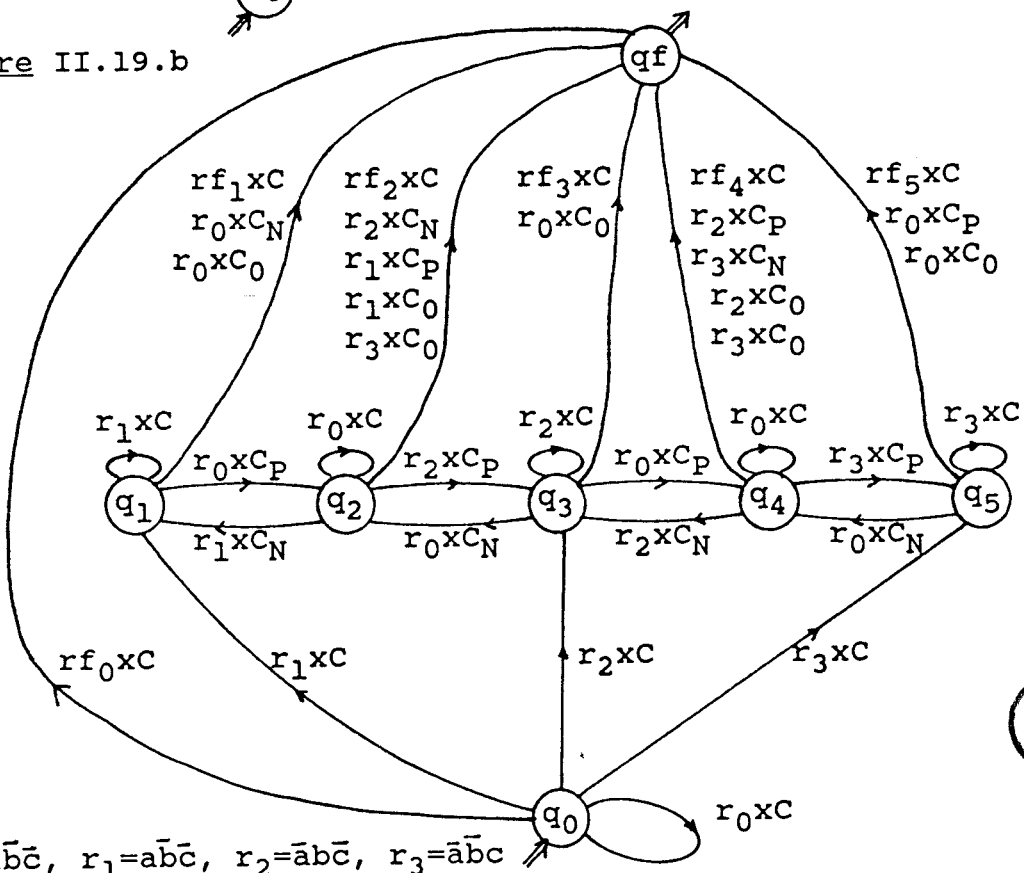
Figure II.19: Automate de diagnostic sur l'alphabet RxC.



$r_0 = \bar{a}\bar{b}\bar{c}$, $r_1 = a\bar{b}\bar{c}$
$r_2 = ab\bar{c}$, $r_3 = abc$

$rf_0 = \{\bar{a}\bar{b}\bar{c}, \bar{a}bc, \bar{a}\bar{b}c, a\bar{b}c\}$
$rf_1 = \{rf_0, r_2, r_3\}$
$rf_2 = \{rf_0, r_3\}$
$rf_3 = \{rf_0, r_0\}$
$rf_4 = \{rf_0, r_0, r_1\}$

Figure II.19.b



$r_0 = \bar{a}\bar{b}\bar{c}$, $r_1 = a\bar{b}\bar{c}$, $r_2 = \bar{a}b\bar{c}$, $r_3 = \bar{a}\bar{b}c$
$rf_0 = \{ab\bar{c}, a\bar{b}c, \bar{a}bc, abc\}$; $rf_1 = \{rf_0, r_2, r_3\}$; $rf_2 = \{rf_0, r_3\}$
$rf_3 = \{rf_0, r_1, r_3\}$; $rf_4 = \{rf_0, r_1\}$; $rf_5 = \{rf_0, r_1, r_2\}$

3.4.Analyse syntaxique des comptes rendus avec prise en compte de la durée de maintien dans un état

L'introduction de temporisations dites de "chien de garde" dans le modèle revient à considérer le temps de maintien dans un état. Les grammaires régulières, utilisées jusqu'ici, interdisent cette pratique. Nous devons donc nous orienter vers d'autres types de modèles.

Il existe de nombreuses grammaires et notamment celles qui sont représentables par des automates stochastiques ou automates flous [24] qui permettent de faire une analyse syntaxique tolérante. La réponse à la question de savoir si une phrase appartient au langage n'est plus alors dichotomique.

De telles méthodes (à décision statistique) présentent le défaut de rejeter toute évolution normale mais peut fréquente. De plus, elles ne peuvent tenir compte des limites temporelles attendues.

Les grammaires programmables [59] semblent mieux adaptées à la formalisation du problème qui nous préoccupe. Dans ce cas les règles de production sont de la forme:

$$A \rightarrow xB \mid xC \mid x \quad \text{avec } x \in V_T \quad \text{et } A,B,C \in V_N$$

Le choix entre les différentes réécritures de A est fait par une règle complémentaire généralement basée sur l'algèbre des prédicats. Nous rappelons ci-après la définition d'une telle grammaire à partir de la notion d'automate pondéré que nous utilisons ensuite pour le test.

3.4.1. Automate pondéré: définition

Un automate pondéré **Ap** est défini par un sextuplet

Ap = (X, Q, δp, q_0, Qf, V) où

X est l'alphabet terminal,

Q est l'ensemble des états,

Qf C Q est l'ensemble des états finals,

$q_0 \in Q$ est l'état initial,

δp: Qx(X,V) -> Q est la fonction de transition,

δ:QxX -> Q constitue la fonction de transition de l'automate de référence associé.

V est un ensemble de prédicats.

A chaque état $q_i \in Q$ on associe un nombre entier appelé poids et noté p_i.

A chaque arc $\alpha_{ij}=(q_i,x_{ij},q_j) \in \delta$ est associé un intervalle de valeurs possibles pour p_i noté I_{ij} et une application:

$$V(\alpha_{ij}) : p_i \rightarrow v(\alpha_{ij}) \in \{0,1\} = B$$

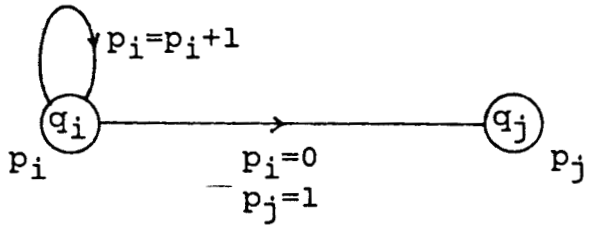
appelée prédicat tel que:

$$v(\alpha_{ij}) = \begin{cases} 1 \text{ si } p_i \in I_{ij} \\ 0 \text{ si } p_i \notin I_{ij} \end{cases}$$

Un arc α_{ij} ne peut alors être empreinté que si le prédicat associé est vrai: $v(\alpha_{ij}) = 1$.

3.4.2. Règle de calcul du poids associé à un état

Toute évolution selon un arc $\alpha_{ij} = (q_i,x_{ij},q_j)$ modifie le poids affecté à q_i , q_j de la façon suivante:



A l'initialisation, les poids sont fixés comme suit:

$$p_i = 0 \quad \forall\, i \neq 0 \; ; \qquad p_0 = 1$$

Nous obtenons de cette façon un automate évolutif au fur et à mesure du déroulement de la séquence.

3.4.3.Interprétation de la pondération:

Le calcul de p_i correspond au nombre d'itérations de r_i R si R=X que l'on peut observer si la partie opérative est dans l'état q_i. A la période d'échantillonnage près, p_i représente donc la durée de maintien dans l'état q_i ou le temps écoulé entre deux variations consécutives de compte rendu. Les valeurs min et max des intervalles I_{ij} utilisés pour définir les prédicats permettent de tenir compte de l'influence de perturbations.

3.4.4.Principe du test et détermination des intervalles Iij

Le principe du test reste inchangé; une erreur est révélée par l'existence d'une séquence menant vers un état final. La différence est que l'automate utilisé ici est modifié en fonction des prédicats associés aux transitions. L'automate pondéré permet de compter le nombre d'itérations de chaque CR contenu dans la séquence à reconnaître.

Les intervalles de valeurs associés aux arcs sont déterminés de la manière suivante. On peut en distinguer 6 types (fig.II.20).

L'évolution normale de la partie opérative fixe les valeurs des intervalles I_{ij}, I_{ik} pour chaque sens d'évolution à l'intérieur des points singuliers en fonction de la "longueur" de ceux-ci et de la vitesse. Tous les autres intervalles sont déduits des I_{ij}, I_{ik} de la manière suivante (fig.II.20.b):

- Les intervalles notés IF_{ij}, IF_{ik} sont les complémentaires de I_{ij}, I_{ik}; soit $IF_{ij} = \mathbf{N} - I_{ij}$.

- L'intervalle I_{ii} représentant le temps maximun de maintien dans l'état q_i, est donné par $I_{ii} = [\ 0\ ,\ Max(max(I_{ij}))]$.

- L'intervalle IF_{ii} est le complémentaire de I_{ii}. Soit $IF_{ii} = \mathbf{N} - I_{ii}$.

- Les intervalles notés Io et If sont égaux à **N**. Les arcs auxquels est associé un de ces intervalles ont donc leur prédicat correspondant toujours vrai.

Parmi les étiquettes d'arcs menant vers l'état final on trouve:

- (r_{ii}, IF_{ii}) traduisant la persistance du compte rendu r_{ii} en dehors du temps maximum alloué.

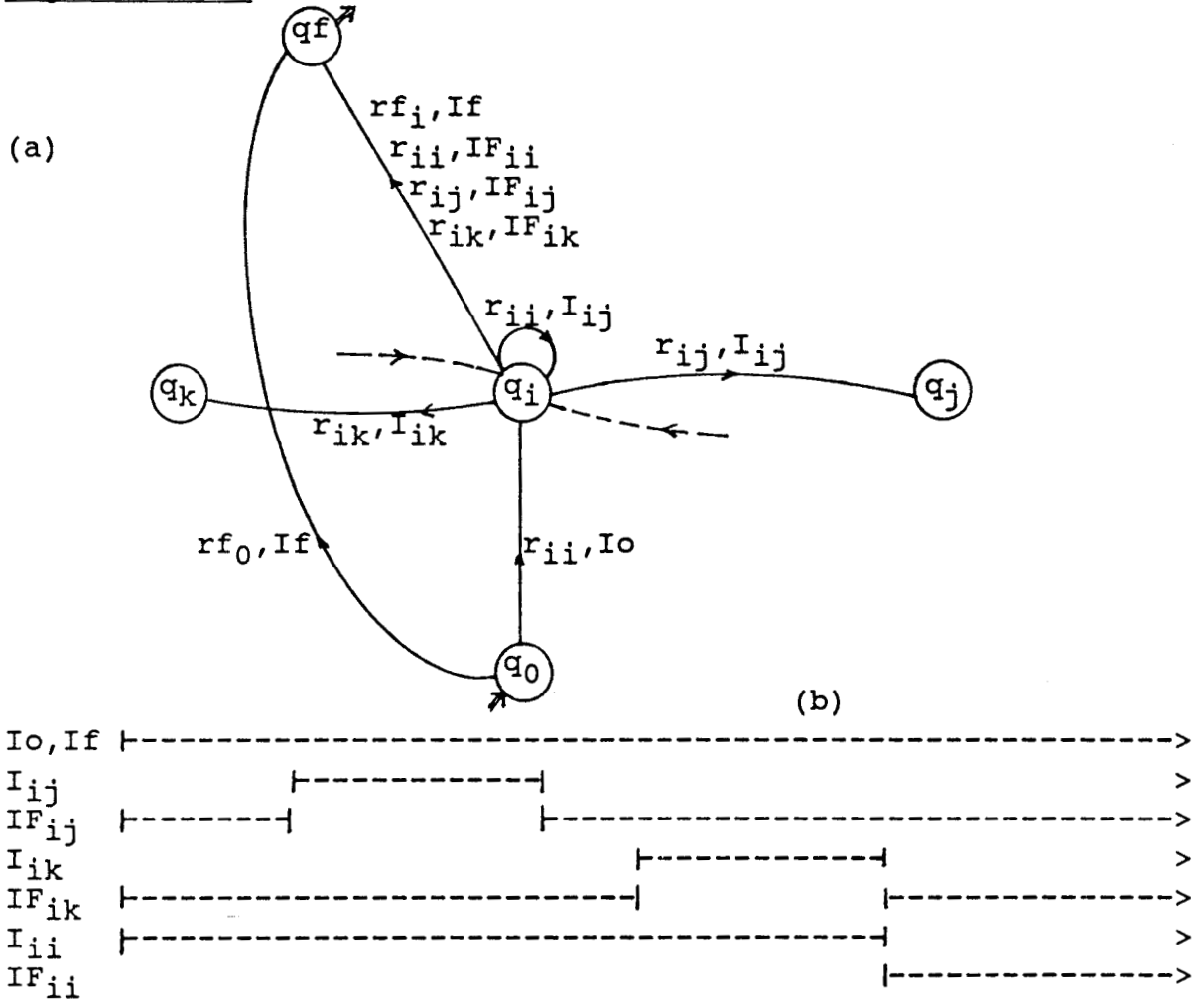
- (r_{ij}, IF_{ij}) et (r_{ik}, IF_{ik}) traduisant la présence de r_{ij}, r_{ik} en dehors de l'intervalle autorisé.

- (rf_i, If) sont les arcs ayant des étiquettes de CR non atteignables à partir de ce point singulier ou des CR aberrants.

- (rf_0, If) traduit la présence d'un compte rendu aberrant.

Tout changement de compte rendu en dehors des intervalles de temps alloués mène ainsi vers l'état final, mettant en évidence la défaillance.

Figure II.20: Détermination des intervalles.



L'affectation aux arcs (q_0, r_{ii}, q_i) de l'intervalle Io a pour but d'accélérer la phase de localisation. Mais se pose alors le problème de synchronisation du modèle sur le processus réel. En phase d'initialisation tous les intervalles doivent être modifiés afin de tenir compte des décalages éventuels et éviter la signalisation de fausses pannes.

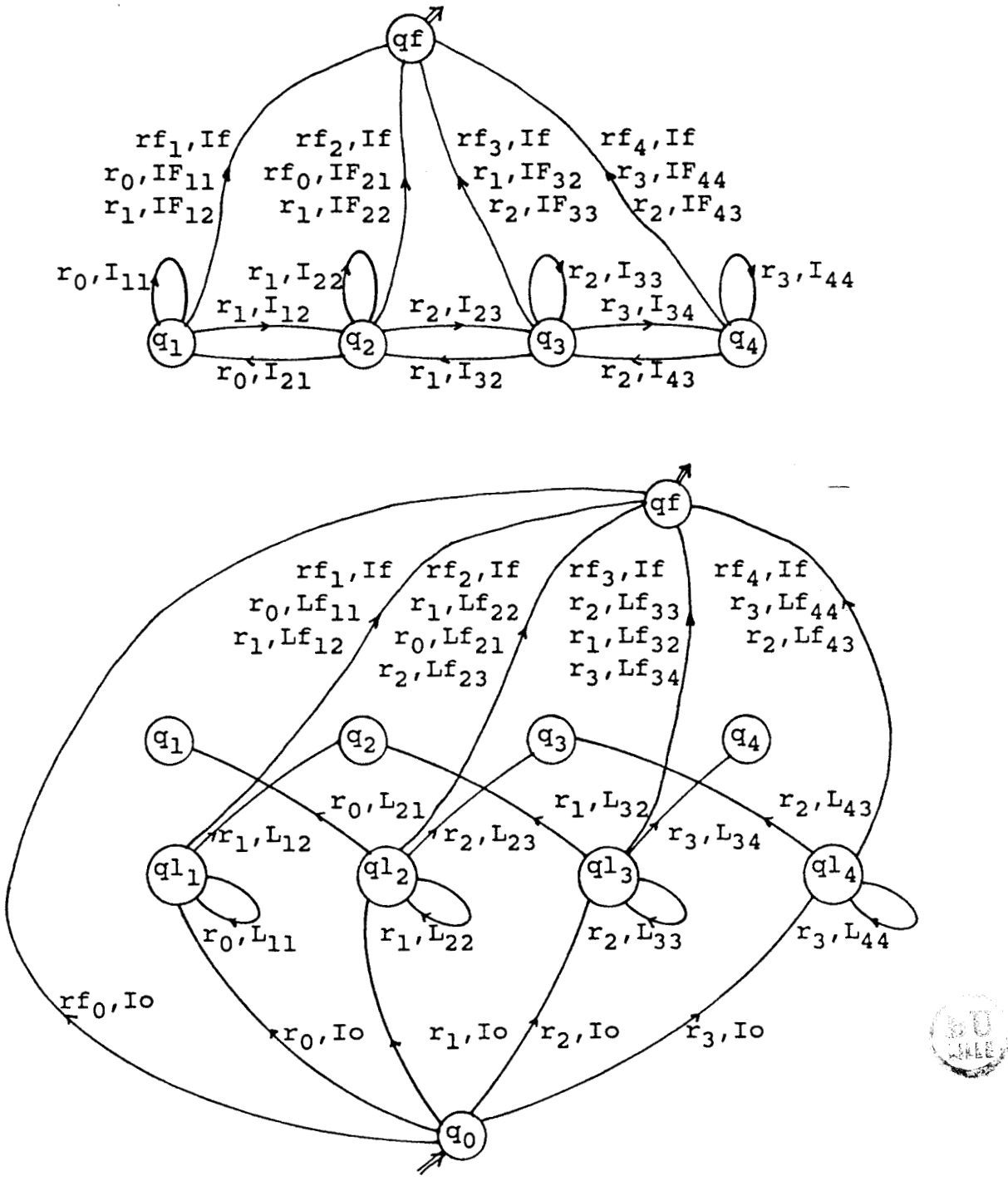
La solution adoptée consiste à créer, pour chaque état q_i, un état homologue ql_i afin de réaliser cette synchronisation (fig.II.21). Les intervalles I_{ij}, I_{ik} sont alors élargis pour donner L_{ij}, L_{ik} tels que:

$$L_{ij} = [\ 0\ ,\ \max(I_{ij})]$$

Les autres intervalles L et LF sont déduits des L_{ij} de la même manière que précédemment.

L'élargissement de ces intervalles a pour conséquence de diminuer l'efficacité du test en phase de localisation. Lorsqu'il y a indétermination, une adaptation des intervalles de valeurs est nécessaire: ceux-ci doivent être élargis afin de tenir compte de l'incertitude sur le sens d'évolution ou le point singulier atteint.

Figure II.21: synchronisation en phase d'initialisation.



3.4.5. Pannes détectables par le modèle

Nous avons supposé dans le présent modèle qui fait intervenir la persistance des comptes rendus, qu'à une marge d'incertitude près, le temps de maintien dans un état est un invariant. Ceci ne peut être vrai que si la partie opérative a un cycle de travail répétitif.

N'ayant pas tenu compte des temps d'arrêt aléatoires (pendant une marche manuelle par exemple ou un changement de cycle de fonctionnement), ce modèle n'est valable que pendant les cycles de travail strictement répétitifs.

De plus, lorsqu'un point singulier peut être parcouru avec des vitesses différentes, ou que la forme de la "came" est telle que les temps avec et sans changement de sens sont nettement différents, les intervalles d'incertitude deviennent trops larges. L'efficacité du test en est réduite d'autant.

Cet automate de diagnostic permet donc de révéler, en plus des pannes détectables par le modèle du paragraphe 3.2, les variations notables de vitesse ainsi que les blocages. Les départs intempestifs et les changements de sens ne sont détectables que s'ils sont accompagnés d'une variation des temps d'éxécution telle que ceux-ci soient en dehors des intervalles de fonctionnement normal.

L'efficacité du test peut être améliorée par une réduction des intervalles d'incertitude en prenant en compte les commandes appliquées.

3.4.6. Remarque sur le choix des intervalles de tolérance

Le choix des intervalles de tolérance constitue un dilemme: l'élagissement des intervalles risque de masquer certains types de pannes. Si les intervalles sont trop serrés, la présence de perturbations importantes mais acceptables, ou l'influence de transitoires non éteintes entraine une détection erronée.

Le choix de la dimension de l'intervalle associé à un arc ou action élémentaire dépend de la distribution des durées et de la confiance accordée. Si cette distribution suit une loi normale l'intervalle peut s'exprimer par:

$I = Im \pm \alpha\sigma$ avec σ l'écart type et α le facteur de confiance.

Pour un coefficient de confiance égal à 2 le nombre de fausses pannes prévisible suite à l'arrivée de comptes rendus hors délais est de 4,6% .

Le suivi de l'évolution ou dérive des durées ou poids associés à chaque PS en cours d'exploitation permet de mettre en évidence certains phénomènes de vieillissement affectant en particulier la mécanique.

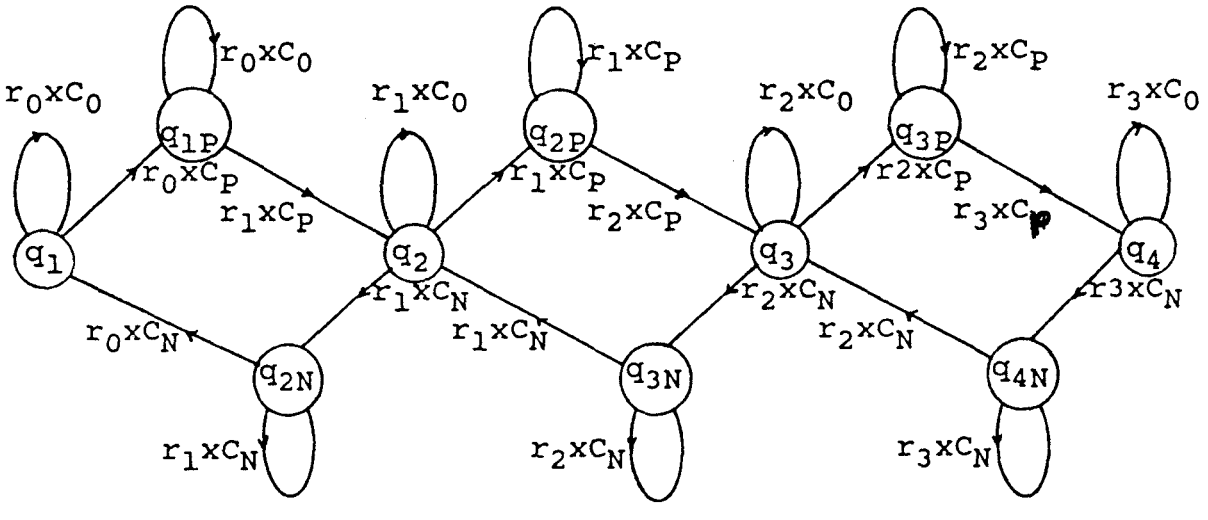
3.5.Analyse syntaxique des comptes rendus avec prise en compte de la durée et des commandes appliquées

3.5.1.Elaboration du modèle de diagnostic

On utilise ici l'automate pondéré défini précédemment avec les mêmes types de prédicats mais les étiquettes sont maintenant définies sur l'alphabet RxC.

Pour chaque état q_i, représentant un point singulier, et pour chaque sens d'évolution possible, on crée un état noté q_{iP} ou q_{iN} destiné à mettre en évidence la date d'application de la commande à vitesse non nulle (fig.II.22).

Figure II.22: Prise en compte des commandes appliquées dans le modèle pondéré. $C = \{C_0, C_P, C_N\}$



Dans les états q_i, le temps de maintien n'est pas limité. Ces états correspondent à la présence d'une commande nulle. Les intervalles associés aux arcs (q_i, r_{ii}, q_i) sont donc égaux à **N**.

Le temps de maintien dans les états q_{iP}, q_{iN} où il y a présence d'une commande non nulle est limité.

En phase de localisation, nous reprenons la solution adoptée précédemment pour la synchronisation du modèle avec le processus.

L'hypothèse de commande maintenue permet de définir les intervalles de temps associés aux actions élémentaires. Pour éviter de sortir en défaut lorsque cette hypothèse n'est pas respectée, nous créons un état final supplémentaire qf' destiné à mettre en évidence les arrêts en cours du parcours d'un point singulier. En effet de tels arrêts demandent une resynchronisation du modèle.

3.5.2.Calcul des intervalles de valeurs

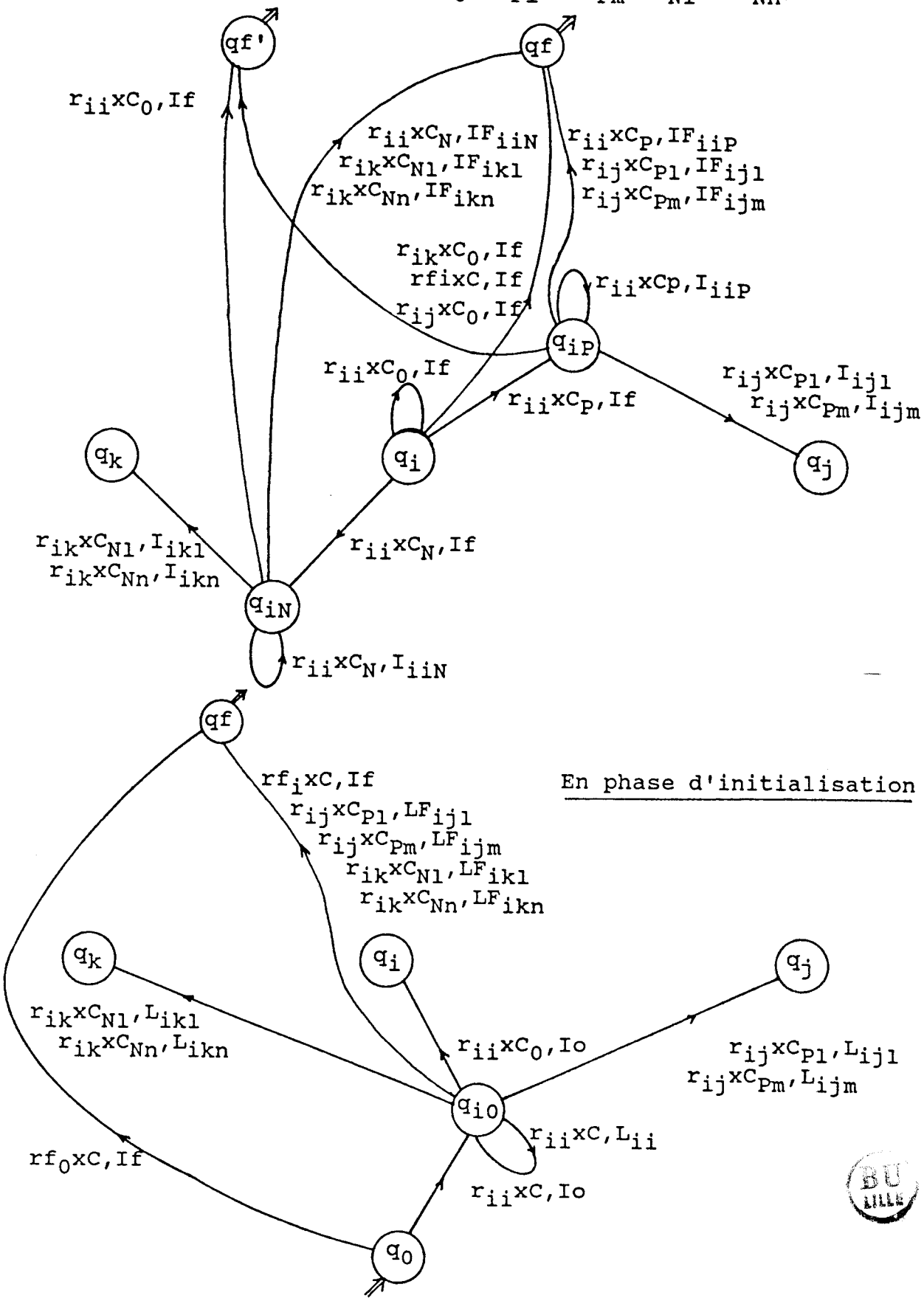
Les intervalles de base sont ici associés aux actions élémentaires représentant l'évolution suivant les arcs $(q_{iP}, r_{ij}xC_P, q_j)$ et $(q_{iN}, r_{ij}xC_N, q_j)$. Ces intervalles sont donc fonction (en plus) du module de la vitesse. (fig.II.23)

- Les intervalles I_{iiP} et I_{iiN}, représentant les temps de maintien dans les états q_{iP} et q_{iN} respectivement, sont tels que: $I_{ii} = [Min(min(I_{ij})) , Max(max(I_{ij}))]$.

- Les intervalles If_{ij} sont les intervalles complémentaires des I_{ij}.

A l'initialisation, les intervalles de bases I_{ij} sont élargis comme précédemment pour donner les intervalles L_{ij} à partir desquels sont calculés de la même façon les intervalles complémentaires $LF_{ij} = N - L_{ij}$. Dans cette phase de localisation l'efficacité du test est donc diminuée.

Figure II.23: Automate pondéré avec prise en compte des commandes. $C=\{C_0, C_{P1}\cdots C_{Pm}, C_{N1}\cdots C_{Nn}\}$



3.5.3.Avantages du modèle

La prise en compte des commandes appliquées dans l'automate pondéré permet une efficacité accrue du test. Tous les types de défaillance cités précédemment sont détectables par ce modèle qui intègre le maximum d'information sur la partie opérative à surveiller.

Le temps de latence a été fortement réduit par une décomposition des intervalles en fonction des modules des commandes possibles.

La mise en évidence de la date d'application de la commande à vitesse non nulle a permis de s'affranchir des cycles de travail strictement répétitifs. On évite ainsi la signalisation de fausses pannes en cas de changement de cycle ou mode de marche.

Il reste à prendre en compte le problème cité au paragraphe 2.7 relatif à la forme de la "came" et des capteurs. Pour cela il faut que le modèle puisse se "souvenir" des arcs empreintés précédemment. Ceci n'est pas possible avec le modèle d'automate pondéré utilisé jusqu'ici.

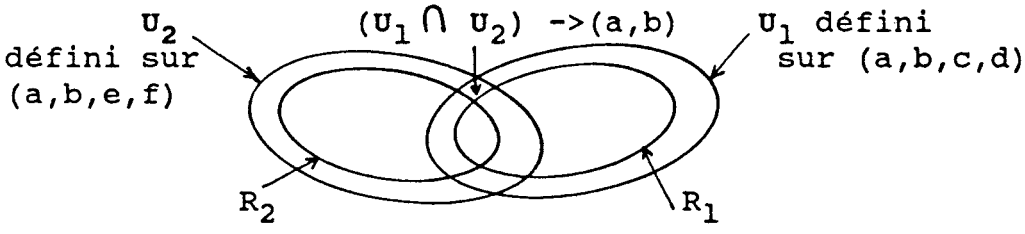
3.6. Etudes des cas où il y a interaction entre trajectoires

Nous avons étudié différents modèles de test en ligne en se basant sur le fait que la partie opérative est décomposable en sous processus indépendants. Nous analysons dans ce paragraphe, les modifications à apporter aux modèles de test dans le cas où il existe des interactions entre trajectoires des grandeurs mesurées.

3.6.1.Cas du partage de trajectoire

L'existence d'une portion de trajectoire commune à plusieurs grandeurs se manifeste au niveau des domaines de définition des alphabets de compte rendu, par une intersection non vide entre ces domaines à cause des capteurs partagés (fig.II.24).

Figure II.24: Domaines de définition.



Dans l'exemple de la figure II.3 où deux chariots se partagent un même tronçon avec exclusion mutuelle, nous distinguons quatres domaines de définition:

- U_1 = (a,b,c,d); $U'_1 = U_1 - (U_1 \cap U_2)$ = (c,d)
- U_2 = (a,b,e,f); $U'_2 = U_2 - (U_1 \cap U_2)$ = (e,f)

Les domaines U'_1 et U'_2 sont propres à chaque trajectoire. Les domaines U_1 et U_2 représentent le tronçon commun vu par les deux grandeurs.

Les étiquettes de compte rendu associées aux arcs de chaque automate sont définies sur deux domaines: avec U' dans le tronçon propre et avec U dans le tronçon partagé (fig.II.25).

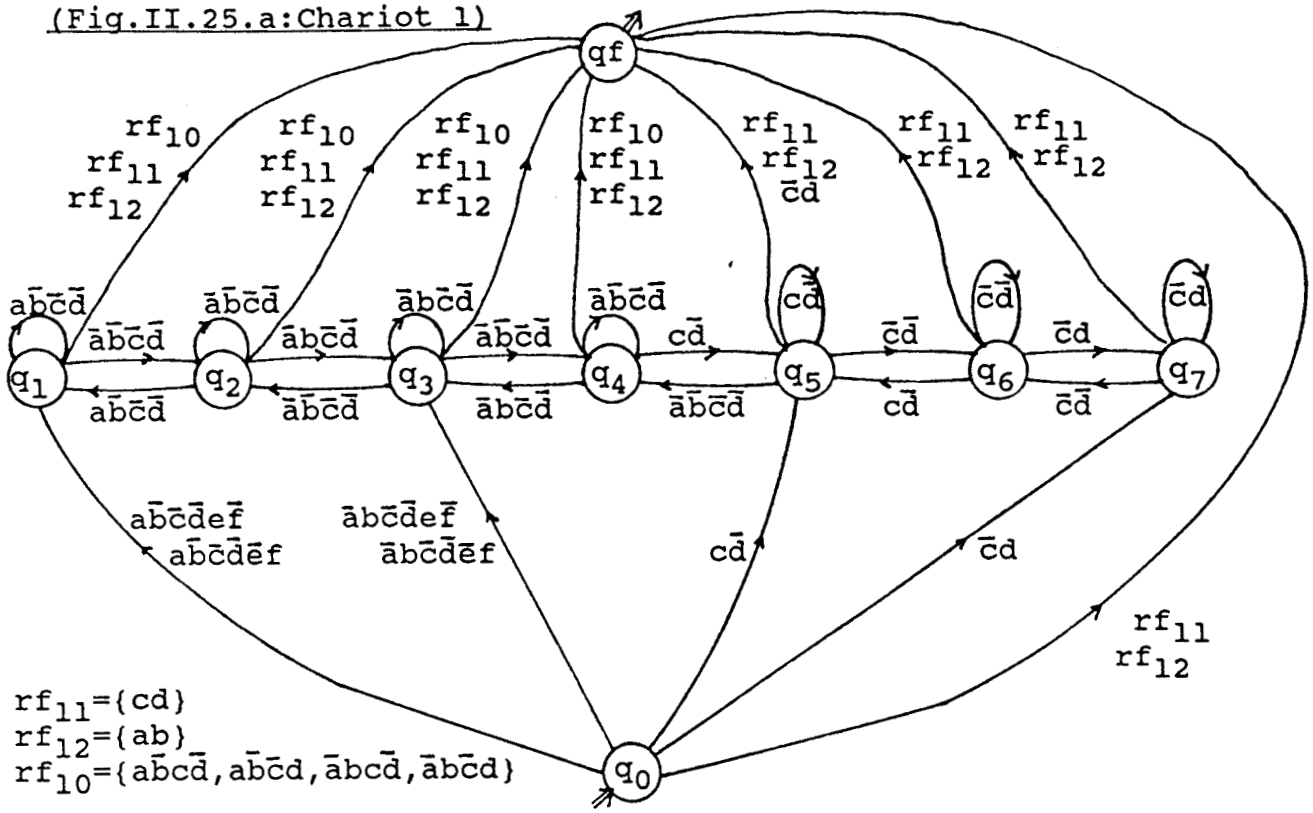
Les étiquettes de compte rendu associées aux arcs menant vers l'état final qf sont modifiées en conséquence. On constate alors que les capteurs du tronçon commun sont surveillés conjointement par les deux automates (Fig.II.25.b).

En phase d'initialisation, les arcs issus de q_0 vers les points singuliers communs ont des comptes rendus associés définis sur les combinaisons de l'ensemble des capteurs soit

$U_1 \cup U_2$ = (a,b,c,d,e,f).

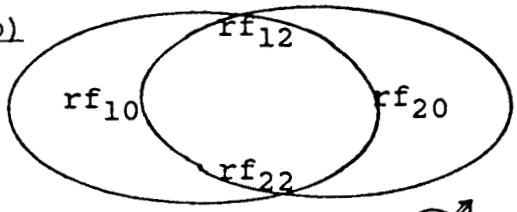
Figure II.25: Automate de diagnostic lorsqu'il y a partage de trajectoire.

(Fig.II.25.a:Chariot 1)

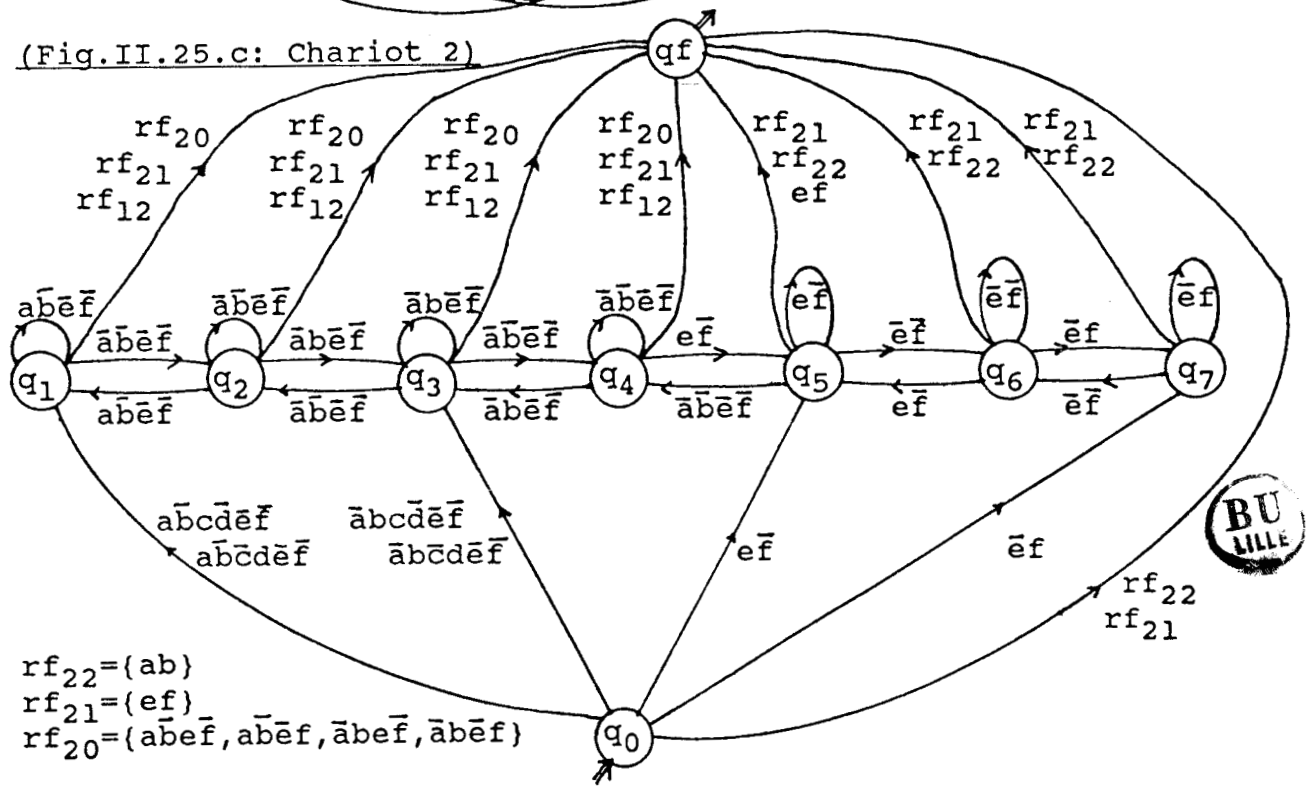


rf_{11}={cd}
rf_{12}={ab}
rf_{10}={abcd, abcd, abcd, abcd}

(Fig.II.25.b)

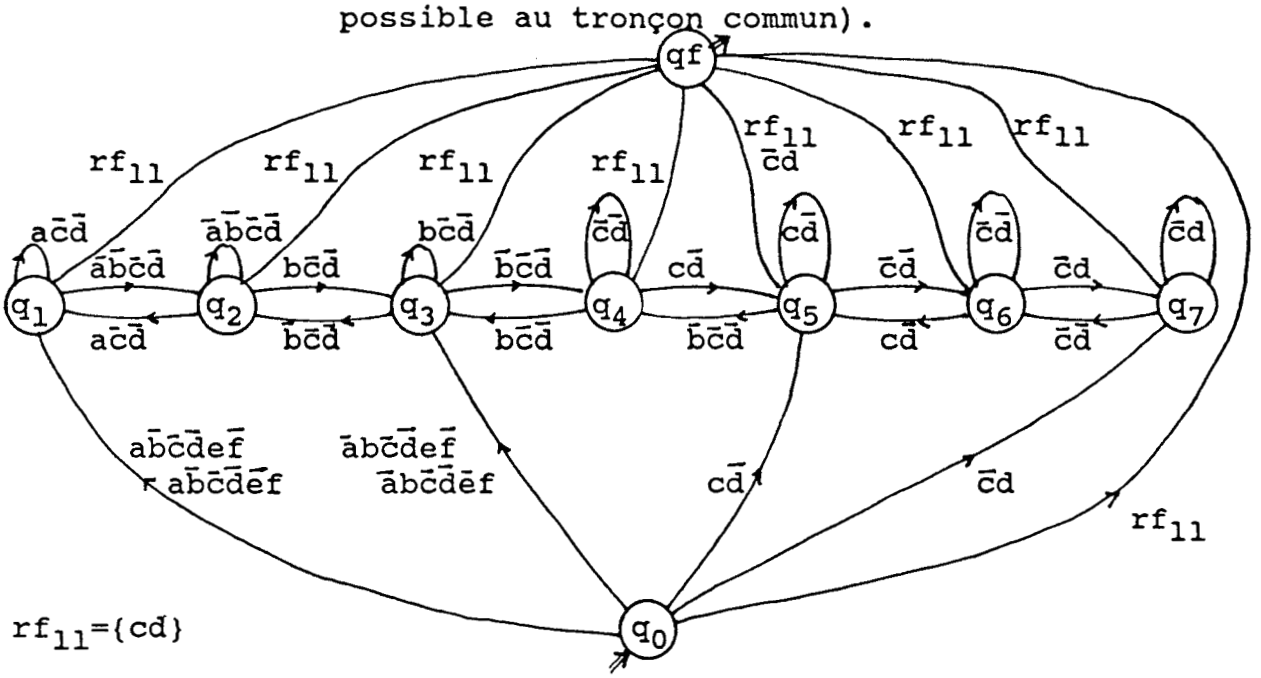


(Fig.II.25.c: Chariot 2)



rf_{22}={ab}
rf_{21}={ef}
rf_{20}={abef, abef, abef, abef}

(Figure II.25.d: Chariot 1 dans le cas d'un accès simultané possible au tronçon commun).



Lorsque la partie commune est simultanément accessible aux deux grandeurs les comptes rendus associés aux points singuliers sont différents ainsi que les arcs menant vers l'état final. Les capteurs du tronçon commun ne sont plus surveillés. La figure II.25.d. représente l'automate de diagnostic obtenu dans ce cas pour l'exemple.

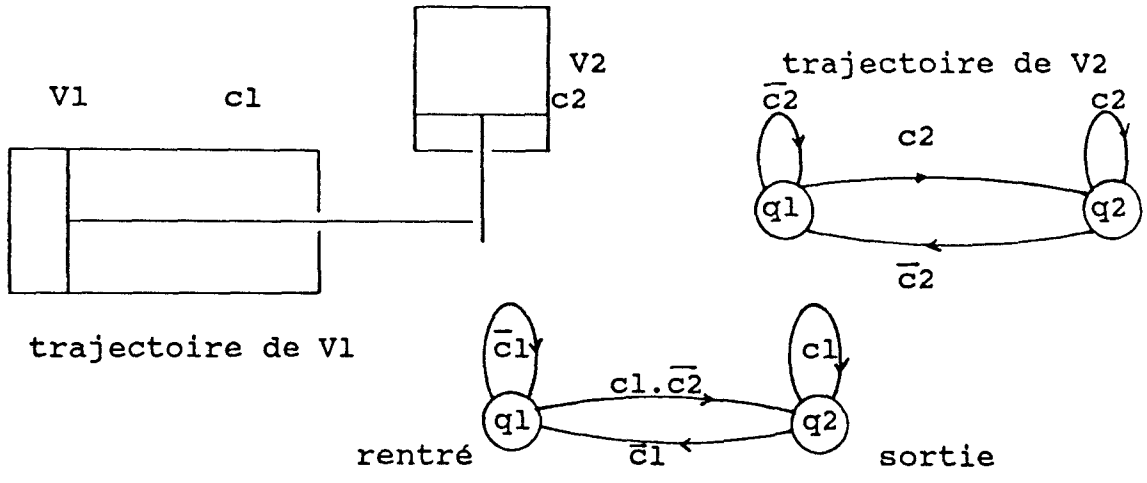
3.6.2.Modification d'une trajectoire par une autre

Nous distinguons deux types de modification possibles:

a) la modification a pour conséquence de créer un autre chemin possible ou bifurcation, c'est le cas du chariot représenté à la figure II.11 dont la trajectoire est modifiée par la position de l'aiguillage.

b) la modification se manifeste par un blocage de la grandeur ou une impossibilité d'évolution dans un sens; c'est le cas de l'exemple de la figure II.26 où la butée en position "sortie" empêche la tige du vérin de sortir.

Figure II.26: Blocage d'une grandeur.



Dans ces deux cas la solution consiste à introduire une condition supplémentaire afin de prévoir l'évolution de la grandeur dans le cas (a) ou éviter de sortir en défaut dans le cas (b). Cette condition est exprimée en fonction de l'état de la grandeur modificatrice (aiguillage ou butée). Si les points singuliers de cette grandeur sont codés d'une façon biunivoque par la position des capteurs, la condition d'accès peut être exprimée à partir de ces derniers.

Conclusion

Nous avons étudié dans ce chapitre différents modèles de test en ligne de la partie opérative pour lesquels le niveau de performance va croissant avec la complexité.

Le test dynamique décrit en utilisant le formalisme des grammaires régulières a un avantage certain par rapport au simple test statique. Mais il ne permet la détection que des défaillances affectant les capteurs et leurs liaisons avec le reste de l'automatisme.

La prise en compte des commandes appliquées permet d'éliminer les ambiguités du modèle précédent en dehors de la phase de localisation tout en gardant la simplicité des grammaires régulières. Ce modèle permet en plus de révéler toute divergence entre le sens d'évolution attendu d'une grandeur mesurée et son évolution réelle.

Le modèle dynamique pondéré fait appel à une grammaire programmée. Le dernier modèle évolue en fonction d'un programme qui prend en compte à la fois la commande appliquée et le temps normalement imparti pour atteindre un nouveau CR. C'est le modèle le plus performant mais necessite une connaissance approfondie de la Partie Opérative.

Il nous reste à étudier l'architecture matérielle et logicielle pour l'implantation de ce mécanisme de test en ligne.

CHAPITRE III: MISE EN OEUVRE ET IMPLANTATION DU MECANISME DE TEST EN LIGNE.

1. CHOIX D'UNE METHODE D'INTEGRATION DU MECANISME DE TEST EN LIGNE DANS L'AUTOMATISME

- 1.1. Différentes méthodologies d'intégration du mécanisme de test dans l'automatisme.
  - a. La méthode duplex
  - b. La méthode de l'inverse
  - c. Méthode de filtrage
  - d. Méthode par analyse syntaxique
- 1.2. Choix d'une architecture matérielle

2. MISE EN OEUVRE DU TEST PAR LA METHODE DUPLEX

- 2.1. Elaboration du modèle de test à l'aide d'une description par Grafcet
- 2.2. Actions réalisées en cas de défaut
- 2.3. Type de défaut signalé et origine probable
- 2.4. Synchronisation entre processeur de commande et de test pour les échanges d'informations

3. CHOIX D'UNE METHODE D'IMPLANTATION DES GRAFCET

- 3.1. Structure de données des graphes et gestion de celle-ci
- 3.2. Description des fonctions combinatoires
- 3.3. Structure de données du combinatoire local et général
- 3.4. Programme général de traitement

4. LANGAGE DE DESCRIPTION DE LA PARTIE OPERATIVE

- 4.1. Principes généraux de la description
- 4.2. Syntaxe de description: règles générales
- 4.3. Déclaration des variables
- 4.4. Description des fonctions combinatoires
- 4.5. Description de la partie "préactionneurs"
- 4.6. Description de la partie actionneurs/capteurs
- 4.7. Exemples de description de parties opératives

5. GENERATION DE LA STRUCTURE DE DONNEES TEMPS REEL ET EXPLOITATION

- 5.1. Génération de la structure de données
- 5.2. Exploitation temps réel

CHAPITRE III

MISE EN OEUVRE ET IMPLANTATION

DU MECANISME DE TEST EN LIGNE

Nous abordons dans ce chapitre les aspects relatifs à la mise en oeuvre du mécanisme de test en ligne de la partie opérative, en l'occurence l'implantation du modèle dynamique pondéré avec prise en compte des commandes appliquées.

Nous discutons, dans une première étape, du choix de la structure matérielle et logicielle compte tenu des remarques faites au chapitre 1 relatives au compromis sécurité/fiabilité de différentes architectures.

La solution retenue utilise un modèle de comportement de la partie opérative. Nous avons opté pour une description de ce modèle à l'aide du formalisme grafcet en y intégrant le test.

L'élaboration de ce modèle pour une application donnée nécessite un langage de description de la PO et un logiciel de traduction afin de créer la structure de données qui sera utilisée en temps réel.

1.CHOIX D'UNE METHODE D'INTEGRATION DU MECANISME DE TEST EN LIGNE DANS L'AUTOMATISME

1.1.Différentes méthodologies d'introduction du mécanisme de test dans l'automatisme

L'introduction d'un mécanisme de test en ligne dans un automatisme peut être envisagée de différentes manières. Le principe de certaines méthodes a été exposé au chapitre 1 paragraphe 5.3 à propos des systèmes autotestables.

Nous décrivons ci-après différentes méthodes d'intégration possibles. Certaines d'entre elles utilisent un modèle de comportement (direct ou inverse) de la partie opérative.

a) La méthode duplex

Cette procédure, décrite à la figure III.1, utilise un modèle de comportement du processus à surveiller et un comparateur. Dans notre cas, le modèle de la partie actionneurs/capteurs est tel que les étiquettes associées aux arcs seraient dépendantes des seules commandes appliquées. En phase d'initialisation les arcs sont étiquetés par les comptes rendus.

Le comparateur met en évidence les discordances entre les comptes rendus r(t) et r'(t) présents à l'instant t et correspondant respectivement au processus et au modèle.

La tolérance sur les temps d'évolution est introduite en masquant le signal de défaut durant l'intervalle admissible $I(\alpha ij)$ associé à l'arc αij. La connaissance de l'état présent et de la commande appliquée permet en effet de prévoir, outre le compte rendu attendu, l'intervalle de tolérance à appliquer.

Cette solution qui sera adoptée ici est décrite en détail dans la suite de cet exposé.

b) Méthode de l'inverse:

Dans cette méthode (figure III.2) un modèle inverse de la partie opérative est utilisé. Dans le modèle inverse de la partie act./capt. les arcs sont étiquetés par les seuls comptes rendus $r \in R$. Le passage de l'état qi à l'état qj permet de choisir parmi les éléments de C' susceptibles de provoquer cette transition, l'élément c' le plus probable compte tenu du temps nécessaire pour atteindre qj.

L'application de c' au modèle inverse de la partie préactionneurs (si celui-ci existe) génère alors une classe d'ordres à laquelle doit appartenir l'ordre réel o appliqué.

Cette méthode comporte donc, en plus, une procédure de classification automatique afin de déterminer la commande (sens et module) à partir du temps mis pour changer de CR.

Figure III.1: Méthode duplex.

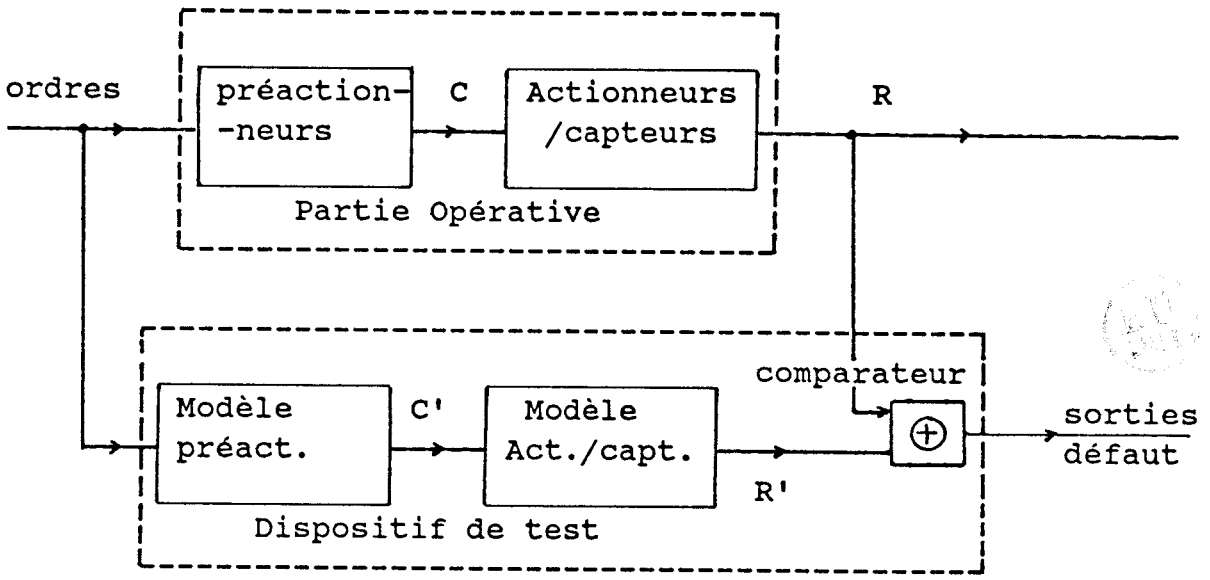
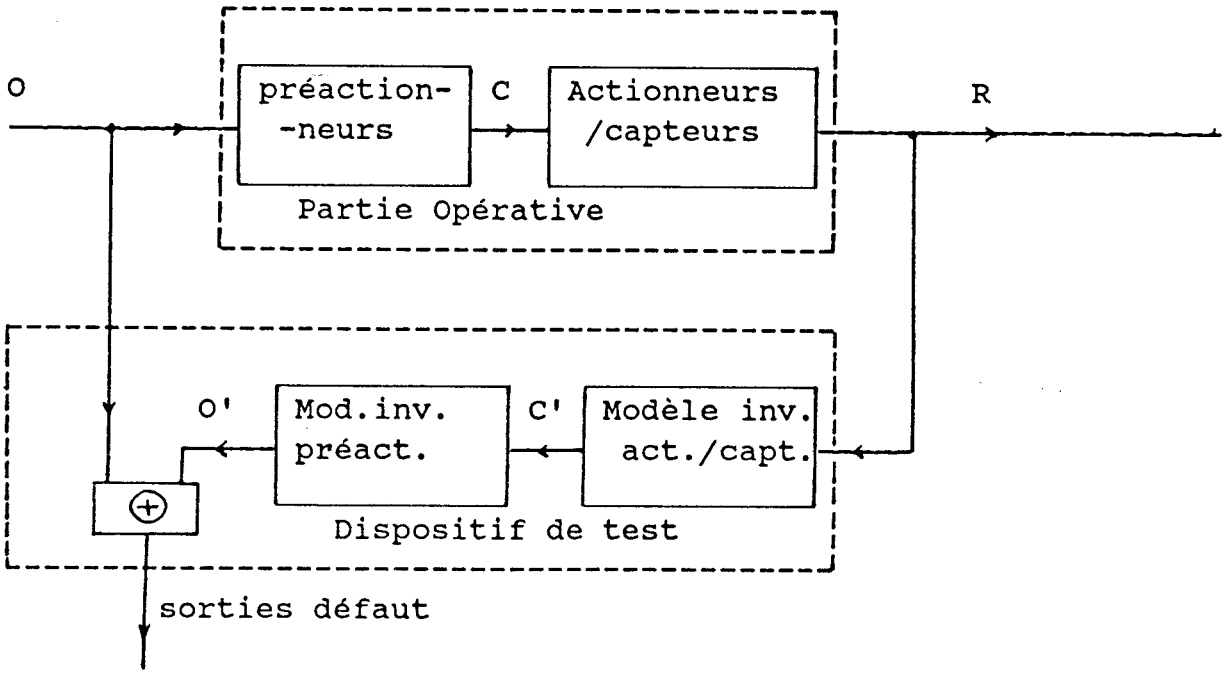


Figure III.2: Méthode de l'inverse.



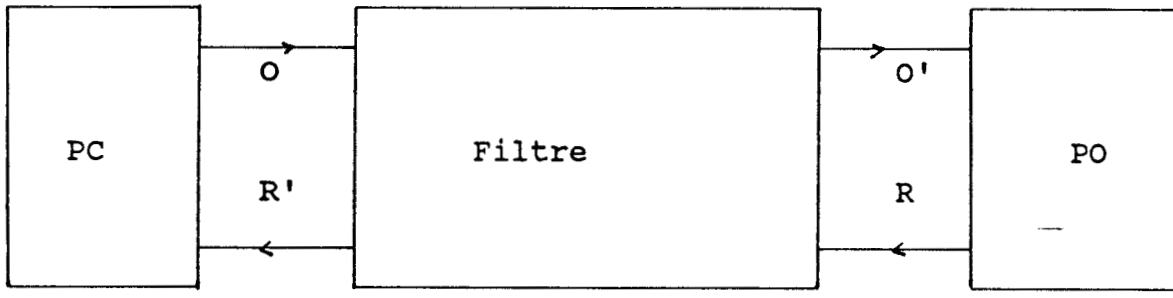
c) Méthode de filtrage:

Dans cette méthodologie (figure III.3) l'idée première est d'éviter la propagation des erreurs de la PO vers la PC et vice-versa. Il y a alors intégration du mécanisme de détection et de réaction dans un même dispositif appelé filtre et placé en transmission entre PO et PC.

Dans notre étude nous nous sommes interéssé aux seules erreurs issues de la PO. Dans ce sens le filtrage consiste à ne transmettre vers la PC que les éléments r ϵ R compatibles avec l'état du modèle.

Dans le sens PC -> PO les ordres sont filtrés de la même façon en ne retransmettant que ceux compatibles avec l'état de la PO.

Figure III.3: filtrage.

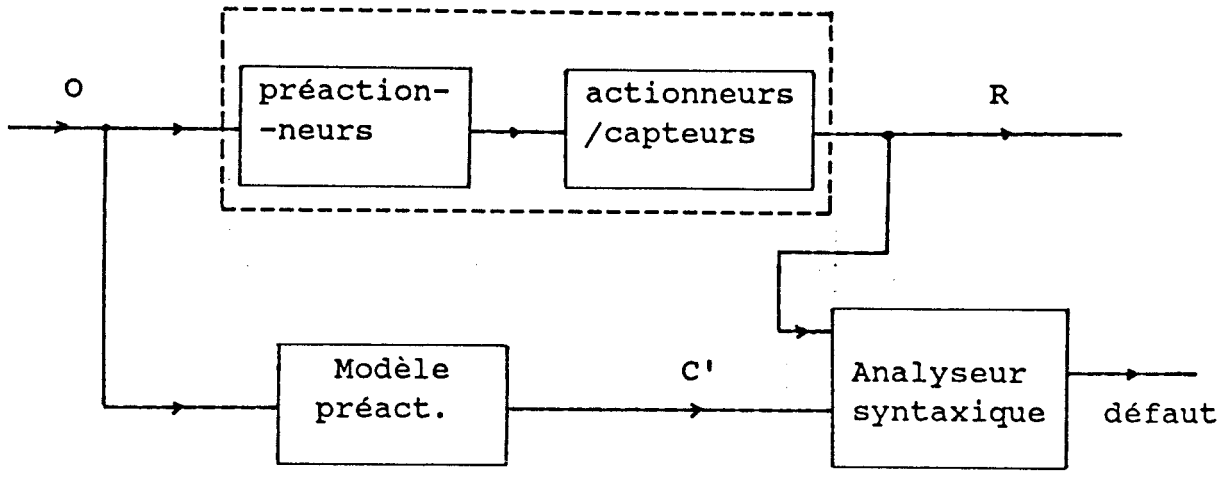


d) Méthode par analyse syntaxique

Dans l'analyse syntaxique (figure III.4) le passage par un état final génère un signal de défaut mettant en évidence les incohérences de syntaxe.

Cette procédure est développée par J.Defrenne dans [51]. Elle utilise une structure de données propre.

Figure III.4: analyse syntaxique.



1.2. Choix d'une architecture matérielle

Nous avons choisi de séparer le mécanisme de détection et celui de réaction. Le problème de la reconfiguration et de la réaction en cas d'erreur n'a pas fait l'objet de notre étude. Il est en effet possible de réagir différemment suivant le type d'erreur détecté. Ce choix est laissé à l'initiative de l'utilisateur.

L'architecture choisie est représentée à la figure III.5. Elle est décrite par Naïfi dans [35] où le dispositif de test est supporté par un micro-ordinateur.

Le système de détection utilise les ordres et les comptes rendus mémorisés par l'automate de commande.

Le volume de traitement nécessité par le test pouvant être important, nous avons choisi d'utiliser un processeur spécialisé pour le test. L'échange d'informations peut être réalisé à travers une laison série, par exemple.

Cette disposition permet d'éviter la duplication des interfaces d'entrée/sortie de la PC réalisant ainsi un compromis sécurité/fiabilité.

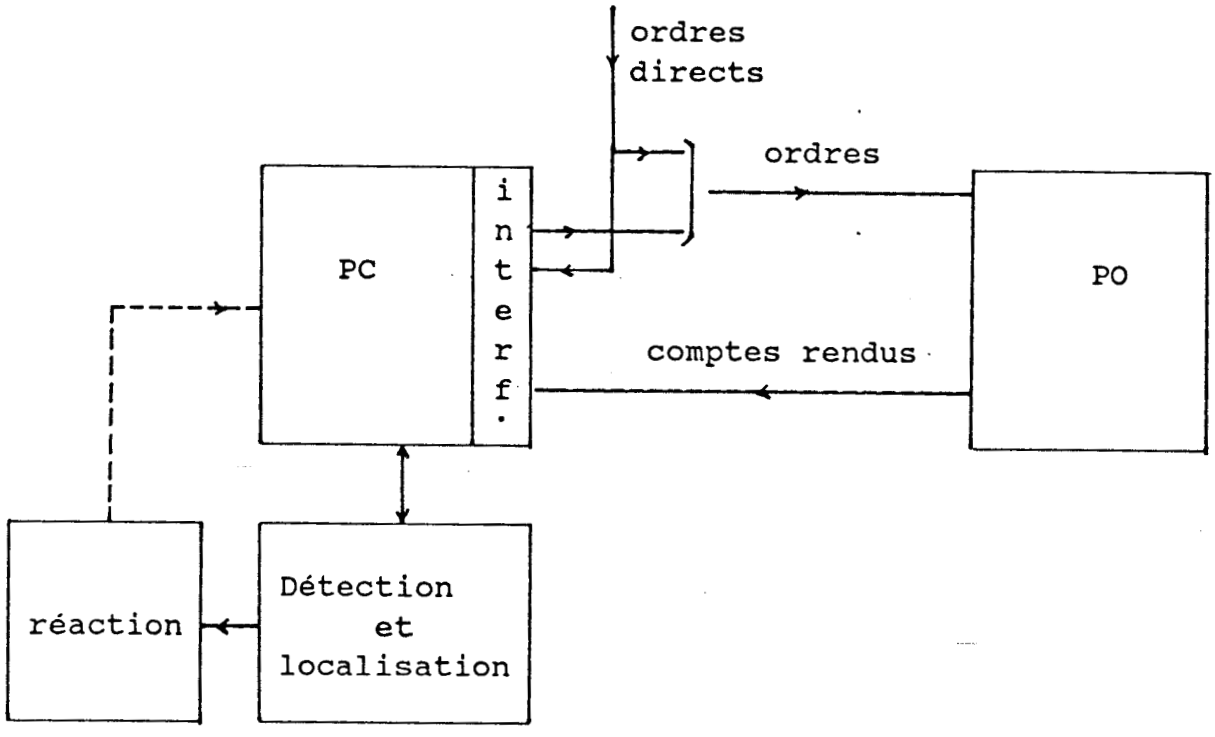
Matériel couvert par le test:

Dans cette architecture, les interfaces d'E/S de l'automate de commande sont considérées comme faisant partie de la PO et sont donc couvertes par le test.

Nous avons supposé l'existence d'ordres appliqués directement à la PO sans passer par la commande. La prise en compte de ces informations est indispensable au bon fonctionnement du test. Nous suggérons de relier ces lignes d'ordres vers la PC afin de simplifier le problème de synchronidsation entre les deux processeurs.

Un contrôle des échanges entres les deux unités commande-surveillance doit être assuré par exemple par une procédure d'appel-réponse ou hand-shaking.

Figure III.5: Architecture matérielle.



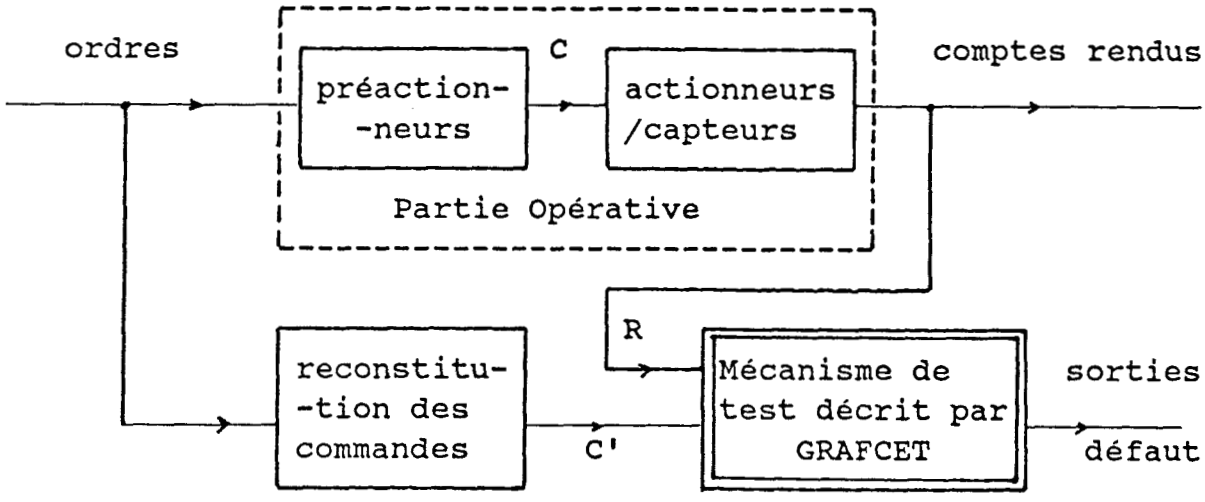
2. MISE EN OEUVRE DU TEST PAR LA METHODE DUPLEX

2.1. Elaboration du modèle de test à l'aide d'une description par Grafcet

Nous avons choisi de décrire le modèle de la partie actionneurs/capteurs à l'aide du Grafcet en y intégrant le comparateur. Ce qui donne le schéma de la figure III.6.

Cette procédure peut permettre éventuellement d'utiliser le même matériel pour la commande et le test, en l'occurence des automates programmables du commerce.

Figure III.6: Description à l'aide du Grafcet.

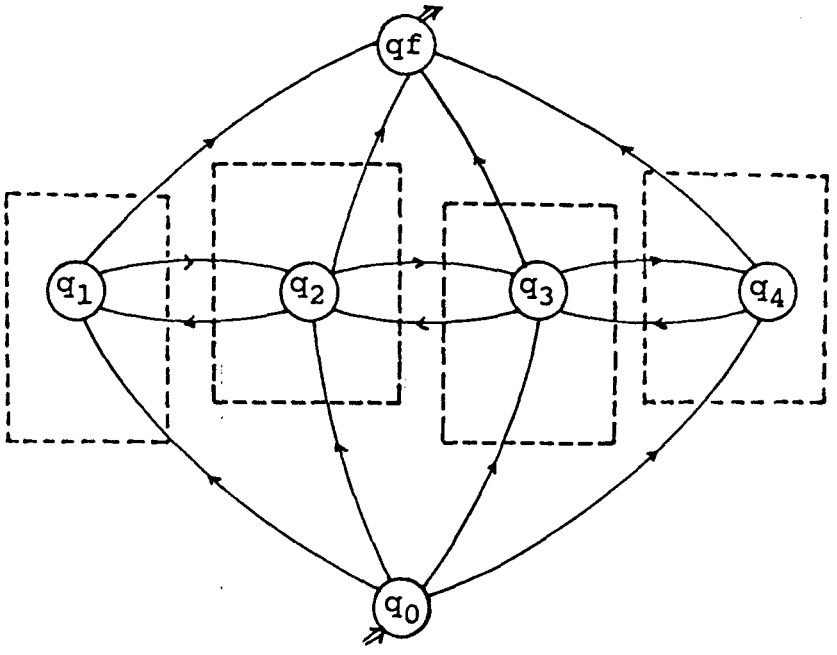


La description par grafcet est faite à partir du modèle de test pondéré décrit à la fin du chapitre II. Nous associons un grafcet à chaque point singulier (Fig.III.7). Ce graphe spécifie donc les évolutions possibles à partir du PS considéré. La figure III.9 représente un tel graphe où le test y est intégré.

La liaison entre les différents graphes est réalisée grace à un ensemble de variables internes mémorisées. A chaque graphe i est associé un indicateur v_{1i} qui est initialement à zéro. Il est positionné à 1 lorsque la PO atteint ce PS. Il est remis à 0 dès que le franchissement est effectué.

Cette variable est utilisée dans les réceptivités des grafcet associés aux points singuliers.

Figure III.7: association d'un grafcet à chaque point singulier



a) En fonctionnement normal:

L'évolution d'un graphe de l'étape initiale 0 à l'étape 1 que nous appelons éveil peut être provoquée par:

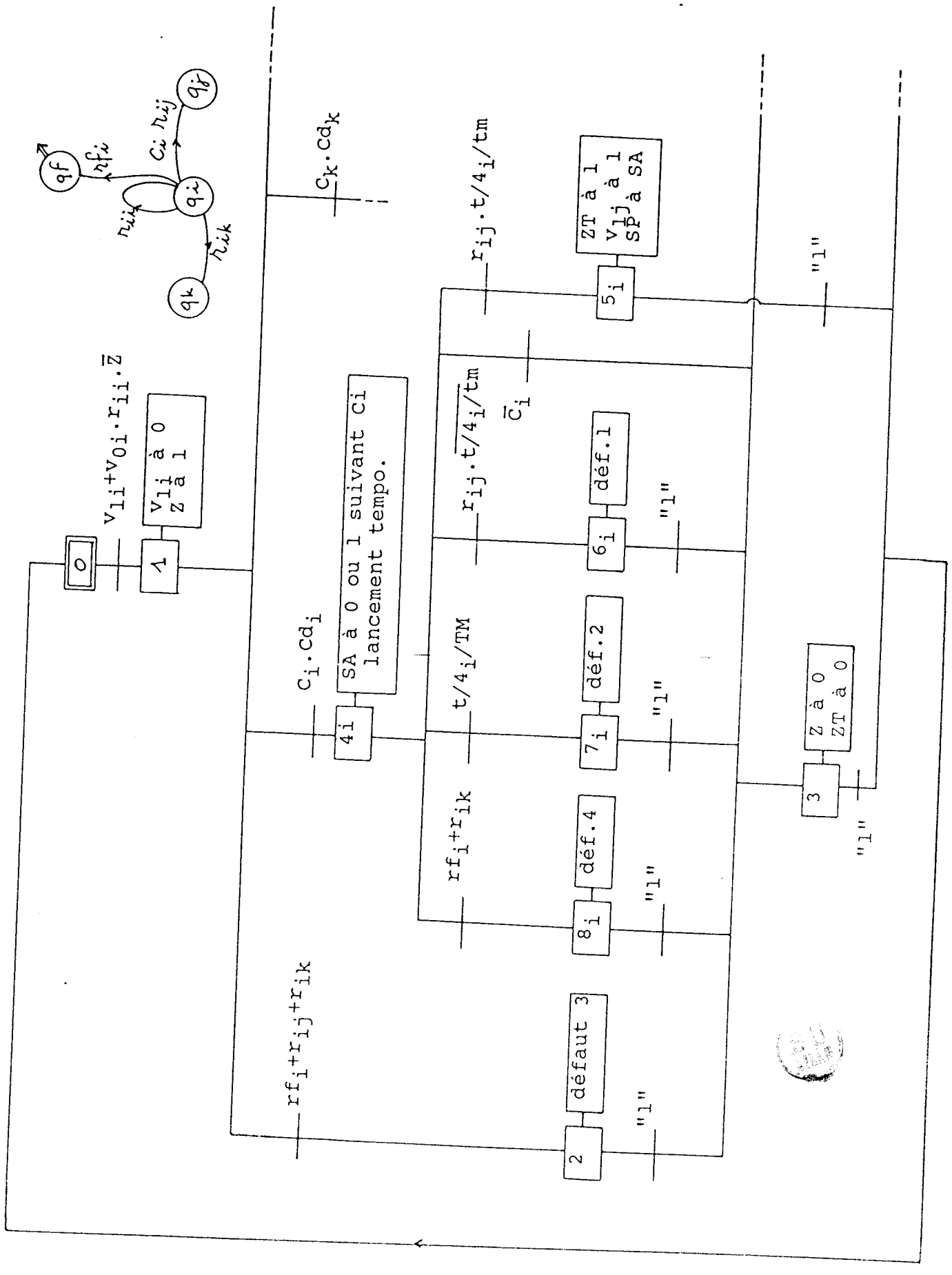
- la présence du compte rendu r_{ii} en phase de localisation du modèle si celui-ci est déterministe (indiqué par v_{0i}). La phase de localisation est signalée par l'indicateur noté Z positionné à 0 pendant cette phase.
- l'arrivée sur le point singulier correspondant signalée par la mise à 1 de l'indicateur v_{1i}.

A l'étape 1 l'indicateur Z est positionné à 1 pour signaler que le modèle est localisé. La variable d'interconnexion v_{1i} est remise à zéro.

En fonctionnement normal, l'application d'une commande C_i non nulle fait évoluer le graphe vers une des étapes 4_i correspondant à cette commande. La condition associée Cd_i permet la prise en compte éventuelle d'une interaction entre trajectoires se manifestant par un blocage de l'évolution d'une grandeur. Une temporisation est alors lancée afin de contrôler les temps d'évolution.

L'atteinte du point singulier suivant dans le temps imparti fait évoluer le graphe vers l'étape 5i où l'indicateur v_{1j} correspondant au PS atteint est positionné à 1 avant de revenir à l'étape initiale.

Figure III.9: Graphe d'état associé à un point singulier



b) les différents cas de défaut:

Nous distinguons 4 types d'évolutions anormales:

- à partir de l'étape 1, tout changement de compte rendu est considéré comme une erreur (en dehors des capteurs partagés). Le passage vers l'étape 2 met en évidence cette anomalie.

- à partir de l'étape 4_i, l'évolution du graphe vers l'étape 8_i met en évidence la présence d'un compte rendu non attendu.

- l'évolution vers l'étape 7_i se produit lorsque le temps maximum imparti à l'action élémentaire est dépassé.

- Lorsque le point singulier suivant est atteint avant le temps minimum, l'évolution du graphe se fait vers l'étape 7_i.

Dans tous ces cas de défaillance détectée, en dehors de la signalisation du défaut, une réinitialisation est effectuée par la mise à 0 de l'indicateur Z à l'étape 3.

c) Cas particuliers:

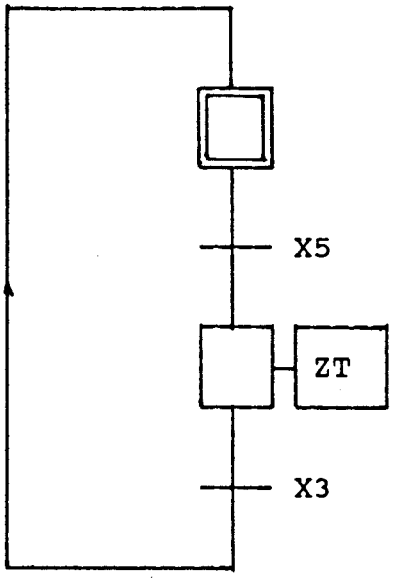
Dans le cas particulier où la commande appliquée est interrompue la réinitialisation est effectuée mais l'anomalie n'est pas signalée.

En phase 5_i, il peut y avoir plusieurs PS atteignables. C'est le cas des trajectoires où il existe des bifurcations. Si on ne dispose pas d'information sur le chemin empreinté, l'indicateur v_{1j} à positionner sera donné par le compte rendu atteint.

d) Synchronisation en phase de localisation:

Nous avons vu qu'en phase de localisation les intervalles de tolérance devaient être élargis afin d'éviter de sortir en défaut. Cet élargissement est réalisé comme décrit au chapitre II. Nous utilisons une variable mémorisée ZT qui est initialement à zéro (Fig.III.10). Elle est remise à zéro chaque fois qu'une anomalie est signalée (étape 3). Les intervalles élargis sont alors applicables. ZT est positionnée à 1 pour appliquer les intervalles de temps normaux dès qu'une action élémentaire s'est déroulé correctement (étape 5_i).

Figure III.10: gestion de l'indicateur ZT.



e) Prise en compte de la forme de la came et des capteurs

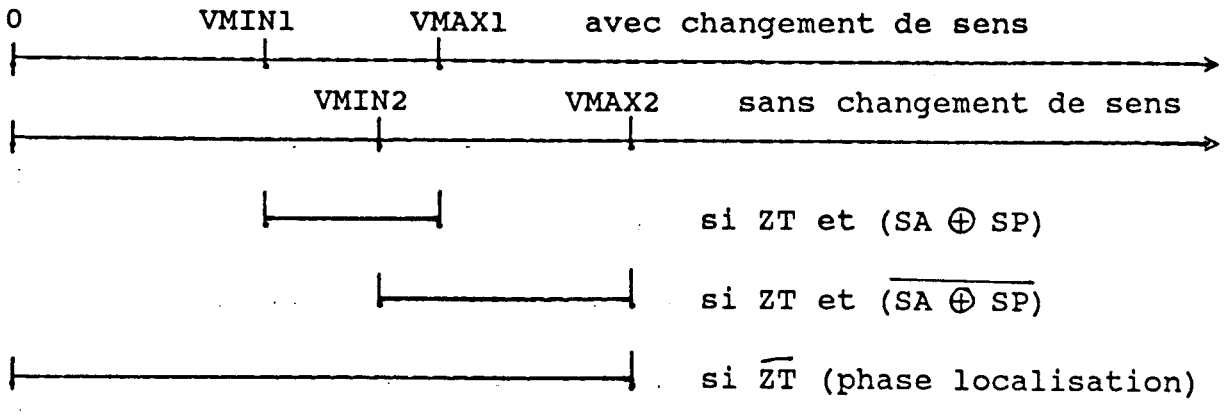
La prise en compte de la forme de la came ou des capteurs nécessite la connaissance du sens de déplacement sur la trajectoire; en particulier le sens de déplacement au cours de l'action élémentaire précédente afin de déterminer l'intervalle de temps à appliquer. Nous utiliserons pour cela deux variables internes mémorisées SA et SP indiquant respectivement le sens de l'évolution en cours et celui de l'action élémentaire précédente.

SA est positionnée au début de l'action élémentaire (à l'activation de l'étape 4i) à 0 ou 1 suivant le sens de la commande présente.

SP mémorise le sens de la commande associée à l'action élémentaire précédente. SP est positionnée à la même valeur que SA dès que l'action élémentaire en cours est achevée (à l'activation de l'étape 5i).

Nous avons ainsi quatre valeurs de temps de base à définir et 3 situations possibles pour l'intervalle de tolérance à appliquer. Nous supposons que les temps sans changement de sens sont plus grands que ceux avec changement (fig.III.11).

Figure III.11: gestion du changement de sens.



2.2. Actions réalisées en cas de défaut

Le passage par une des étapes 2, 6, 7 ou 8 révèle donc une erreur. Les actions réalisées à l'activation d'une de ces étapes sont choisies en fonction de deux objectifs:

a) Le premier objectif concerne la signalisation de l'erreur vers le dispositif de réaction-reconfiguration. Dans le but d'une réaction de sécurité par l'automate de commande, nous avons choisi de réaliser la signalisation à l'aide d'un code de faute tenant dans un mot de 16 bits. Il y a un code de faute possible par type d'erreur et par action élémentaire. La valeur de ce code peut être programmée par l'utilisateur en fonction des réactions souhaitées.

b) Le deuxième objectif est une aide pour la maintenance. Afin de localiser l'origine probable de la défaillance avec plus de précision, un certain nombre d'informations sont enregistrées et disponibles pour une éventuelle consultation.

Les informations enregistrées dans l'historique sont:
- la sous partie opérative concernée,
- le PS concerné,
- le type de défaut détecté (1, 2, 3 ou 4),
- la commande appliquée,
- le PS atteint ou à atteindre s'il est connu.
- le numéro de la temporisation utilisée. (Ce numéro correspond au numéro de l'action élémentaire).

2.3. type de défaut signalé et origine probable

Nous résumons ci-après les anomalies signalées et leurs origines probables.

Défaut 1: PS atteint trop rapidement (avant le temps minimum).
Cette erreur peut avoir comme origine probable:
- une augmentation anormale de la vitesse d'évolution suite à une défaillance d'actionneur ou préactionneur,
- le collage du capteur à une valeur qui est attendue dans l'intervalle [0,Tm[.

Défaut 2: PS non atteint alors que le temps maximum est écoulé.
Ce défaut peut avoir pour causes:
- un blocage de l'actionneur,
- un blocage du capteur dont le changement est attendu.

Défaut 3: Changement de compte rendu pendant la présence d'une commande d'arrêt. Les causes possibles sont:
- collage d'un capteur occasionnant un CR impossible ou non attendu,
- Déplacement non commandé de l'actionneur.

Défaut 4: Arrivée d'un compte rendu non attendu.
Ce compte rendu arrive dans l'intervalle]Tm, TM[et ne correspond pas à celui attendu. Les causes possibles sont:
- changement de sens d'évolution de l'actionneur ou changement de trajectoire,
- collage d'un capteur dont la variation n'est pas attendue.

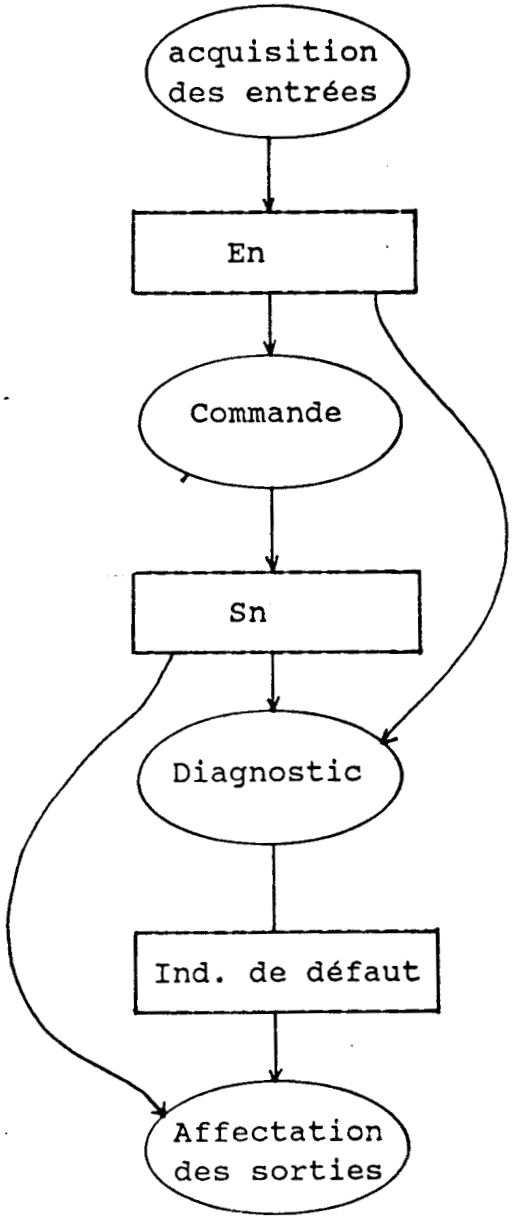
Nous constatons que la connaissance du seul type de défaut détecté ne suffit pas à localiser la panne. La consultation de l'historique enregistré avec les informations citées ci-dessus permet de mieux cerner la cause probable de la défaillance.

2.4. Synchronisation entre processeur de commande et de test pour les échanges d'informations

Le modèle de test, tel que décrit par Grafcet, travaille sur les comptes rendus échantillonnés par la PC au début d'un cycle de traitement et les ordres élaborés à la fin de ce cycle. De plus la situation du graphe de commande doit être stable à la fin de ce cycle (pas d'évolution sans changement des entrées); ceci, afin d'éviter l'éveil intempestif du Grafcet de test au moment d'un changement de commande.

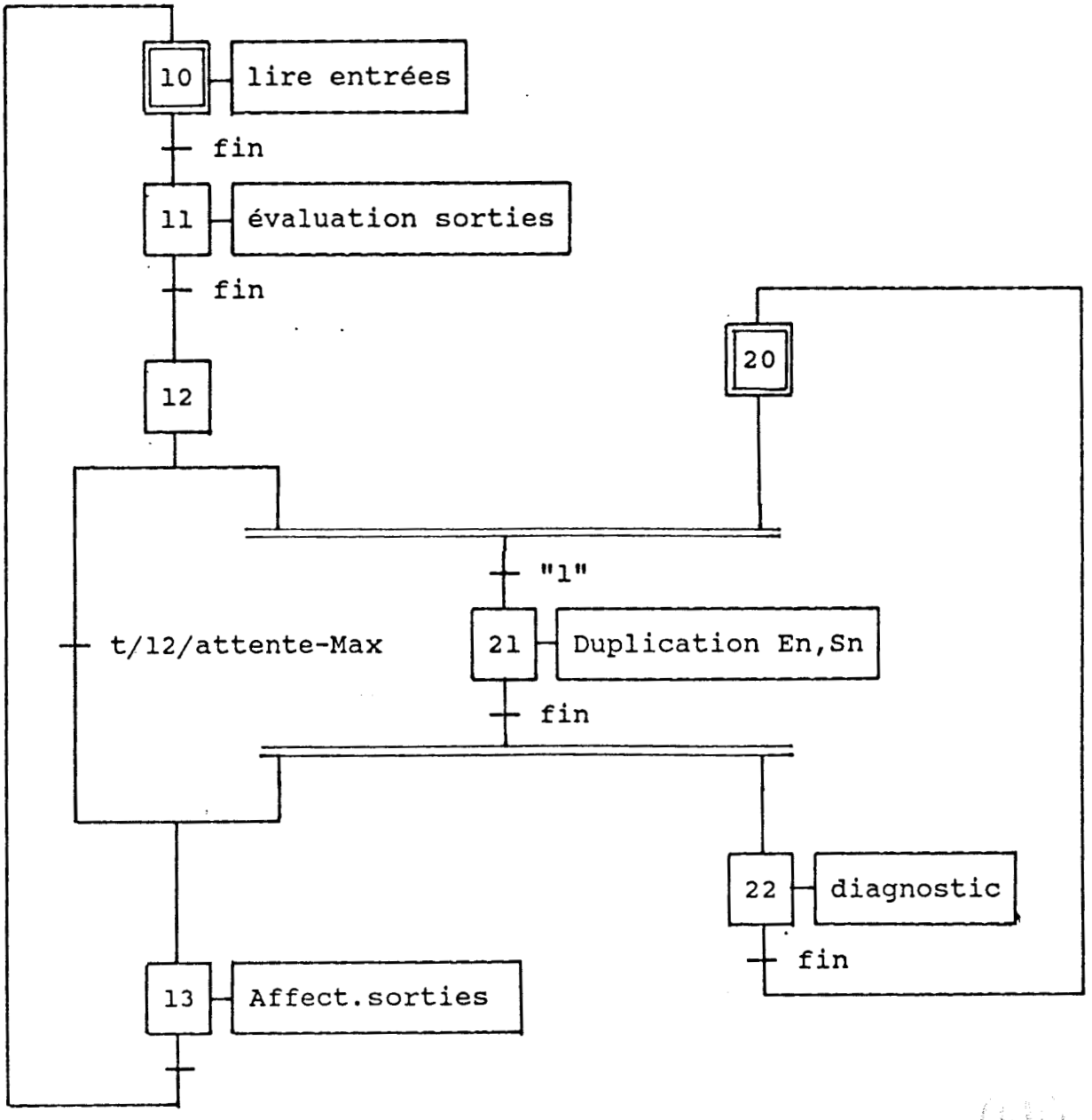
Dans le cas où les deux tâches commande et test sont confiées à un même processeur, les différentes tâches de traitement doivent être éxécutées suivant le diagramme de la figure III.12. Le processeur, après acquisition des entrées, traite successivement les tâches de commande puis de test. L'affectation des sorties ou ordres peut être conditionnée par le résultat du test.

Figure III.12: Traitement dans le cas monoprocesseur.



Lorsqu'on utilise un processeur spécialisé pour le test, la synchronisation est réalisée suivant le grafcet de la fig.III.13. Afin d'éviter des blocages, nous suggérons de donner l'initiative de l'échange à la partie commande pour une meilleure sécurité. De plus, une temporisation limite le temps d'attente pour la réponse du processeur de test.

Figure III.13: Grafcet de synchronisation dans le cas de deux processeurs spécialisés.



3. CHOIX D'UNE METHODE D'IMPLANTATION DES GRAFCET

Plusieurs méthodologies d'implantation sont possibles [38], [39]. Dans le cadre de la réalisation que nous avons développée sur un micro-ordinateur, nous avons choisi l'implantation d'un ensemble de graphes d'états avec une structure orientée données. Un programme fixe gère alors cette structure en temps réel [56]. Nous rappelons ci-après le principe de cette méthode.

3.1. Structure de données des graphes et gestion de celle-ci

a) Structure de données d'un graphe d'état:

Chaque graphe d'état est décrit par une structure de données telle que représentée à la figure III.14. Chaque étape i est repérée par son adresse Adpi. Pour chaque étape on trouve:

- l'adresse du combinatoire local correspondant ou actions à éxécuter,
- l'adresse des réceptivités associées aux transitions de sortie ainsi que les étapes de sortie correspondantes.

FL est un indicateur de Fin de Liste.

A chaque graphe d'état est associé un pointeur qui contient l'adresse Adpi de l'étape active en cours. A l'initialisation, ce pointeur contient l'adresse de l'étape initiale. Avec une telle structure de données, la table des marquages ne donne que l'étape active en cours.

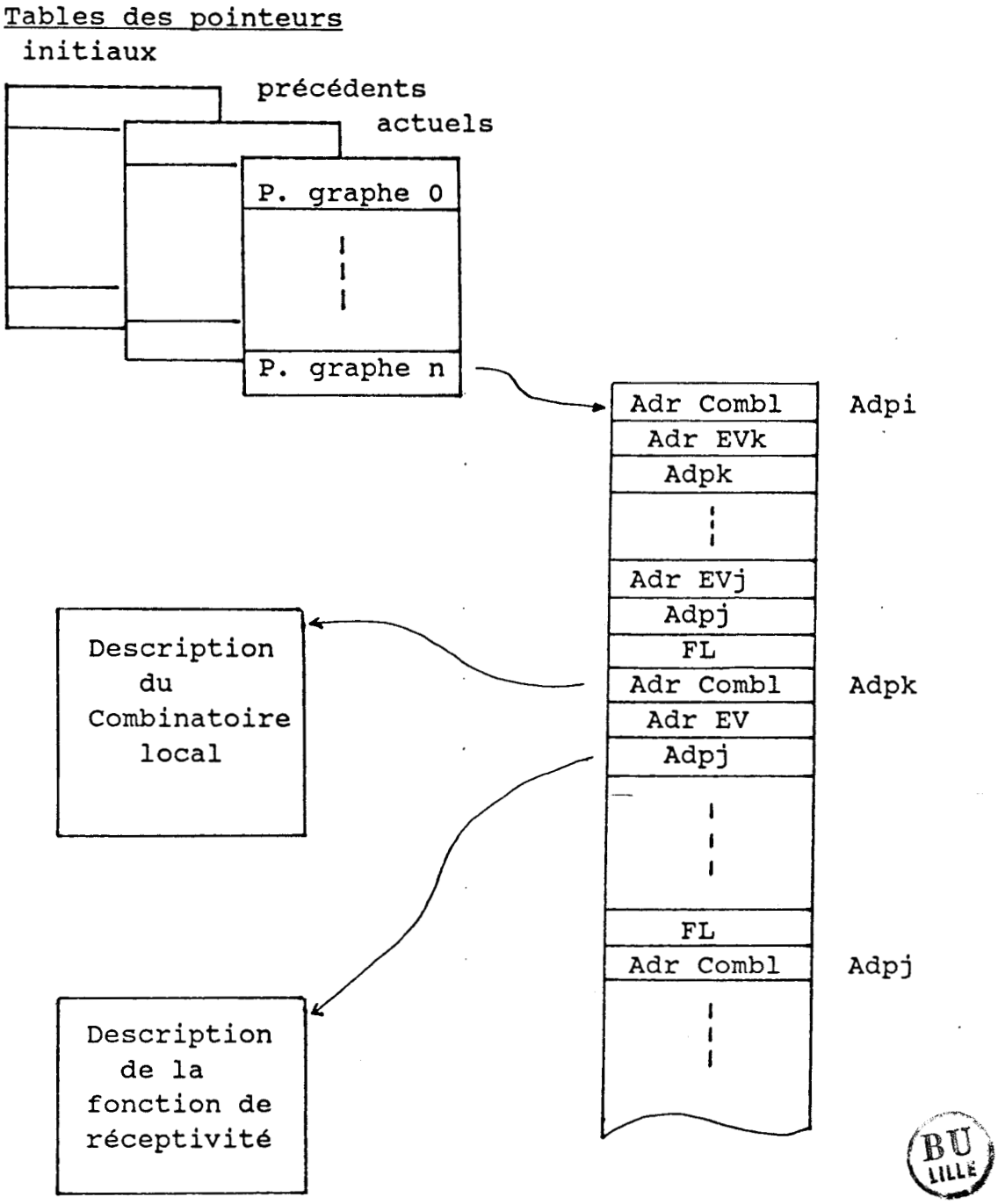
Il n'est pas associé de variable interne Xi systématiquement aux étapes. Il convient donc de considérer les variables Xi, associées aux étapes et définies selon les besoins, comme des variables internes qui doivent être prises en compte dans le combinatoire local.

b) Gestion des graphes d'état:

Tous les graphes d'état sont traités un par un de la façon suivante. A partir de la valeur du pointeur donnant l'adresse de l'étape active en cours, on évalue dans l'ordre les récèptivités associées aux transitions de sortie jusqu'à l'indicateur FL.

Dès qu'une de ces transitions est validée l'adresse de l'étape de sortie correspondante est prise comme la nouvelle valeur du pointeur du graphe.

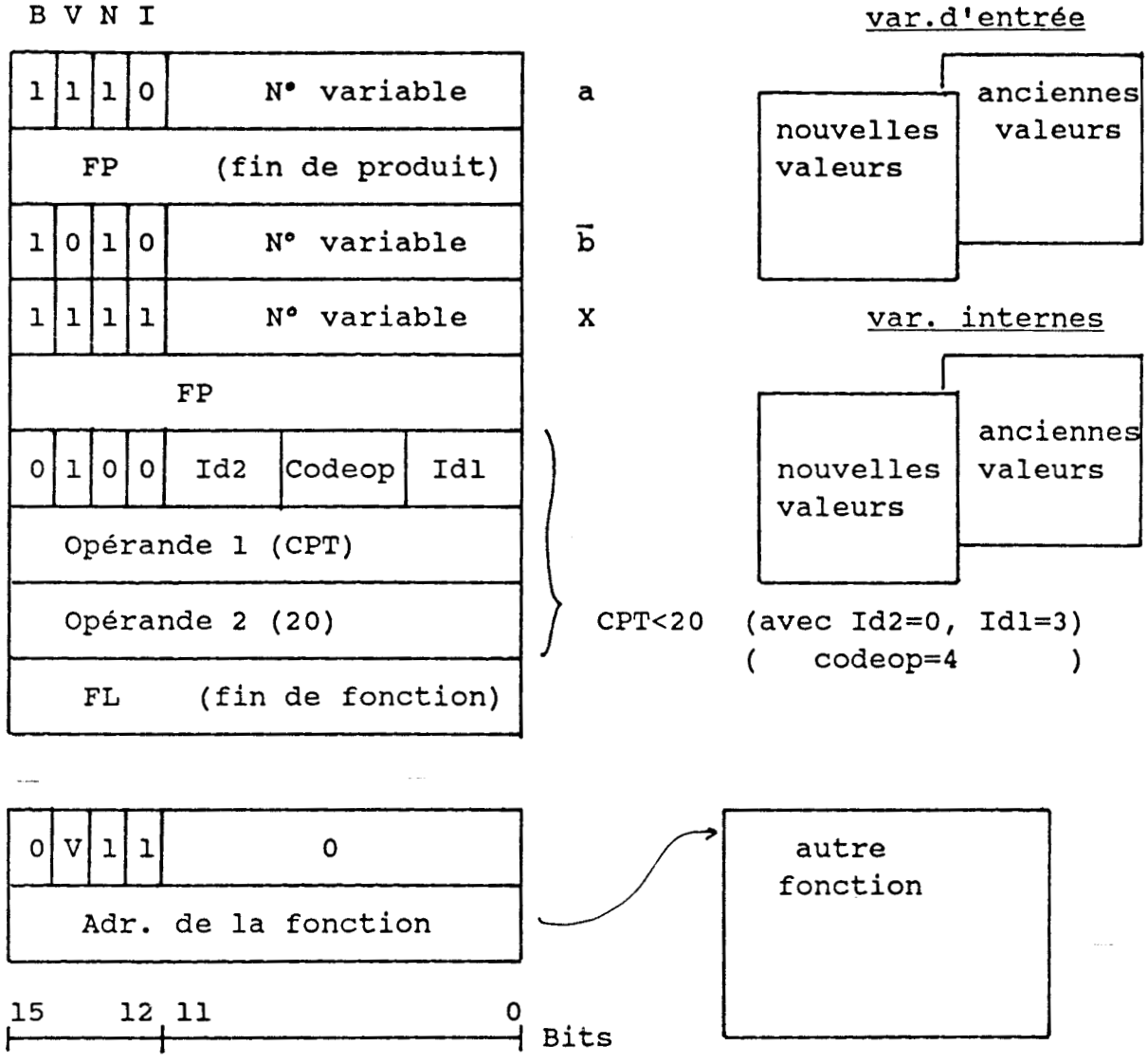
Figure III.14: structure de données des graphes d'état.



3.2. Description des fonctions combinatoires:

La description des fonctions combinatoires utilise une structure orientée données et un programme fixe pour leur évaluation. La fonction est décrite sous forme d'une somme de produits ou implicants premiers sans parenthèses. La figure III.15 donne un exemple.

Figure III.15: Exemple de description d'une fonction.
soit F = a + $\bar{b}$X + CPT < 20
où CPT est une variable numérique d'entrée.
a,b sont des variables d'entrée booléennes.
X est une variable interne booléenne.



Les différents indicateurs utilisés ont les significations suivantes:

B: B=1 indique une variable booléenne, B=0 sinon.
V: indique la valeur attendue, directe ou complémentée.
N: indique s'il s'agit de l'ancienne ou nouvelle valeur pour la détection des changements d'une variable ou front.
I: indique une variable interne (I=1) ou de sortie.

Les codes Id1 et Id2 identifient respectivement le type d'opérande 1 et 2. Les codes possibles sont les suivants:

0: constante
1: adresse de la variable
2: N de l'ancienne variable d'entrée
3: N de la nouvelle variable d'entrée
4: N de l'ancienne variable interne
5: N de la nouvelle variable interne
6: N de la variable de sortie (pour le combinatoire).

Les codes opératoires de comparaison possibles entre des valeurs numériques non signées sur 16 bits sont les suivants:
0: <> (différents)
1: >= (supérieur ou égal)
2: <= (inférieur ou égal)
3: > (supérieur)
4: < (inférieur)
5: = (égal)

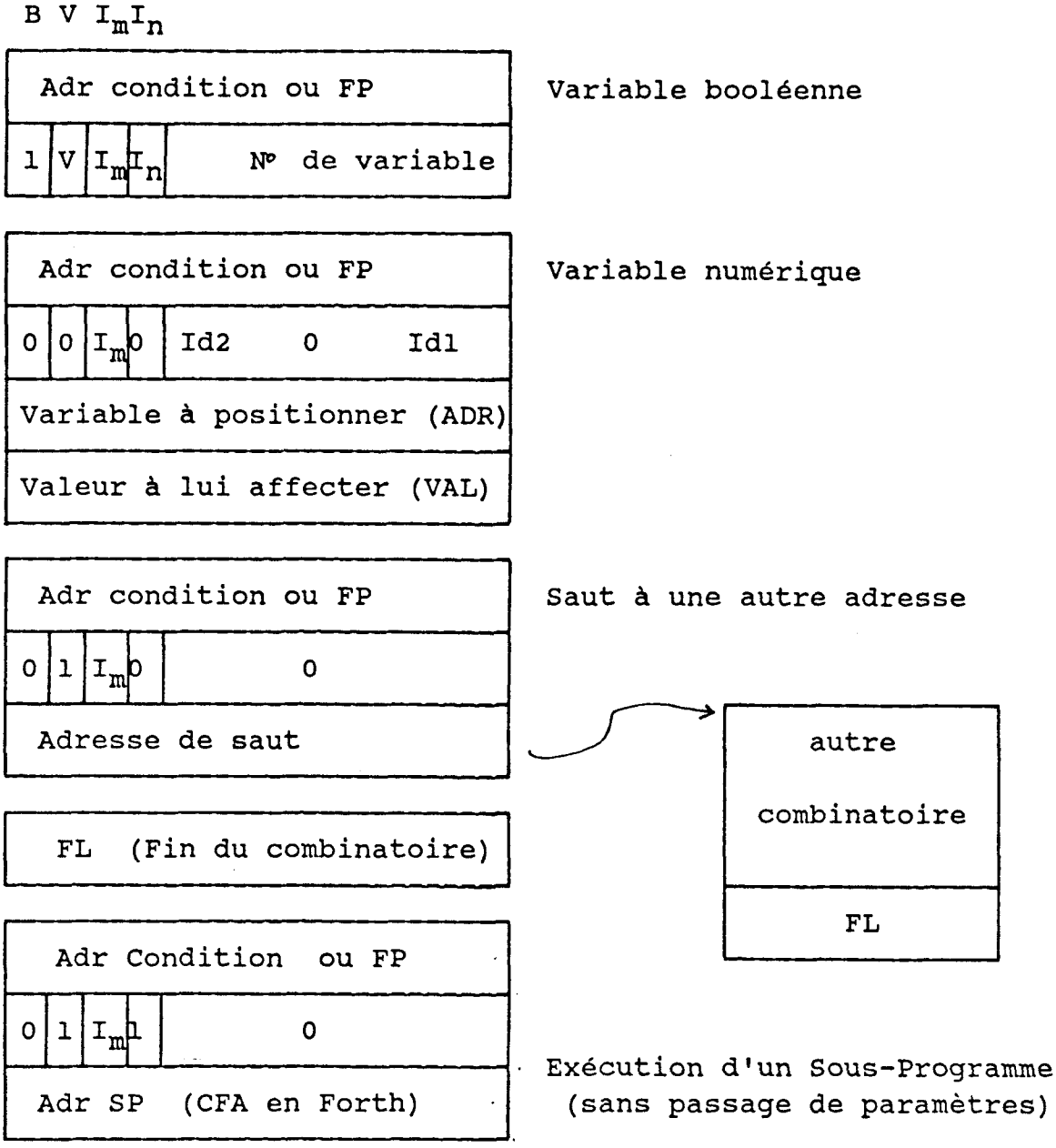
Le numéro de variable donne son adresse relative et sa position dans le mot de 16 bits. Dans le cas d'une variable numérique ce numéro correspond directement à son adresse relative.

La gestion de la structure de données ou évaluation de la fonction fait appel à une procédure récursive. L'évaluation est réalisée de façon à optimiser le temps de traitement (saut au produit suivant dès qu'une variable d'un minterme est nulle).
L'évaluation des fonctions combinatoires représente en effet un temps de traitement non négligeable.

3.3. Structure de données du combinatoire local et général:

La structure de données du combinatoire (local, général) décrit les actions à réaliser. Nous distinguons deux types de variables internes ou de sortie concernées par le combinatoire: variables continues ou mémorisées.
Un indicateur associé à chaque variable interne ou de sortie précise ce type. Cet indicateur est à 1 si la variable est mémorisée.
Toutes les actions se résument à un positionnement à une valeur donnée des variables booléennes ou numériques. Il existe deux types d'actions: celles qui ne sont effectuées qu'à l'activation de l'étape (actions impulsionnelles) et celles qui sont effectuées pendant toute la durée d'activation de l'étape (actions continues). Ces deux types d'actions peuvent être conditionnelles.

Figure III.16: Structure de données du combinatoire.



Les indicateurs utilisés dans la structure de données ont les significations suivantes:

B: type de variable (B=1 pour une variable booléenne).
V: Valeur de positionnement (mise à 1, mise à 0).
Im: Type d'action: continue (Im=0) ou impulsionnelle (Im=1).
In: Indique s'il s'agit d'une variable interne (In=1) ou de sortie (In=0).

Dans le cas de positionnement de variables numériques, les indicateurs Id1 et Id2, identifiant respectivement le type de variable et la valeur à lui affecter, sont identiques à ceux utilisés pour les fonctions combinatoires.

Application du combinatoire:

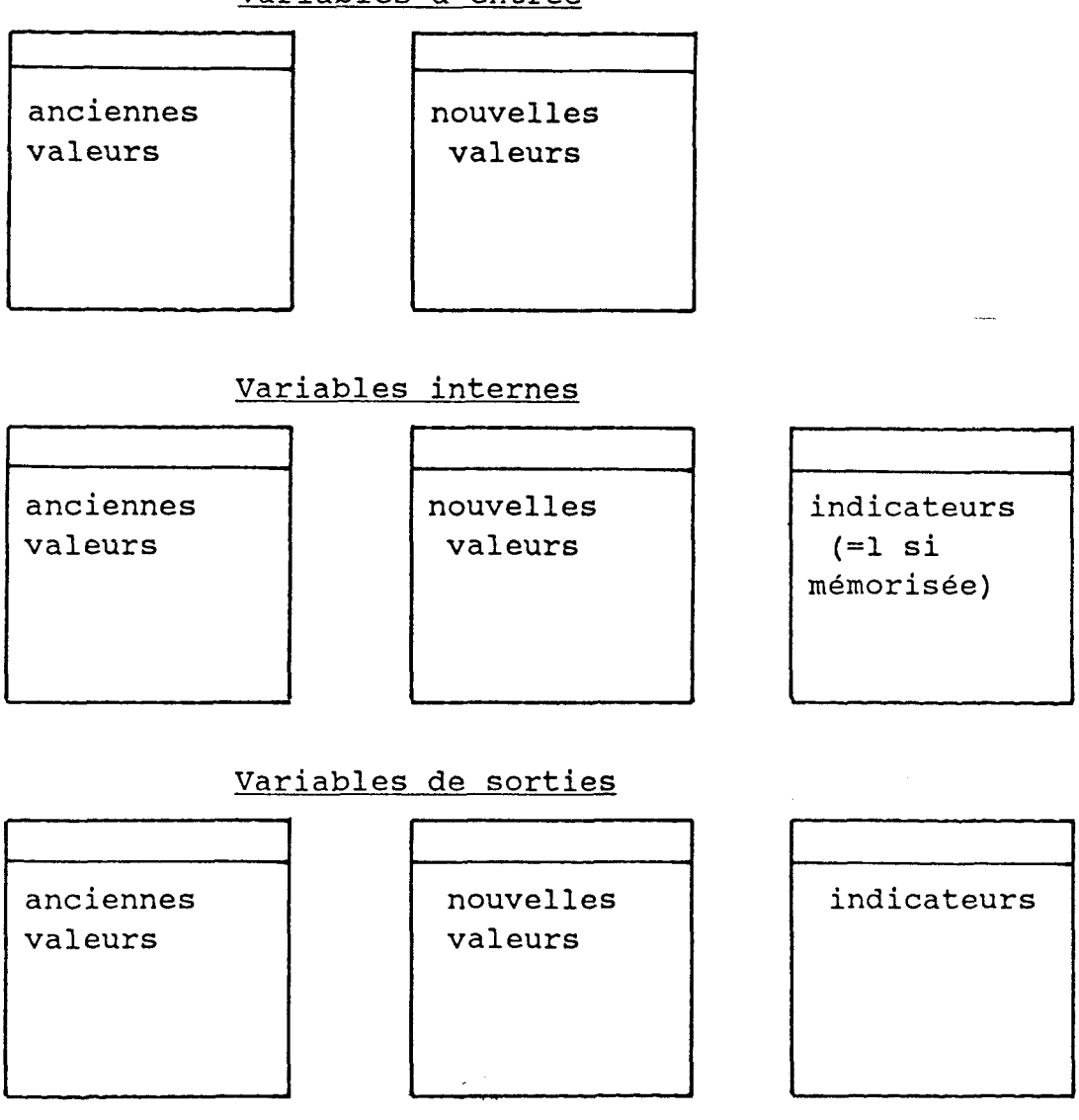
L'application du combinatoire commence par une remise à zéro de toutes les variables internes et de sortie sauf celles qui sont mémorisées, lesquelles restent inchangées.

Pour ce faire, nous avons besoin de plusieurs tables pour la mémorisation des différentes variables et indicateurs (Figure III.17).

Le combinatoire est élaboré dans les tables contenant les anciennes valeurs des variables internes et de sortie. Les autres tables sont figées pendant tout le cycle de traitement. Les rôles de ces tables sont intervertis à la fin du cycle.

Remarque:La prise en compte des fronts sur les variables internes dans la définition des conditions associées aux actions est impossible.

Figure III.17: Mémorisation des différentes variables.



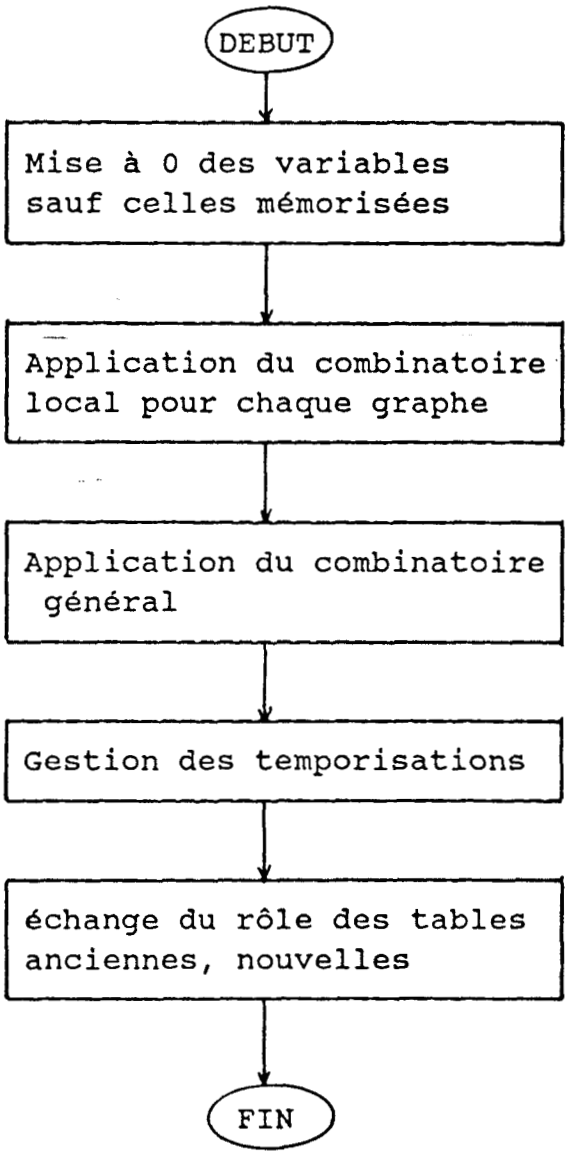
L'organigramme de la figure III.18 montre la procédure utilisée pour l'application du combinatoire.

La table des pointeurs actuels des graphes d'état permet d'accéder à l'adresse du combinatoire local des étapes actives en cours de chaque graphe. L'adresse du combinatoire général est donnée ailleurs.

Les actions de positionnement à une valeur donnée sont éxécutées dans les cas suivants:

- Pour une action impulsionnelle il faut que l'étape associée vienne d'être activée (ancien et nouveau pointeurs du graphe différents) et que la condition éventuellement associée soit vraie.
- Pour une action continue ou à niveau il faut que la condition éventuellement associée soit vraie.

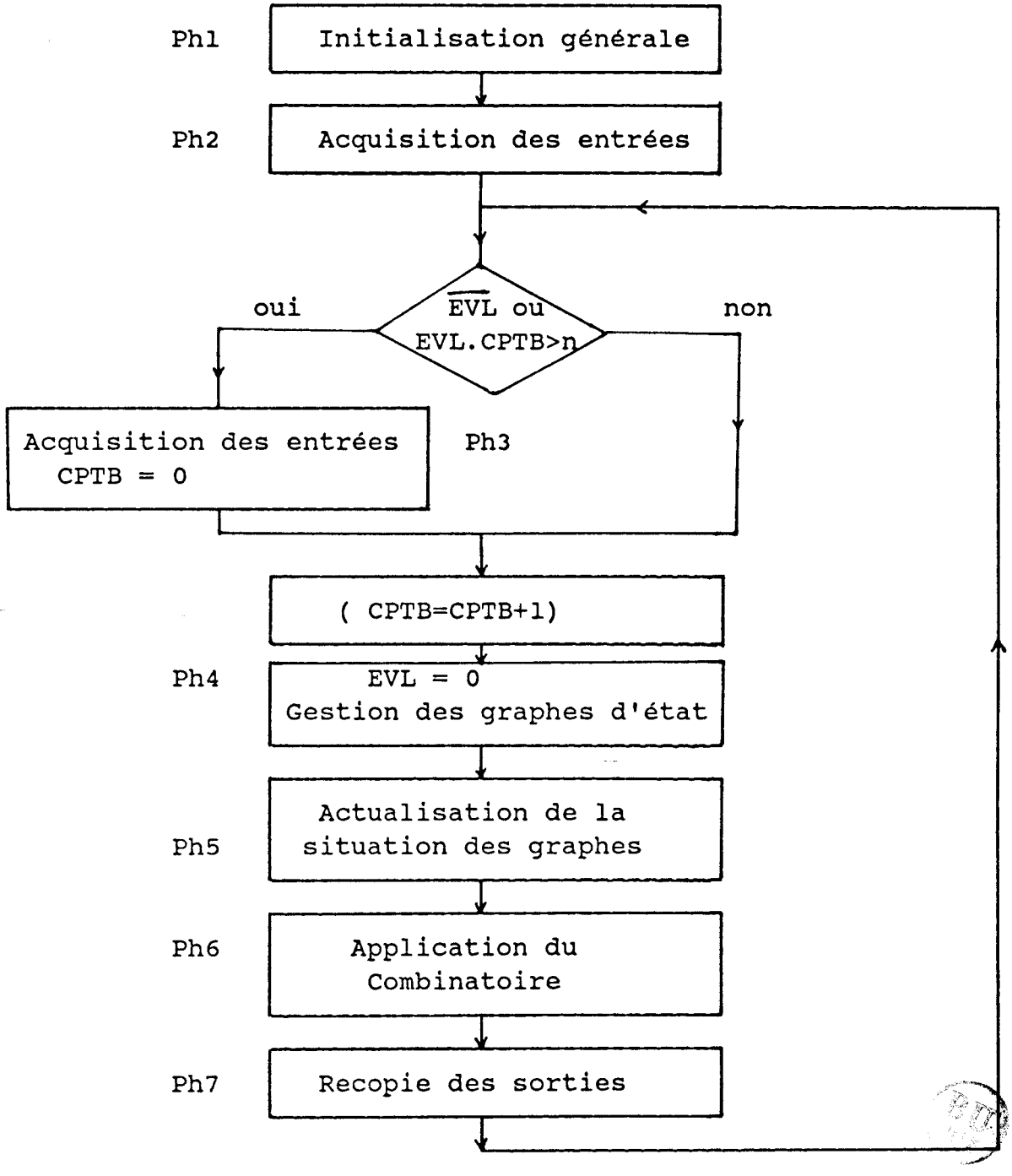
Figure III.18: Application du combinatoire.



3.4. Programme général de traitement

La gestion d'un ensemble de graphes d'état est réalisée suivant l'algorithme de la figure III.19.

Figure III.19: Organigramme de traitement général.



Nous utilisons un algorithme avec recherche de stabilité mais où le nombre de cycles d'évolution est limité à une valeur choisie n afin d'éviter des situations stationnaires. Ceci est réalisé grace à un compteur de boucles (CPTB). Un indicateur EVL d'évolution est positionné à 1 chaque fois qu'un des graphes d'état évolue. Les traitements réalisés au cours des différentes phases sont les suivants.

Phase 1: Initialisations:

- Cette initialisation consiste à rendre actives les étapes initiales des grahes d'état. Ceci est réalisé en mettant à zéro l'ancienne table des pointeurs et à recopier la table des pointeurs initiaux dans celle des pointeurs actuels.
- Toutes les variables internes et de sortie sont remises à zéro (anciennes et nouvelles valeurs).
- L'indicateur EVL d'évolution est positionné à 0.
- CPTB est remis à zéro pendant cette phase.

Phase 2: Acquisition des entrées au démarrage du système:

L'acquisition des entrées est toujours réalisée dans la table contenant les anciennes valeurs. Les rôles de ces tables sont intervertis en fin d'acquisition.

Phase 3: Acquisition des entrées et remise à zéro de CPTB:

Cette phase n'est éxécutée que s'il n'y a pas eu évolution ou que la valeur de CPTB est supérieure au nombre maximum de cycles permis sans acquisition.

Phase 4: Gestion des Grafcet:

La gestion des graphes d'état commence par une remise à zéro de l'indicateur EVL. Celui-ci est positionné à 1 dès qu'un des graphes évolue.

Phase 5: actualisation de la situation des graphes:

L'actualisation consiste à changer simplement les rôles des tables de pointeurs.

Phase 7: Recopie des sorties:

La recopie des sorties vers les interfaces est réalisée à chaque cycle même s'il n'y a pas stabilité de la situation.

4. LANGAGE DE DESCRIPTION DE LA PARTIE OPERATIVE

Nous proposons dans ce paragraphe, un langage de description et de spécification du comportement de la partie opérative afin de générer la structure de données de l'application, en l'occurence celle décrite au paragraphe 3 précédent.

Pour une meilleure portabilité du logiciel et afin de simplifier au maximum la tâche de l'utilisateur, nous nous sommes orienté vers un langage déclaratif plutôt que vers un procédé de questions-réponses qui devient vite fastidieux.

4.1. Principes généraux de la description

Les logiciels d'interprétation de la description et de génération de la structure de données ont été développés en langage Forth sur un micro-ordinateur MICRAL 90-50 sous CPM-86. Nous avons utilisé au mieux les possibilités de ce langage, de sorte que la syntaxe choisie pour le langage de description reprend certaines de ses caractéristiques en particulier la notion de mots et la notation polonaise inverse.

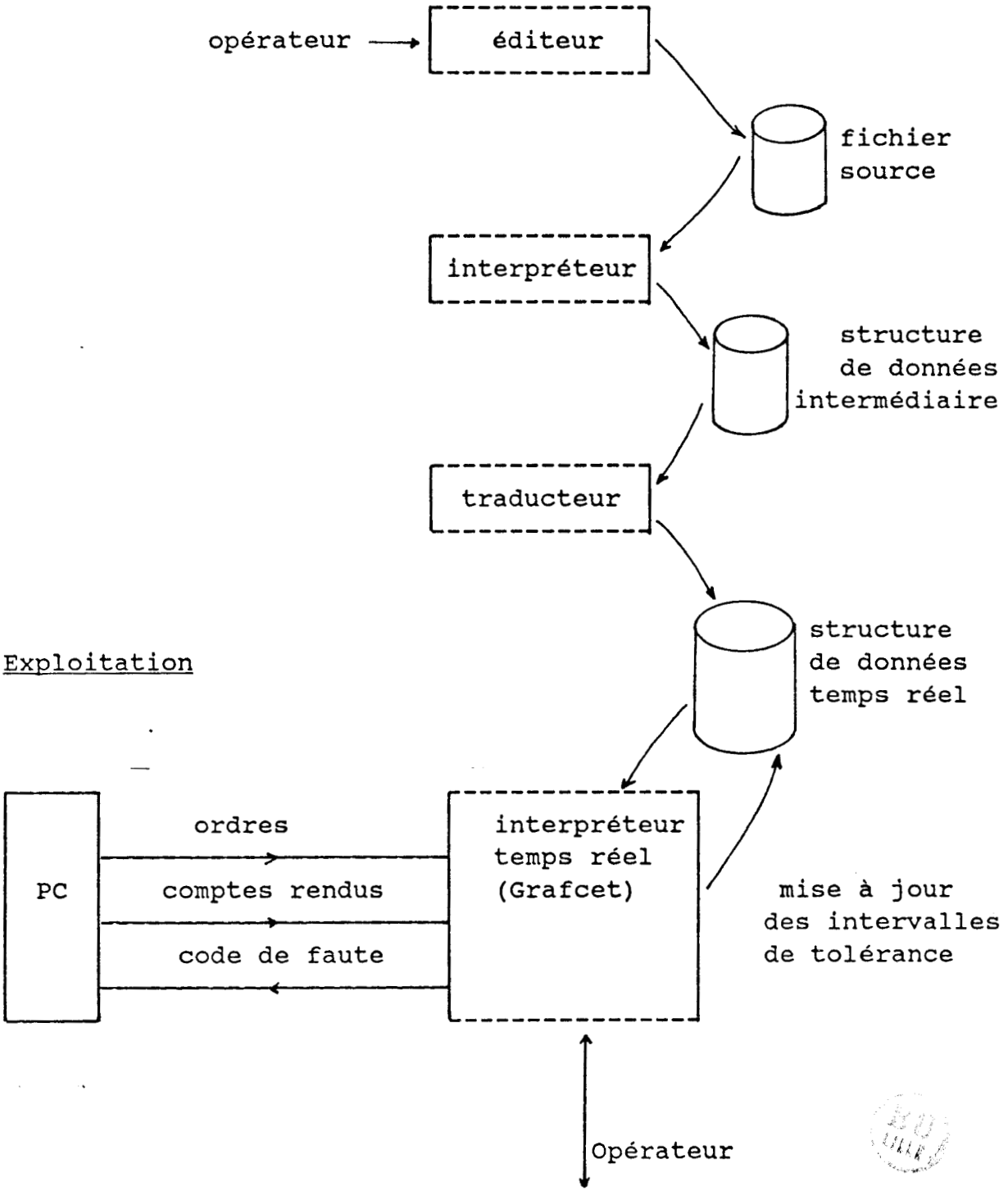
Le langage de description utilise dans sa syntaxe des mots qui sont définis dans le dictionnaire forth et effectuent chacun un certain traitement.
Ce traitement consiste en la création d'une structure de données intermédiaire où sont mémorisés au fur et à mesure de l'interprétation tous les renseignements sur le modèle de la partie opérative (Figure III.20).

4.2. Syntaxe de description: règles générales

La description commence par le mot DEBUT et se termine par le mot FIN. La rencontre du mot FIN déclence la traduction et génère la structure de données qui sera utilisée en temps réel. Le seul séparateur entre mots est le blanc ou espace. Tout commentaire peut être indiqué entre parenthèses séparées par au moins un espace.

Les identificateurs doivent avoir moins de 31 lettres, chiffres ou signes et ne pas commencer par un chiffre.
Les mots en majuscules et minuscules désignent des mots différents.

Figure III.20: Procédure générale de mise en ouvre.



4.3. Déclaration des variables

La déclaration des différentes variables d'entrée, internes et de sortie consiste à leur attribuer un nom, une adresse logique et définir leurs caractéristiques.

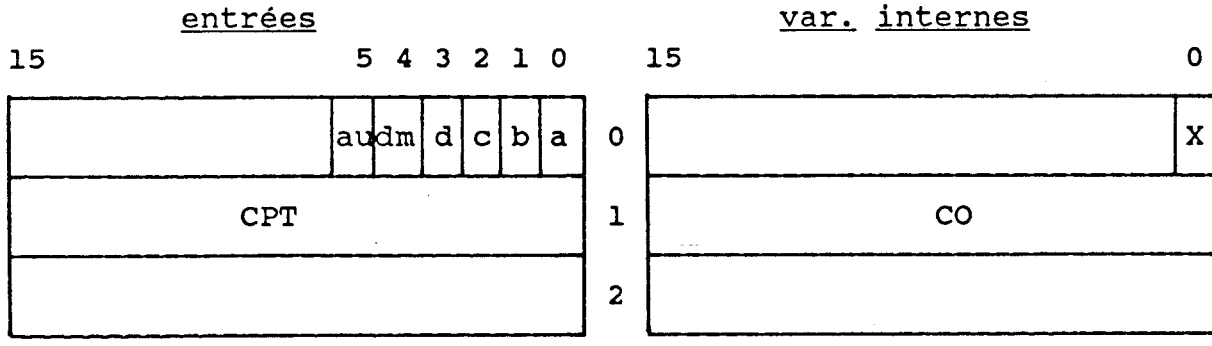
Nous avons prévu les cas suivants:

Syntaxe	Observations
<N° > DEFVARE <nom>	Variable d'entrée booléenne
<N° > DEFVAREN <nom>	Variable d'entrée numérique
<N° > NMEM DEFVARI <nom>	Var. interne bool. non mémorisée
<N° > MEMO DEFVARI <nom>	Var. interne bool. mémorisée
<N° > NMEM DEFVARIN <nom>	Var. interne num. non mémorisée
<N° > MEMO DEFVARIN <nom>	Var. interne numérique mémorisée
<N° > NMEM DEFVARS <nom>	Var. de sortie bool. non mémorisée
<N° > MEMO DEFVARS <nom>	Var. de sortie bool. mémorisée
<N° > NMEM DEFVARSN <nom>	Var. de sortie num. non mémorisée
<N° > MEMO DEFVARSN <nom>	Var. de sortie num. mémorisée

Exemples de déclaration

```
0  DEFVARE  a        1  DEFVARE  b        0  MEMO  DEFVARI  X
1  DEFVAREN CPT      2  DEFVARE  c        3  DEFVARE  d
4  DEFVARE  au       5  DEFVARE  dm
1  DEFVARIN CO
```

Ce qui donne la disposition suivante pour les variables:



4.4. Description des fonctions combinatoires

Les fonctions combinatoires sont décrites sous forme d'une somme de produits sans parenthèses et entre accollades { }.

En plus des signes de comparaison, les signes suivants sont utilisés:

+ pour la somme logique.

| pour la complémentation.

^ pour la détection de front sur une variable.

Ces deux derniers mots sont placés avant la variable concernée. La détection de front sur une fonction n'est pas directement possible avec la syntaxe adoptée.

Lorsqu'il s'agit d'une comparaison pour des variables numériques un nom de variable doit être écrit en premier.

Le détection de front sur une variable numérique revient à une comparaison entre ancienne et nouvelle valeur.

On remarquera les espaces utilisés comme séparateurs dans les exemples suivants.

Exemples de déclaration

```
{ a + | b X + CPT < 20 }

{ ^ c d + CO >= CPT }   ( front montant de c: c_t.c̄_t-1 )

{ au + dm }

{ ^ CPT }               ( équivalent à CPT_t <> CPT_t-1 )

{ ^ | au }              ( front descendant de au: ā̄u_t.au_t-1 )
```

4.5. Descrition de la partie "préactionneurs"

La description de la partie "préactionneurs" a pour but d'obtenir le modèle de comportement des préactionneurs afin de reconstituer les commandes ou vitesses.

Chaque sous ensemble de préactionneurs associé à un actionneur est déclaré entre les mots PREACTIONNEURS et FIN-PREACT.

Différents cas de descriptions sont envisagés.

a) Cas d'une partie "préactionneurs" combinatoire:

Lorsqu'il y a une correspondance directe ordres-vitesses la syntaxe de déclaration est la suivante. On donne un nom à chaque sous partie "préactionneurs" ainsi qu'aux différentes vitesses.

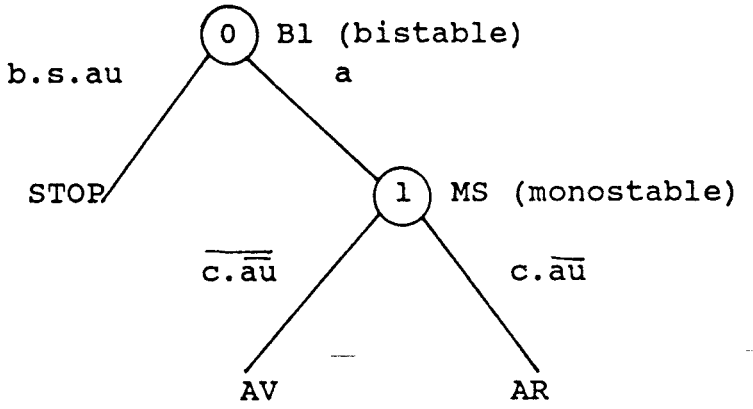
```
PREACTIONNEURS  <nom-préact>
      VITESSE   <expression booléenne>   <nom-vitesse>

      ( ... )

      VITESSE   <expression booléenne>   <nom-vitesse>
FIN-PREACT
```

b) Cas des préactionneurs séquentiels

Dans une première approche nous admettons que le circuit de puissance commuté par les préactionneurs peut se représenter par un arbre. Chaque préactionneur représente un noeud de l'arbre, les branches étant associées à ses positions.
La déclaration comporte les éléments suivants:

- Le numéro de l'élément de préactionneur. la numérotation commence par 0 et par la racine de l'arbre.
- Son type (monostable, bistable ...). ces types sont prédéfinis.
- Les fonctions de pilotage et l'affectation des positions ou sorties correspondantes.

Figure III.21: Représentation sous forme d'arbre.



La syntaxe de description est la suivante:

```
PREACTIONNEURS  <nom-préact>  SEQUENTIEL
    <N°>  PREACT-TYPE  <type>  { exp.bool }  <sortie>
                                    (  ...  )
                               { exp.bool }  <sortie>  ;

    (  - - - - - - - - - - - - - - - - - - - - - - - - - )

    <N°>  PREACT-TYPE  <type>  { exp.bool }  <vitesse>
                                    ( ... )
                               { exp.bool }  <sortie>  ;
FIN-PREACT
```

L'exemple de la figure III.21 est décrit de la façon suivante:

```
PREACTIONNEURS  pr1  SEQUENTIEL
    0   PREACT-TYPE   B1  ( b  s  au ) STOP  ( a )  1  ;
    1   PREACT-TYPE   MS  ( c | au  )  AR  ( | c  +  au  ) AV ;
FIN-PREACT
```

L'ordre dans lequel sont décrites les différentes branches doit correspondre à celui du type prédéfini en bibliothèque.

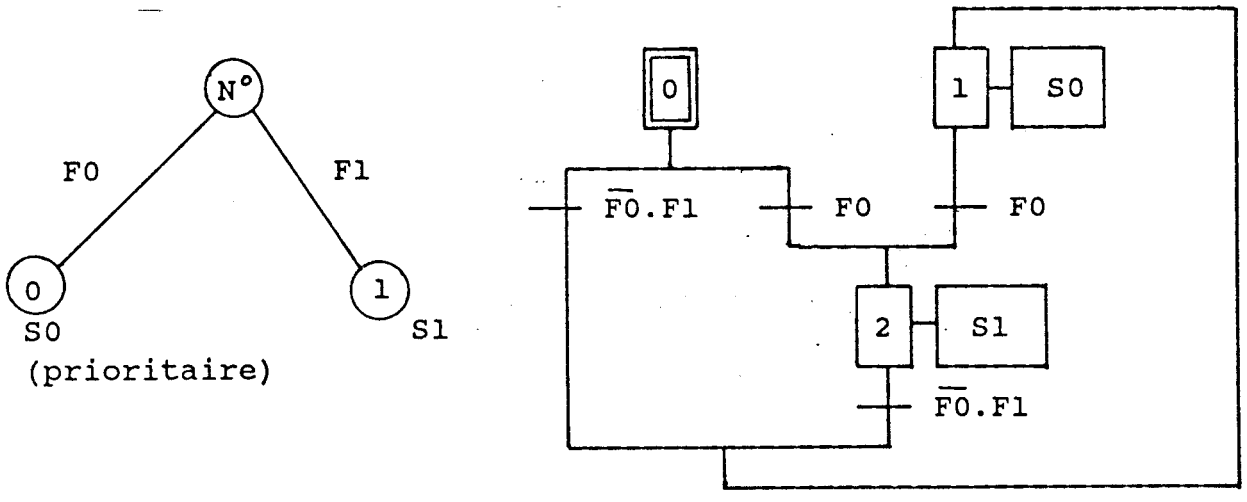
c) Introduction des types de préactionneurs en bibliothèque

Pour être introduit en bibliothèque, le préactionneur doit être décrit par un graphe d'état. Cette description doit être faite de façon à réaliser la synchronisation avec le modèle réel.

A titre d'exemple nous traitons le cas d'un préactionneur bistable (Figure III.22). Les fonctions F0, F1 et sorties S0, S1 sont ici des grandeurs muettes. L'équivalence est faite automatiquement à la traduction.

Figure III.22: Description d'un préactionneur en bibliothèque.

```
N°   PREACT-TYPE   B1   ( F0 )  S0  ( F1 )  S1  ;
```



La syntaxe suivante est utilisée pour décrire le graphe d'état.

```
nbt  nbe  nbp  DEF-PREACT  <nom>
       ne   FP  ns   TRANSITION  ( exp.bool )
       (  - - - - - - - - - - - - - - - - - - - )
       ne   s   ns   TRANSITION  ( exp.bool )
FIN-DEF
```

Avec nbt : le nombre total de transitions du graphe.
nbe : le nombre d'étapes.
nbp : le nombre de positions du préactionneur.
(Ces grandeurs permettent d'allouer la place mémoire nécessaire)
ne : le numéro de l'étape d'entrée de la transition.
s : le numéro de sortie associée à ne ou FP sinon.
ns : le numéro de l'étape de sortie de la transition.
La réceptivité associée à la transition est décrite non pas avec des symboles mais avec des numéros de fonctions.

Exemple: Description du préactionneur de la figure III.22

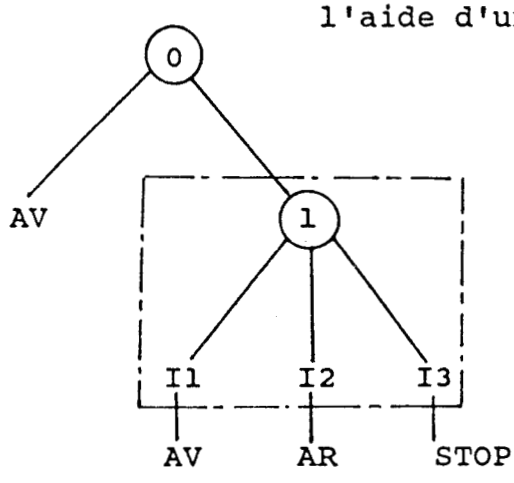
```
4  3  2  DEF-PREACT  B1
          0  FP  1  TRANSITION  {  |  0  1  }
          0  FP  2  TRANSITION  {  0  }
          1  0   2  TRANSITION  {  0  }
          2  1   1  TRANSITION  {  |  0  1  }
FIN-DEF
```

d) Description à l'aide d'un graphe d'état

L'ensemble de la partie "préactionneurs" peut être décrit par un graphe d'état lorsque celle-ci ne peut être spécifiée à l'aide des éléments de préactionneurs prédéfinis.

Cette description peut éventuellement porter uniquement sur un élément particulier qui n'est pas prédéfini (Figure III.23).

Figure III.23: Description d'un élément de préactionneur à l'aide d'un graphe d'état.



L'élément de préactionneur est d'abord décrit en utilisant comme sorties des variables internes (I1, I2, ...) à l'aide d'un éditeur de grafcet. La correspondance variables internes-vitesses est ensuite établie à la déclaration de la partie "préactionneurs" avec la syntaxe suivante.

```
PREACTIONNEURS <nom-préact> SEQUENTIEL
          0  PREACT-TYPE  B1  { | a }  AV   { a   } 1  ;
          1  PREACT-SP    I1   AV   I2   AR   I3   STOP  ;
FIN-PREACT
```

Lorsque l'ensemble de la partie "préactionneurs" est décrit par un graphe d'état la syntaxe suivante est utilisée.

```
PREACTIONNEURS  <nom-préact>  GRAPHE
     VITESSES  I1   <vitesse>  I2  <vitesse>  ...   ;
FIN-PREACT
```

Le graphe lui-même est déclaré avant, avec la syntaxe ci-après. Les variables internes utilisées comme sorties doivent être déclarées auparavant ainsi que les fonctions de récéptivité et les conditions associées aux actions.

```
DEFONCT  F1  { exp. bool. }     ( déclaration des fonctions    )
DEFONCT  C1  { exp. bool. }     ( de réceptivité et conditions )
nbe   DEFGRAF                     ( nbe=nombre total d'étapes )
    ne  f  ns   GTRANSITION        ( ne = étape d'entrée       )
    ( - - - - - - - - - - - )      ( f = récéptivité           )
    ne  F1  ns  GTRANSITION        ( ns = étape de sortie      )
    FINTRANSITION

val var CONT   cond ne COMBL       ( var= variable concernée   )
                                   ( val = valeur à affecter   )
val var IMPUL   C1  ne COMBL       ( cond= condition associée  )
                                   ( prédéfinie, FP sinon      )
    FINCOMB                        ( CONT, IMPUL: action       )
FINGRAF                            ( continue, impulsionnelle  )
```

La numérotation des étapes doit commencer par 0 pour l'étape initiale qui est ainsi reconnue automatiquement. Un exemple de description est donné dans le paragraphe 4.7.

4.6. Description de la partie actionneurs/capteurs

La description de la partie actionneurs/capteurs doit spécifier les caractéristiques des points singuliers (comptes rendus associés et éventuellement partage de capteurs) ainsi que les trajectoires et les vitesses dans chaque sens.

La déclaration de chaque sous partie actionneurs/capteurs est encadrée par les mots **ACTIONNEUR** et **FIN-ACT**. Cette description suit celle de la sous partie "préactionneurs" correspondante.

a) Déclaration des points singuliers:

Chaque point singulier se voit attribuer un nom. La syntaxe de description générale d'une partie actionneurs/capteurs est la suivante.

```
ACTIONNEUR <nom-act>

   PT-SING  { compte rendu associé }  <nom-ps>  ...  ;
   PT-SING  { compte rendu associé }  <nom-ps>  ...  ;  COMMUN

   TRAJET  <ps1>  ...  <psn>  ;
   TRAJET  <psj>  ...  <psk>  ;

   ALLER   <vitesses aller>      ...    ;
   RETOUR  <vitesses retour>     ...    ;

FIN-ACT
   PARTAGE  <act1>  <act2>
```

Tous les points singuliers ayant le même compte rendu associé, sont déclarés ensemble de façon à détecter implicitement les comptes rendus déterministes. Lorsqu'il y a partage de trajectoire, le PS commun est suivi par le mot COMMUN.

Le compte rendu associé à chaque PS est défini sur l'alphabet des capteurs atteignables par la grandeur mesurée. Ce compte rendu est en général un produit. Les variables doivent être déclarées toujours dans le même ordre dans chaque CR afin de simplifier la recherche des capteurs communs à plusieurs grandeurs. Ce partage de capteurs est déclaré avec le mot PARTAGE à un endroit quelconque de la description.

Exemple de déclaration:

(Une came se déplace devant 3 capteurs c1, c2, C3)

```
ACTIONNEUR   came
     PT-SING  { | c1 | c2  | c3  }    PS2  PS4  ;
     PT-SING  {   c1 | c2  | c3  }    PS1  ;
     PT-SING  { | c1   c2  | c3  }    PS3  ;
     PT-SING  { | c1 | c2    c3  }    PS5  ;
   TRAJET  PS1  PS2  PS3  PS4  PS5  ;
     ALLER  VA1  VA2  ;
     RETOUR  VR  ;
FIN-ACT
```

b) Déclaration des trajectoires:

Lorsqu'il s'agit d'une trajectoire simple, la déclaration à l'aide du mot TRAJET énumère la suite des points singuliers atteignables dans le sens arbitraire "aller".

Si la trajectoire à décrire comporte des bifurcations ou des interactions entre trajectoires, celle-ci est décomposée en "trajets" simples avec éventuellement des conditions d'accès à ces tronçons (condition d'entrée à l'aller et au retour). Ces conditions sont exprimées à l'aide de la position de capteurs.

Le logiciel d'interprétation reconstitue la trajectoire par concaténation en fusionnant les points singuliers de même nom aux extrémités des tronçons.

La syntaxe de description est alors la suivante.

```
TRAJET  <ps1>  <ps2>   ...   <psn>  ;  SI  { conda }  { condr }
TRAJET  <psn>  <psj>   ...   <psk>  ;  SI  { NIL   }  { condr }
```

La condition "conda" porte donc sur l'évolution à partir de ps1 vers ps2 alors que "condr" conditionne l'évolution à partir du dernier PS vers l'avant dernier.

Des exemples de trajectoires sont décrits dans le paragraphe ci-après.

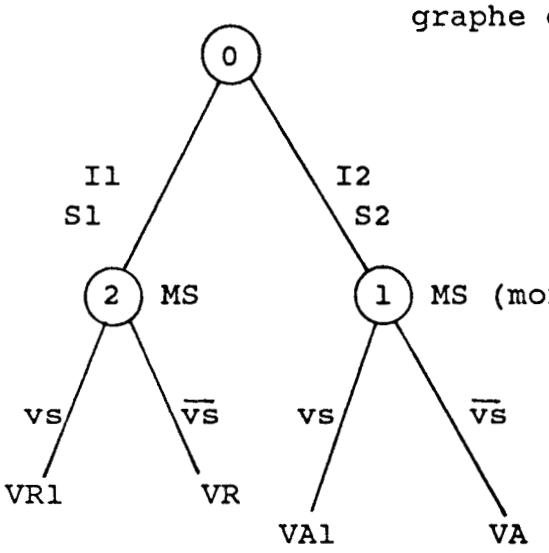
4.7. Exemples de description de parties opératives

a) Exemple N°1:

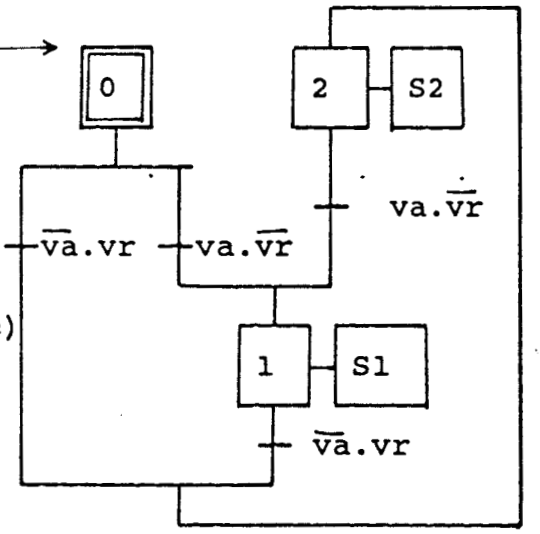
Soit un chariot se déplaçant suivant la trajectoire représentée figure III.24.c. La partie "préactionneurs" est décrite à la figure III.24.a et b. (l'aiguillage n'est pas décrit ici).

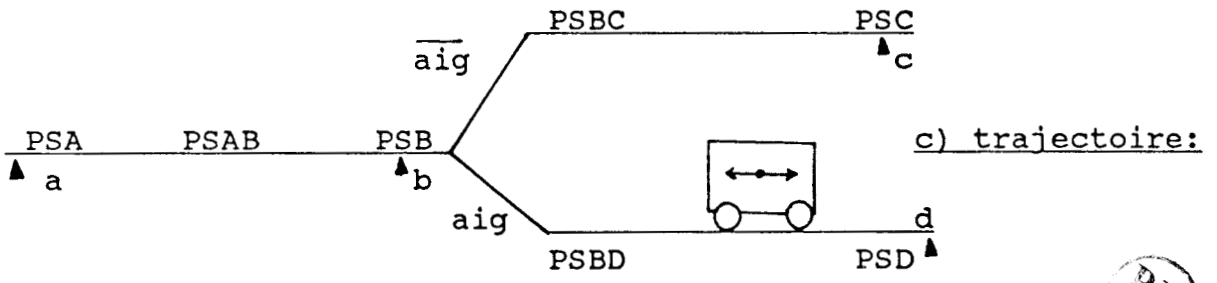
Figure III.24: Evolution d'un chariot sur une voie ferrée avec bifurcation commandée par un aiguillage.

a) Partie "préactionneurs":



b) graphe





c) trajectoire:

Disposition des variables d'entrée

	aig	d	c	b	a
			vs	vr	va

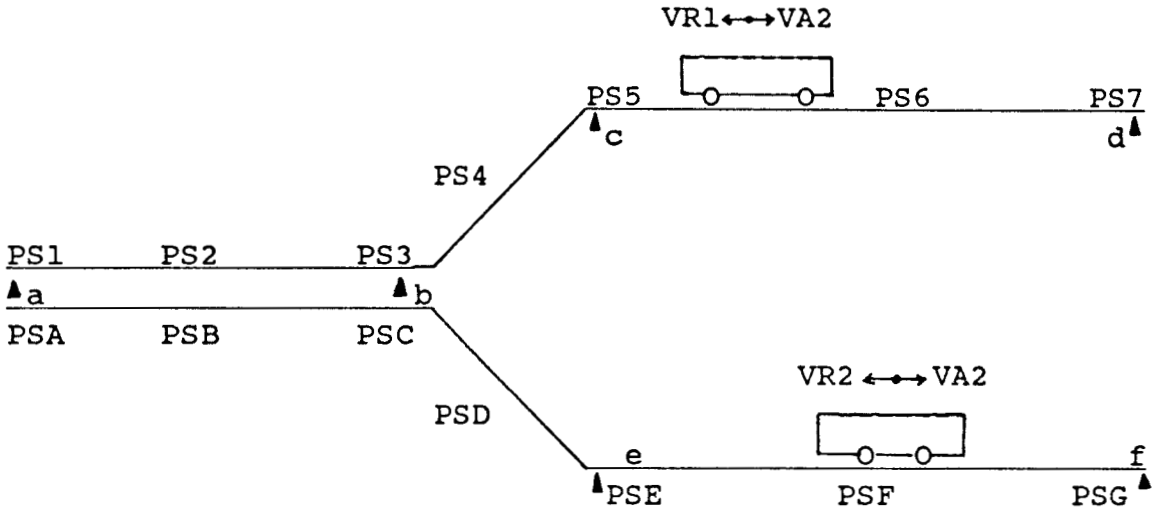
La déclaration de la partie opérative est alors la suivante:

```
DEBUT
    0  DEFVARE  a        1  DEFVARE  b       2  DEFVARE  c
    3  DEFVARE  d        4  DEFVARE  aig    16  DEFVARE  va
   17  DEFVARE  vr      18  DEFVARE  vs
   DEFONCT  FC1  ( va | vr )
   DEFONCT  FC2  ( | va vr )
     0  NMEM  DEFVARI  S1
     1  NMEM  DEFVARI  S2
   3  DEFGRAF
        0  FC1  1  GTRANSITION      0  FC2  2  GTRANSITION
        1  FC2  2  GTRANSITION      2  FC1  1  GTRANSITION
        FINTRANSITION
          1  S1  CONT  FP  1  COMBL
          1  S2  CONT  FP  2  COMBL
        FINCOMB
  FINGRAF
  ( description de l'aiguillage non effectuée ici )
  PREACTIONNEURS  pmr  SEQUENTIEL
        0  PREACT-SP   S1  0  S2  1  ;
        1  PREACT-TYPE MS  ( vs )  VA1  ( | vs )  VA  ;
        2  PREACT-TYPE MS  ( vs )  VR1  ( | vs )  VR  ;
  FIN-PREACT
  ACTIONNEUR  mr
        PT-SING  ( | a | b | c | d )  PSAB  PSBC  PSBD  ;
        PT-SING  (   a | b | c | d )  PSA  ;
        PT-SING  ( | a   b | c | d )  PSB  ;
        PT-SING  ( | a | b   c | d )  PSC  ;
        PT-SING  ( | a | b | c   d )  PSD  ;
      TRAJET  PSA  PSAB  PSB ;
      TRAJET  PSB  PSBD  PSD ;  SI  (   aig )  NIL
      TRAJET  PSB  PSBC  PSC ;  SI  ( | aig )  NIL
        ALLER  VA  ;
        RETOUR VR  ;
  FIN-ACT
FIN
```

b) Exemple N°2:

La partie opérative à décrire est représentée à la figure III.25. Les trajectoires suivies par les 2 chariots possèdent une partie commune. (la sous PO constituée par l'aiguillage n'est pas décrite).

Figure III.25: Cas d'un partage de trajectoire.



La partie "préactionneurs" est de type combinatoire: les vitesses sont obtenues de la façon suivante pour les deux chariots:

vr \ va	0	1
0	Stop	VA
1	VR	Stop

La decription correspondante est la suivante:

```
DEBUT
   0  DEFVARE  a        1  DEFVARE  b       2  DEFVARE  c
   3  DEFVARE  d        4  DEFVARE  e       5  DEFVARE  f
  16  DEFVARE  val     17  DEFVARE  vrl    18  DEFVARE  va2
  19  DEFVARE  vr2

   ( aiguillage non décrit ici )

   PREACTIONNEURS  PMR1
       VITESSE  {   val | vrl }  VA1
       VITESSE  { | val   vrl }  VR1
   FIN-PREACT
   PREACTIONNEURS  PMR2
       VITESSE  {   va2 | vr2 }  VA2
       VITESSE  { | va2   vr2 }  VR2
   FIN-PREACT
   ACTIONNEUR  MR1
       PT-SING  { | a | b | c | d }  PS2 PS4 PS6 ;
       PT-SING  {   a | b | c | d }  PS1 ; COMMUN
       PT-SING  { | a   b | c | d }  PS3 ; COMMUN
       PT-SING  { | a | b   c | d }  PS5 ;
       PT-SING  { | a | b | c   d }  PS7 ;
     TRAJET  PS1  PS2  PS3  PS4  PS5  PS6  PS7 ;
     ALLER  VA1  ;
     RETOUR VR1  ;
   FIN-ACT
     PARTAGE  MR1  MR2
   ACTIONNEUR  MR2
       PT-SING  { | a | b | e | f }  PSB  PSD  PSF ;
       PT-SING  {   a | b | e | f }  PSA ; COMMUN
       PT-SING  { | a   b | e | f }  PSC ; COMMUN
       PT-SING  { | a | b   e | f }  PSE ;
       PT-SING  { | a | b | e   f }  PSG ;
     TRAJET PSA  PSB  PSC  PSD  PSE  PSF  PSG  ;
     ALLER  VA2  ;
     RETOUR VR2  ;
   FIN-ACT
FIN
```

5. GENERATION DE LA STRUCTURE DE DONNEES TEMPS REEL ET EXPLOITATION

5.1. Génération de la structure de données

L'interprétation de la description de la partie opérative a permis de générer une structure de données intermédiaire contenant toutes les caractéristiques de l'application (fig.III.20).

La deuxième étape consiste à traduire ces données en une structure de données sous forme d'un ensemble de graphes d'état laquelle sera gérée en temps réel par l'interpréteur de Grafcet.

Les logiciels d'interprétation et de traduction, assez complexes, ne seront pas décrits dans cet exposé. (voir [67]).

5.2. Exploitation temps réel

En phase d'exploitation du dispositif de test, l'utilisateur a la possibilité de programmer un certain nombre de paramètres concernant:

- les temporisations,
- la constitution de l'historique,
- la gestion des codes de faute.

a) Les temporisations:

l'utilisateur peut visualiser et programmer les valeurs des temporisations pour chaque action élémentaire avec ou sans changement de sens de déplacement. Il peut aussi autoriser ou non leur remise à jour.

La remise à jour n'est effectuée que si l'action élémentaire s'est correctement déroulé et le sens de l'action élémentaire précédente est connu. On recalcul alors les nouvelles valeurs de la moyenne et de l'écart type en fonction des anciennes valeurs, de la nouvelle mesure faite (m) et du nombre total de mesures Nbm. Les nouvelles valeurs de Tmin et Tmax sont déduites en tenant compte du facteur de confiance alpha fixé globalement. Les formules utilisées pour ces calculs se présentent comme suit:

$Nbm = Nbm + 1$

$SOMX = SOMX + m$

$SOMX2 = SOMX2 + m^2$

$MOY = SOMX/Nbm$

$TAU = SQRT((SOMX2-(SOMX^2/Nbm))/(Nbm-1))$

$Tmin = MOY - alpha * TAU$

$Tmax = MOY + alpha * TAU$

Cette procédure a l'inconvéniant de ne tolérer aucune dérive des intervalles de tolérance même si celle-ci peut être considérée comme normale. Une première solution pour tolérer une certaine dérive consiste à réajuster le facteur de confiance alpha. Le calcul d'une moyenne sur une fenêtre glissante a l'inconvéniant de nécessiter l'enregistrement des mesures. Mais une solution de compromis peut être trouvée si à chaque calcul on enlève alternativement une quantité égale à: $MOY_k + TAU_k$. Les calculs suivants sont alors effectués avant la remise à jour des temporisations décrite précédemment.

$$SOMX = SOMX - (MOY_k + TAU_k)$$
$$SOMX2 = SOMX2 - (MOY_k + TAU_k)^2$$
$$Nbm = Nbm - 1$$

b) Constitution de l'historique:

La capacité mémoire étant en général limitée, l'opérateur a la possibilité de moduler la constitution de l'historique en programmant les parties actionneurs/capteurs pour lesquelles les messages doivent être enregistrés. Il peut également ne conserver que les messages d'erreur.

La visualisation de l'historique ne peut être effectuée qu'après arrêt de la surveillance.

c) les codes de faute:

l'utilisateur, en dehors de la possibilité de programmer un code de faute par action élémentaire et par type de défaut détecté, peut soit activer ou non la transmission de ce code vers la partie commande. La surveillance peut ainsi fonctionner en mode passif.

Conclusion

La méthodologie d'implantation du mécanisme de test en ligne de la partie opérative qui a été adoptée, a nécessité la redéfinition du modèle de détection en utilisant le formalisme du Grafcet. Ceci permet l'utilisation d'outils non spécifiques.

Le choix d'une structure de réalisation orientée données des grafcet a l'avantage de simplifier la génération du modèle de l'application pour l'utilisateur.

Le langage de spécification proposé pour la description du modèle de comportement de la partie opérative est relativement simple. Une procédure d'apprentissage automatique du modèle de la PO sera envisagée dans les chapitres suivants.

CHAPITRE IV: APPRENTISSAGE AUTOMATIQUE DU MODELE DE LA PO.

CHAPITRE IV

APPRENTISSAGE AUTOMATIQUE DU MODELE DE LA PARTIE OPERATIVE

L'apprentissage automatique du modèle de comportement entrée/sortie de la partie opérative à partir d'observations des seuls ordres et comptes rendus pose un certain nombre de problèmes difficiles à résoudre dans le cas général. Ces difficultés viennent de ce que les commandes ne sont pas observables.

Notre but est en effet de retrouver le modèle de comportement tel que décrit au chapitre 2 pour pouvoir l'utiliser pour le test en ligne de la partie opérative.

Nous étudions dans ce chapitre les possibilités d'identification des modèles de comportement de la partie actionneurs/capteurs et de la partie "préactionneurs" à partir de l'observation des seuls ordres et comptes rendus.

1. APPROCHE GENERALE DU PROBLEME D'APPRENTISSAGE

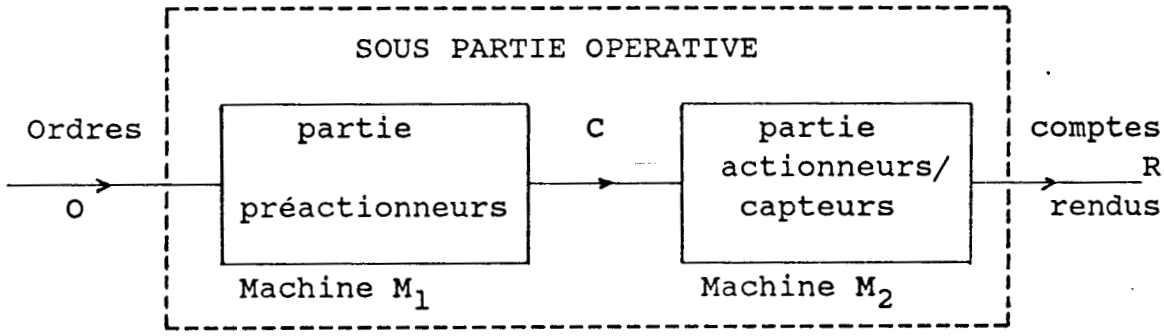
1.1. Position du problème:

L'ensemble de la PO peut être modélisé par une machine séquentielle M: $O^+ \rightarrow R^+$ et identifiée en temps que telle. Si cette procédure permet de surmonter les difficultés liées à la non observabilité des commandes et des interactions entre trajectoires des sous processus, elle a l'inconvénient d'aboutir à un modèle trop complexe pour être exploitable à cause de l'explosion combinatoire occasionnée par l'évolution en parallèle des sous processus. Nous conserverons donc la décomposition en sous PO opérée précédemment.

Le modèle de comportement recherché pour chaque sous partie opérative est constitué de deux machines (fig.IV.1):

- la partie "préactionneurs" considérée ici, d'une façon générale, comme une machine séquentielle M_1: $O^+ \rightarrow C^+$ mais où les sorties, constituées par les commandes C, ne sont pas directement observables.
- En dehors des temps d'évolution, la partie "actionneurs/capteurs" est caractérisée par la trajectoire suivie par la grandeur mesurée ou les séquences de comptes rendus possibles sous l'action des commandes appliquées.

Figure IV.1: modèles recherchés.



1.2. Approche générale:

L'approche générale envisagée consiste à déterminer successivement, pour une sous partie opérative donnée, les modèles suivants:

- l'identification de la trajectoire (géométrie et comptes rendus associés aux PS) à partir d'une séquence de comptes rendus et éventuellement des ordres appliqués.
- l'utilisation de ce modèle de trajectoire et la mesure des temps d'évolution permet de réaliser un apprentissage des classes de commande.

- La connaissance du modèle dynamique de la partie actionneurs/capteurs va permettre d'identifier les commandes C possibles à partir d'une séquence O/R/t avec les mesures de temps entre deux changements consécutifs d'ordre et/ou de compte rendu.

- L'obtention d'une séquence O/p(C) probabiliste va alors permettre l'identification de la machine séquentielle constituée par la partie "préactionneurs". Ce problème d'identification de machine séquentielle à partir d'observations incertaines sera traité au chapitre 5.

Certaines interactions entre trajectoires des grandeurs mesurées ou sous parties opératives sont particulièrement génantes pour l'identification. Dans une première étape, nous admettrons l'hypothèse suivante:

Hypothèse 1:
Il n'existe pas de capteurs partagés entre les différentes sous parties opératives.

2. IDENTIFICATION DE LA TRAJECTOIRE

La détermination de la géométrie de la trajectoire et des comptes rendus associés à chaque point singulier est faite à partir d'une séquence O/R observée sur la partie opérative.

La relation entre comptes rendus et points singuliers n'étant pas biunivoque, retrouver les points singuliers et leur agencement à partir d'une séquence de comptes rendus est un problème théoriquement insoluble. Néamoins, la prise en compte d'autres informations comme la séquence d'ordre, ou certaines hypothèses simplificatrices permettent de se rapprocher du modèle réel.

Afin de mieux éclaircir le type de modèle de trajectoire recherchée, nous analysons les caractéristiques géométriques de celles-ci à l'aide de la théorie des graphes [60].

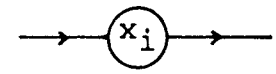
2.1 . Caractéristiques géométriques des trajectoires

Dans le graphe G = (X,U) représentant la trajectoire orientée suivant un sens de parcours positif arbitraire, on peut distinguer les différents types de sommets suivants.

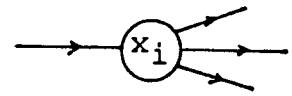
On note:
X: l'ensemble des sommets x_i représentant les points singuliers
U: l'ensemble des arcs orientés (x_i, x_j).
$d_G(x_i)$: le degré du sommet x_i ou le nombre d'arcs ayant x_i comme origine ou extrémité.
$d^+_G(x_i)$: représente le nombre d'arcs ayant x_i comme extrémité.
$d^-_G(x_i)$: est le nombre d'arcs ayant x_i pour origine.

a) sommet extrémité: 

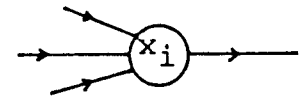
Un sommet du graphe est une extrémité lorsque celui-ci est de degré $d_G(x_i) = 1$.

b) sommet de liaison: 

C'est un sommet ayant un seul arc incident vers l'intérieur et un seul arc incident vers l'extérieur: $d^-_G(x_i) = d^+_G(x_i) = 1$.

c) sommet divergent: 

Un sommet divergent est tel que $d^-_G(x_i) \leq 1$ et $d^+_G(x_i) > 1$. Il représente une bifurcation.

d) sommet convergent: 

Un sommet convergent est caractérisé par $d^+_G(x_i)<1$ et $d^-_G(x_i)>1$.

La trajectoire orientée suivant un sens positif arbitraire est ainsi représentée par un graphe connexe semi-fonctionnel.

Remarque: les arcs tels que (q_i, r_{ii}, q_i) ne sont pas représentés.

2.2. Apprentissage par inférence grammaticale [23], [24]

2.2.1. Position du problème: cas général

Soit un échantillon I de phrases d'un certain langage. On définit les notions d'automate canonique maximal et d'automate dérivé.

Automate Canonique Maximal (ACM):
L'automate canonique maximal correspondant à un échantillon I est construit directement à partir des phrases de cet échantillon. Le langage accepté par cet automate est égal à I.

exemple:

L'automate canonique maximal correspondant à l'échantillon I=(abc, abde, dbc) est représenté ci-dessous: (Fig.IV.2).

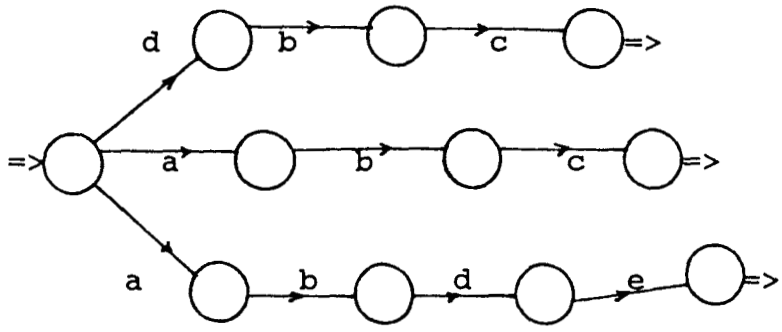


Figure IV.2

Automate dérivé:

C'est un automate fini construit en confondant certains états de l'automate initial. Si **A'** est l'automate dérivé de **A** pour une partition quelconque des états de **A** nous avons:

$L(A') \subset L(A)$

Echantillon complet:

Un échantillon I est dit structurellement complet pour une grammaire $G = (V_N, V_T, R, S)$ si:

a) $I \subset L(G)$
b) L'alphabet sur lequel est écrit I est égal à V_T
c) Toutes les règles de R ont été utilisées au moins une fois dans la génération de I.

Le problème d'inférence peut alors être exprimé ainsi:
Etant donné un échantillon I complet, trouver une grammaire G telle que:
- $I \subset L(G)$,
- I est complet par rapport à G.

Le nombre de solutions à ce problème est en général très grand. Pour une séquence échantillon de longueur 10, par exemple, le nombre de solutions est d'environ 10^5.

Une condition nécessaire et suffisante pour qu'un automate fini **A** soit solution du problème d'inférence est que:
- $I \subset L(A)$
- I complet par rapport à **A**
- et que **A** soit dérivé de l'automate canonique maximal.

Cette dérivation est à la base de la plupart des méthodes d'inférence parmi lesquelles on peut citer:

- la méthode des k-finales
- l'agrégation de finales

Il est clair qu'entre l'automate canonique maximal que l'on peut construire directement à partir de l'échantillon I observé et l'automate universel (pouvant correspondre au test statique) il existe un automate particulier et un seul correspondant au modèle recherché. L'exemple suivant illustre un des problèmes posés par la dérivation.

exemple:

Dans l'automate de la figure II.17.b les séquences de comptes rendus (r_2 r_0 r_3 r_0 r_1) et (r_2 r_0 r_1 r_0 r_2) mènent de q_0 à qf. La dérivation amène à considérer comme acceptables les séquences (r_2 r_0 r_1 r_0 r_1) et (r_2 r_0 r_3 r_0 r_2), or dans le modèle réel celles-ci ne mènent pas vers l'état final. L'élargissement du langage accepté par l'automate dérivé à pour conséquence dans notre cas la signalisation de fausses pannes.

Il sera donc nécessaire de définir des critères ou une heuristique afin de guider l'inférence pour retrouver le modèle de la trajectoire.

2.2.2. Règles d'inférence: [51]

Soit une séquence échantillon Sr de comptes rendus telle que Sr(t)<>Sr(t+1). Les conditions de complétude de l'échantillon se traduisent comme suit pour la partie opérative:

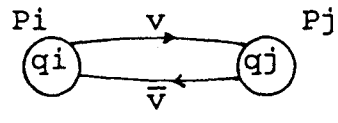
a) Sr est observée sur la partie opérative saine.

b) Sr contient toutes les combinaisons des capteurs observables en fonctionnement normal.

c) Tous les points singuliers de la trajectoire sont parcourus dans les deux sens au cours de la séquence Sr.

Hypothèse 2:

Si le passage d'un point singulier Pi à un autre Pj entraine la mise à "1" d'une variable booléenne, le passage de Pj à Pi force à "0" cette même variable. Ceci exclu donc les capteurs de sens de passage.



Sous forme de règles de production cette hypothèse peut être écrite sous la forme: Pi —> vPj <==> Pj --> $\bar{v}$Pi

Moyennant cette hypothèse, on réalise la transformation suivante de la séquence échantillon Sr.
On remplace chaque compte rendu par le nom de la variable qui à changé par rapport au compte rendu précédent. On obtient une suite contenant les noms des variables qui changent de valeurs sous forme affirmée si la valeur atteinte est "1" ou niée sinon.

Exemple:

Soit la séquence de CR: Sr = ($\bar{a}b\bar{c}$ $\bar{a}bc$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}\bar{b}c$ $\bar{a}\bar{b}\bar{c}$).
La transformation envisagée permet d'obtenir la séquence suivante: Sv=(b $\bar{b}$ c $\bar{c}$).

Le nombre d'éléments de Sv est égal au nombre de changements dans Sr, ce qui correspond aux nombre d'arcs (qi,r,qj) tels que qi <> qj parcourus pendant la période d'observation.

Suites antisymétriques:

Notons S(i) le $i^{ième}$ terme de S. Deux séquences S_1 et S_2 de même longueur n sont dites antisymétriques si pour tout i [1,n] on a $S_1(i) = \overline{S_2(n+1 - i)}$. Nous noterons la suite antisymétrique de S_1 par $(\overline{S_1})^{-1}$.
exemple: S_1 = a $\bar{b}$ c; S_2 = $\bar{c}$ b $\bar{a}$;

Ces suites permettent de détecter des changements potentiels de sens de parcours.

Suites englobées ou facteur gauche

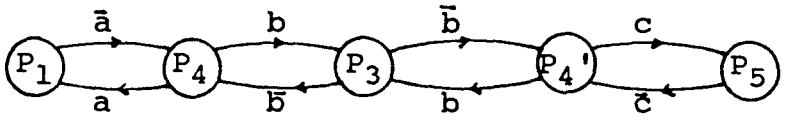
Une séquence S_1 de longueur n est dite englobée dans une séquence S_2 de longueur m tel que $m \geq n$ si:

$\forall$ i ϵ [1,n] : $S_1(i) = S_2(m-n + i)$.
exmple: S_1 = (a $\bar{b}$ b $\bar{a}$) englobée dans S_2 = (a $\bar{c}$ a $\bar{b}$ b $\bar{a}$)

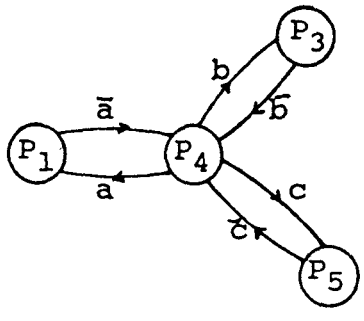
palindrome:
On est en présence d'un palindrome lorsque le parcours sur la trajectoire à partir d'un point singulier (sommet pivot) dans un sens ou dans l'autre, produit la même séquence de comptes rendus.

Exemple: la séquence de comptes rendus suivante Sr=(a$\bar{b}\bar{c}$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}bc$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}\bar{b}c$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}bc$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}bc$ $\bar{a}\bar{b}\bar{c}$) qui donne Sv= ($\bar{a}$ b $\bar{b}$ c $\bar{c}$ c $\bar{c}$ b $\bar{b}$ a $\bar{a}$) peut correspondre aux différentes trajectoires ci-dessous: (Fig.IV.3)

Figure IV.3:



palindrome
(P_3=sommet pivot)



Cette trajectoire peut être orientée de plusieurs manières différentes suivant le sens relatif dans chaque branche.

Echantillon propre:

L'échantillon sera dit **propre** s'il est complet et s'il a été observé pour une évolution telle que le nombre de changements de sens à l'intérieur d'un palindrome est au plus égal à l'unité.

Détermination des changements de sens:

La présence dans Sv de deux termes consécutifs de la forme Sv(i)=v et Sv(i+1) = $\bar{v}$ est considéré comme un changement potentiel de sens d'évolution.

Soient deux sous suites S_1, S_2 de Sv consécutives délimitées par 3 changements potentiels de sens telles que l'une des suites S_1, $(\overline{S_2})^{-1}$ englobe l'autre.

Exemple: Sv = (... a $\bar{a}$ b $\bar{b}$ a $\bar{a}$ ...) S_1=($\bar{a}$ b) S_2=($\bar{b}$ a)
$(S_2)^{-1}$ = (a $\bar{b}$) $(\overline{S_2})^{-1}$ = ($\bar{a}$ b)

En dehors des palindromes l'existence de telles sous suites révèle un changement réel de sens sur une trajectoire sans bifurcation.

A partir de ces considérations Defrenne [51] propose un algorithme de construction du modèle de la trajectoire. L'ambiguité relative à l'existence de bifurcation est levée par interrogation de l'opérateur. Cette procédure permet d'inférer tout modèle de trajectoire étoilée.

2.3.Identification à partir d'une séquence ordre/compte rendu

Nous étudions ci-après une autre alternative à l'identification du modèle de la trajectoire à partir d'une séquence ordres/comptes-rendus.

2.3.1. Principe général

Soit $S = ((r_1,o_1),(r_2,o_2),\ldots,(r_j,o_j))$ une séquence échantillon ordres/comptes rendus telle que $r(t)<>r(t+1)$ et/ou $o(t)<>o(t+1)$. La séquence Sr extraite de S est telle que $r(t)<>r(t+1)$. On considère que la séquence Sr est complète au sens défini au paragraphe 2.2.

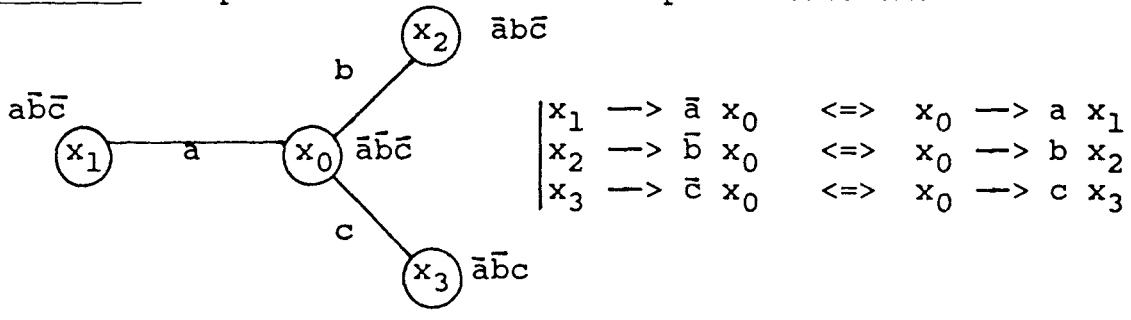
Pour déterminer la trajectoire on considère l'ensemble des transitions $(r_i \text{—>} vr_j)$ observées dans la séquence. La présence d'une transition $(r_i \text{—>} vr_j)$ entraine l'existence de la transition $(r_j \text{—>} vr_i)$ suivant l'hypothèse 2. Mais attention! les points singuliers correspondants ne sont pas nécessairement les mêmes (voir Fig.IV.9).

L'ensemble des transitions simples observées dans Sr permet d'obtenir un graphe non orienté G = (X,U) où les sommets $X=\{x_1,x_2,\ldots,x_k\}$ sont étiquetés par l'alphabet de compte rendu. Les arcs $U = \{(x_i,x_j),\ldots\}$ sont étiquetés par les noms de variables qui changent de valeur.

Exemple:

Soit la séquence Sr = ($a\bar{b}\bar{c}$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}b\bar{c}$ $\bar{a}\bar{b}\bar{c}$ $a\bar{b}\bar{c}$ $\bar{a}\bar{b}\bar{c}$ $\bar{a}\bar{b}c$ $\bar{a}\bar{b}\bar{c}$ $a\bar{b}\bar{c}$ $\bar{a}\bar{b}\bar{c}$) le graphe obtenu est le suivant:

Figure IV.4 Graphe des transitions simples entre CR.



En considérant le graphe non orienté on peut distinguer deux types de sommets:

a) les sommets reliés aux autres par au maximum deux arcs. Ces sommets sont considérés comme déterministes. (Un sommet déterministe correspond à un seul point singulier).

b) les sommets ayant plus de deux arcs sont étiquetés par des comptes rendus dits **non déterministes**. Ces sommets peuvent correspondre soit à des points singuliers différents, soit à des sommets de divergence ou convergence.

Toute la difficulté de l'identification consiste à trouver si le compte rendu non déterministe correspond à une bifurcation ou deux points singuliers différents. Pour cela nous allons essayé de mieux caractériser ces deux types de PS.

On considère les sommets déterministes (x_1,x_2,x_3) dans l'exemple) et on cherche dans la séquence échantillon Sr les doubles, triples, ... transitions entre ces sommets.
Pour l'exemple fig.IV.4 on trouve: (x_1, x_2), (x_1, x_3).

En éclatant les sommets non déterministes pris à l'intérieur de ces transitions entre sommets déterministes on obtient un graphe "éclaté" tel que celui représenté à la figure IV.5.a pour l'exemple de la fig.IV.4. Cet éclatement revient à étiqueter dans la séquence Sr chaque CR r_i déterministe par x_i; un CR r_j non déterministe est étiqueté par $x_{jkl} = x_{jlk}$ en fonction des CR r_k, r_l tels que $r_k <> r_l$ qui lui sont adjacents dans la séquence.

L'orientation relative des arcs suivant un sens positif arbitraire permet d'obtenir un ensemble de trajectoires possibles. Les fig.IV.5.b,c,d,e illustrent quelques unes des trajectoires que l'on peut déduire du graphe éclaté fig.IV.5.a en fusionnant éventuellement certains sommets ayant même CR.

Figure IV.5 Graphe éclaté et trajectoires possibles.

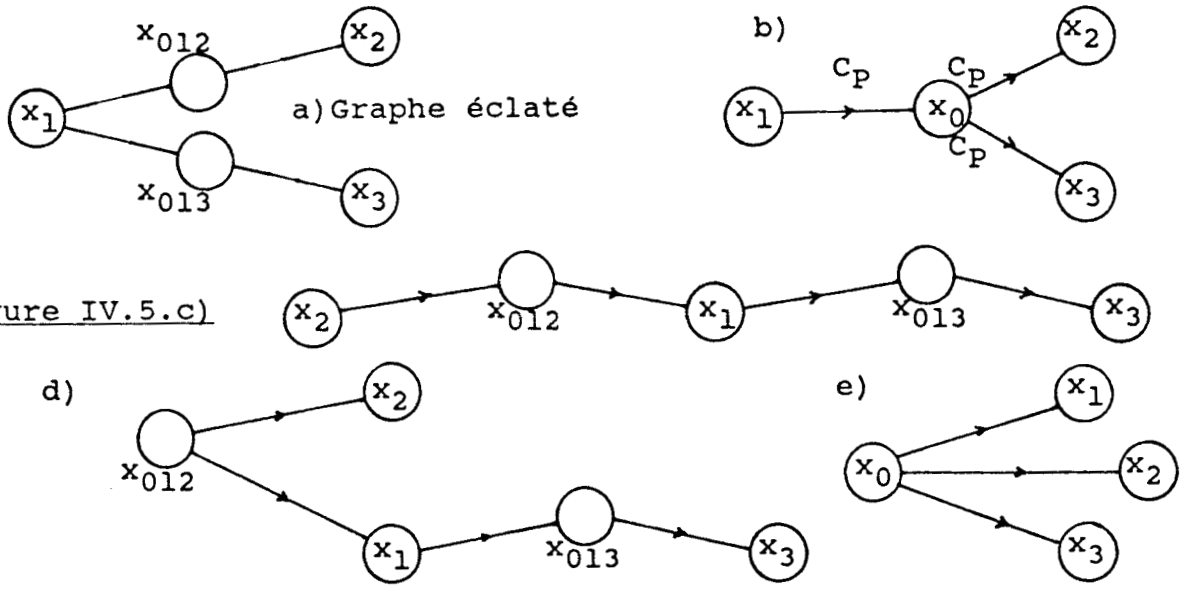


Figure IV.5.c)

La connaissance de l'orientation relative des arcs permet de retrouver la trajectoire réelle. Pour ce faire l'exploitation de la séquence d'ordres associée à la séquence de comptes rendus échantillon peut permettre de réaliser cette orientation suivant un sens positif arbitraire.

Hypothèse 3

Nous supposons que tant qu'il n'y a pas de changement d'ordre, il n'y a pas de changement de commande. Ce qui suppose que la machine séquentielle formée par les préactionneurs est stable.

Condition pour une orientation complète:

Pour que l'orientation soit complète il faut que chaque tronçon de trajectoire qui n'est pas une extrémité soit parcouru sans changement d'ordre avec le trançon suivant et précédent au moins une fois au cours de la séquence échantillon.

Dans le cas d'une trajectoire simple une condition nécessaire et suffisante pour que l'orientation soit complète est que la trajectoire soit parcourue entièrement sans changement d'ordre.

Avec le formalisme des règles de production, l'orientation permet de classer ces règles en deux groupes suivant les sens d'évolution arbitraires positif et négatif.

2.3.2. Algorithme d'identification de la trajectoire

L'identification de la trajectoire à partir d'une séquence échantillon S ordres/comptes rendus est effectuée de la manière suivante:

A) A partir de la séquence Sr on recence les éléments de l'alphabet de compte rendu.

B) La recherche des transitions entre les éléments de cet alphabet permet d'obtenir l'ensemble des règles de production $r_i \longrightarrow v\ r_j$ ou le graphe non orienté des transitions simples.

Suivant l'hypothèse 2, chaque règle de production $r_i \longrightarrow v\ r_j$ doit avoir une homologue dans l'autre sens $r_j \longrightarrow \bar{v}\ r_i$. Si cela n'est pas le cas, les règles de production sont complétées par l'introduction des règles homologues.

Ainsi la séquence Sr peut être complète même si la trajectoire n'est parcourue que dans un seul sens au cours de l'observation.

C) A partir du nombre de fois qu'un compte rendu est représenté dans ces règles de production comme facteur gauche ou droit on partage l'alphabet de compte rendu en deux classes:
- CRD contient les éléments déterministes et
- CRND les éléments non déterministes.

D) En éliminant les comptes rendus non déterministes dans la séquence Sr on obtient une séquence Sr_D. Nous supprimons aussi les éléments $r_D(t+1)$ si $r_D(t)=r_D(t+1)$.
Pour que ceci soit possible il faut que le nombre de comptes rendus déterministes soit supérieur ou égal à 2.

E) La séquence Sr_D permet de recencer les transitions entre comptes rendus déterministes: (r_{Di}, r_{Dj}).

F) Ces transitions entre les r_D sont complétées par les comptes rendus intermédiaires non déterministes pour obtenir un ensemble de règles de production de la forme:

$$r_{Di} \dashrightarrow_v r_{ND} \dashrightarrow_v \ldots \dashrightarrow_v r_{Dj}$$

ou le graphe "éclaté".

G) L'orientation du graphe ainsi obtenu suivant un sens d'évolution positif arbitraire est réalisée à l'aide de la séquence d'ordres So. On suppose que cette orientation est complète.

H) La trajectoire est obtenue par fusionnement de certains sommets ayant le même compte rendu non déterministe.
Afin d'obtenir un automate déterministe, en particulier lorsqu'il y a des bifurcations, et puisqu'on ne dispose d'aucune autre information, certains sommets du graphe ayant des CR non déterministes sont fusionnés.
Deux sommets x_{jrk}, x_{jrl} ayant un même compte rendu r_j non déterministe sont fusionnables si à partir d'un sommet x_r l'évolution vers x_{jrk} ou x_{jrl} peut être effectuée avec le même sens de déplacement (Fig.IV.6).

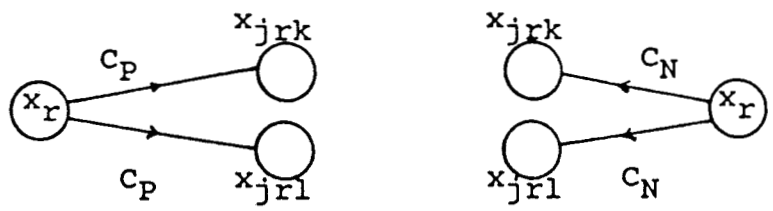
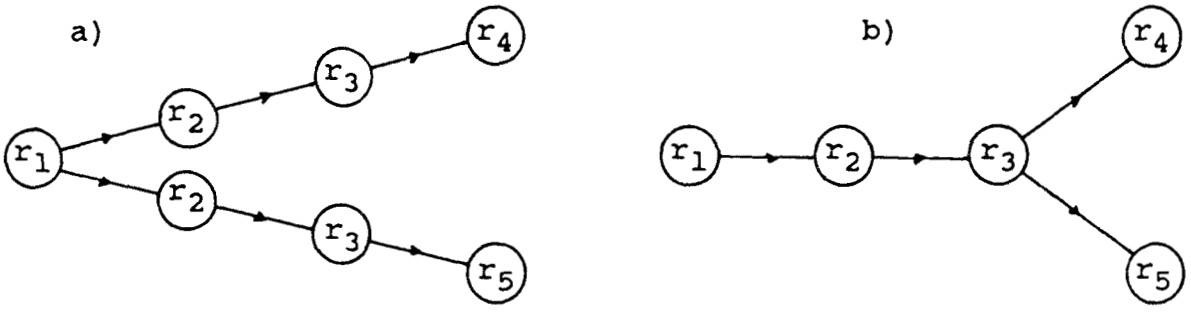


Figure IV.6

Ce fusionnement est réalisé par transitivité: à partir du graphe orienté de la figure IV.7.a, le fusionnement permet d'obtenir la trajectoire de la figure IV.7.b.

Figure IV.7: Obtention de la trajectoire par fusionnement successif.



Si l'orientation est complète la trajectoire obtenue est unique. Si au contraire, l'orientation n'est pas complète le problème d'inférence admet plusieurs solutions.

La présence, en plus, d'une séquence (r_2,r_1,r_2) parcourue avec le même ordre aurait donné la trajectoire linéaire suivante:

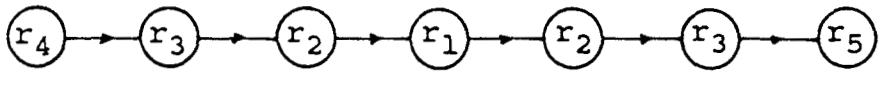


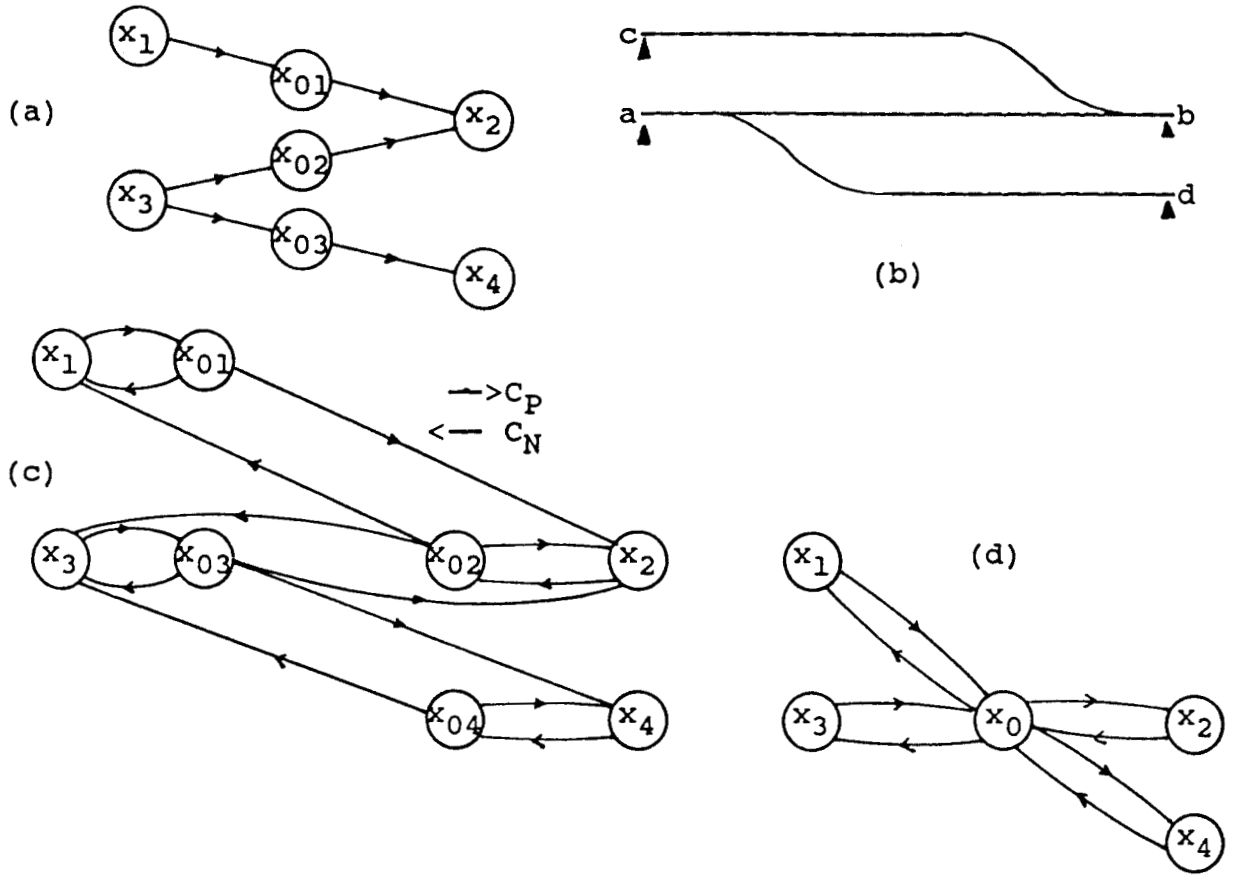
Figure IV.8

Plusieurs sommets ayant un même CR associé sont fusionnables en un seul s'il sont fusionnables deux à deux, donc la propriété prise pour le fusionnement est une relation d'équivalence.

L'exemple de trajectoire de la figure IV.9.b représente un cas particulier: les 3 sommets x_{01}, x_{02} et x_{03} du graphe éclaté de la fig.IV.9.a ne sont pas fusionnables deux à deux.

La trajectoire réelle est celle représentée à la Fig.IV.10.c. Le fusionnement des 3 sommets aurait donné la trajectoire de la figure IV.10.d qui n'est pas fausse mais modifie le langage accepté par l'automate correspondant.

Figure IV.9 Exemple de trajectoire particulière.



2.4. Exemples d'identification de trajectoire:

2.4.1. Exemple N°1:

Soit la séquence Ordres/comptes rendus suivante supposée complète au sens du paragraphe 2.2 et de l'orientation:

Ordre	o_1 o_1 o_2 o_3 o_4 o_4 o_2 o_3 o_1 o_1 o_3 o_4 o_4 o_2 o_1 o_1 o_3
CR	r_1 r_0 r_2 r_2 r_2 r_0 r_1 r_1 r_1 r_0 r_3 r_3 r_0 r_1 r_1 r_0 r_4

Alphabet de compte rendu:

$\{r_0=\bar{a}\bar{b}\bar{c}\bar{d},\ r_1=a\bar{b}\bar{c}\bar{d},\ r_2=\bar{a}b\bar{c}\bar{d},\ r_3=\bar{a}\bar{b}cd,\ r_4=\bar{a}\bar{b}\bar{c}d\}$

Sr=(r_1 r_0 r_2 r_0 r_1 r_0 r_3 r_0 r_1 r_0 r_4)
Sv=($\bar{a}$ b $\bar{b}$ a $\bar{a}$ c $\bar{c}$ a $\bar{a}$ d)

Règles de production simples et graphe associé:

Observées:

$r_1 \longrightarrow \bar{a}\ r_0$ $\quad r_0 \longrightarrow a\ r_1$

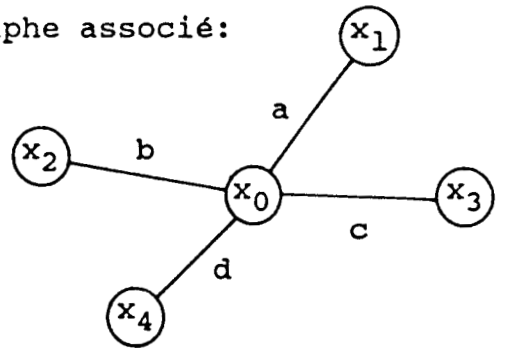
$r_2 \longrightarrow \bar{b}\ r_0$ $\quad r_0 \longrightarrow b\ r_2$

$r_0 \longrightarrow c\ r_3$ $\quad r_3 \longrightarrow \bar{c}\ r_0$

$r_0 \longrightarrow d\ r_4$

Ajoutées:

$r_4 \longrightarrow \bar{d}\ r_0$



Les comptes rendus déterministes sont: $\{r_1, r_2, r_3, r_4\}$

On obtient les règles de production suivantes entre comptes rendus déterministes:

Observées:

$r_1 \longrightarrow \bar{a} r_0 \longrightarrow b r_2$ $\quad r_2 \longrightarrow \bar{b} r_0 \longrightarrow a r_1$

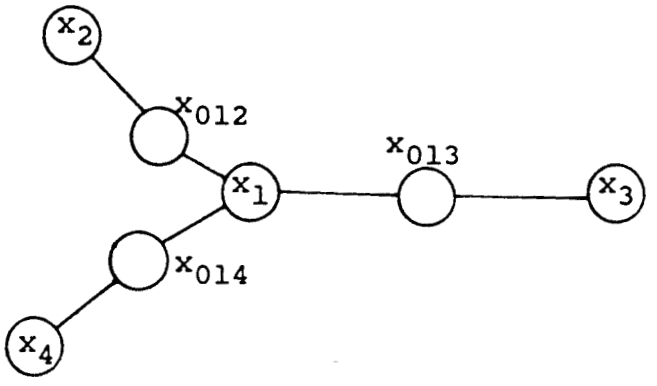
$r_1 \longrightarrow \bar{a} r_0 \longrightarrow c r_3$ $\quad r_3 \longrightarrow \bar{c} r_0 \longrightarrow a r_1$

$r_1 \longrightarrow \bar{a} r_0 \longrightarrow d r_4$

Ajoutées:

$r_4 \longrightarrow \bar{d} r_0 \longrightarrow a r_1$

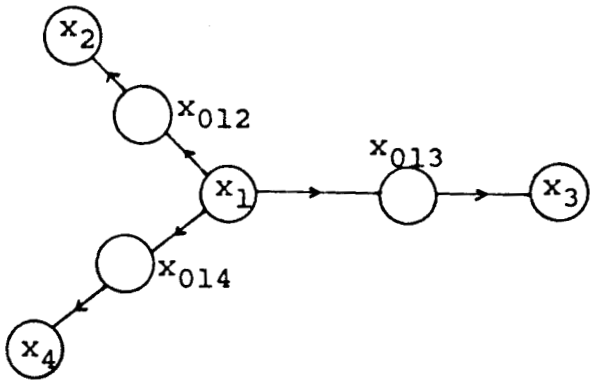
Ce qui donne le graphe éclaté suivant:



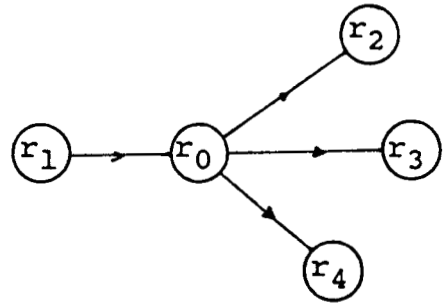
La séquence d'ordres permet de repérer les séquences de comptes rendus suivantes qui sont parcourues dans un même sens:

$(r_1\ r_0\ r_2)$, $(r_2\ \ r_0\ \ r_1)$, $(r_1\ r_0\ r_3)$, $(r_3\ \ r_0\ \ r_1)$, $(r_1\ r_0\ r_4)$

ce qui donne l'orientation suivante:



Le fusionnement permet de retrouver la trajectoire:



Remarque:

La présence d'une séquence telle que (r_0 r_1 r_0) parcourue avec le même ordre aurait aboutit à une trajectoire différente.

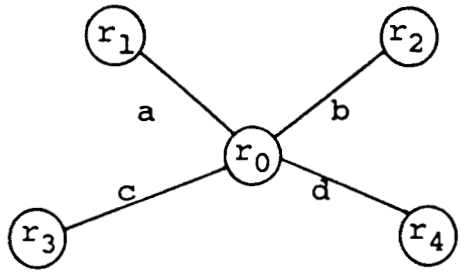
2.4.2. Exemple N°2:

La séquence échantillon est la suivante:

Ordre	o_1	o_2	o_3	o_3	o_2	o_4	o_4	o_2	o_2	o_2	o_2	o_4	o_3	o_2
CR	r_0	r_1	r_1	r_0	r_2	r_3	r_4	r_1	r_0	r_3	r_0	r_4	r_0	r_4

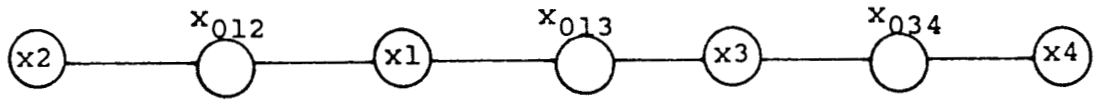
Avec l'alphabet de compte rendu:
$\{r_0=\bar{a}\bar{b}\bar{c}\bar{d},\ r_1=a\bar{b}\bar{c}\bar{d},\ r_2=\bar{a}b\bar{c}\bar{d},\ r_3=\bar{a}\bar{b}c\bar{d},\ r_4=\bar{a}\bar{b}\bar{c}d\}$

Sr=(r_0 r_1 r_0 r_2 r_0 r_1 r_0 r_3 r_0 r_4 r_0 r_4)
Sv=(a $\bar{a}$ b $\bar{b}$ a $\bar{a}$ c $\bar{c}$ d $\bar{d}$ d)
Ce qui donne le graphe et les règles de production:



$r_1 \longrightarrow \bar{a}\ r_0$ $\quad$ $r_0 \longrightarrow a\ r_1$
$r_2 \longrightarrow \bar{b}\ r_0$ $\quad$ $r_0 \longrightarrow b\ r_2$
$r_3 \longrightarrow \bar{c}\ r_0$ $\quad$ $r_0 \longrightarrow c\ r_3$
$r_4 \longrightarrow \bar{d}\ r_0$ $\quad$ $r_0 \longrightarrow d\ r_4$

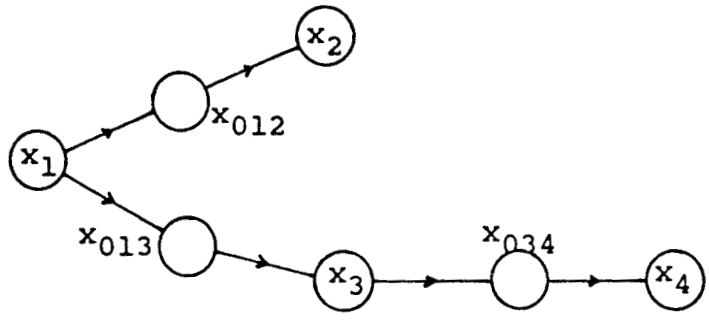
Seul le compte rendu r0 est non déterministe. Le graphe éclaté est le suivant:



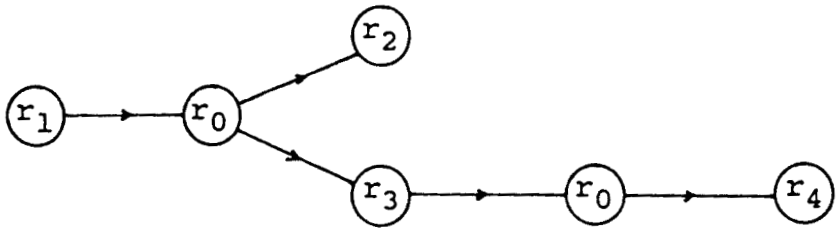
L'orientation à l'aide de la séquence d'ordres associée permet d'obtenir les séquences suivantes sans changement d'ordre:

$(x_1\ x_{012}\ x_2)$, $(x_1\ x_{013}\ x_3\ x_{034}\ x_4)$

et par suite le graphe orienté suivant:



On obtient la trajectoire suivante par fusionnement:



3. IDENTIFICATION DES COMMANDES

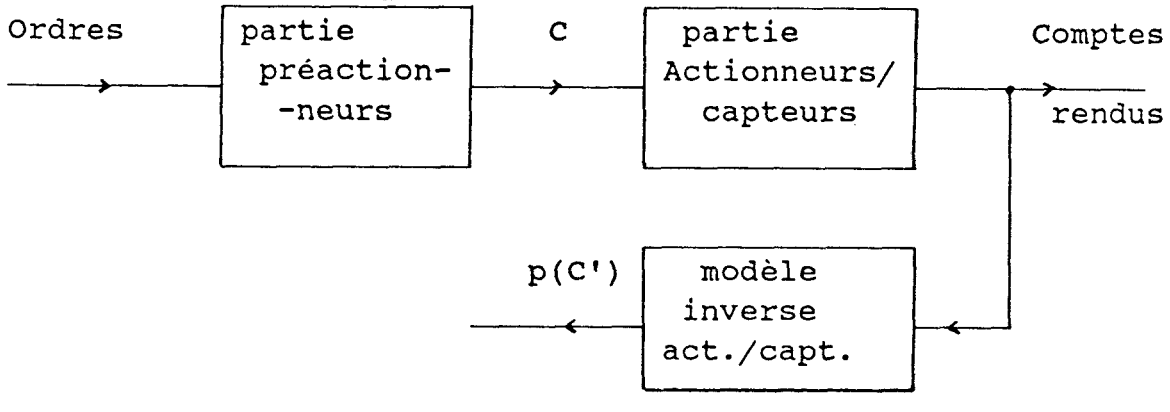
3.1. Généralités:

L'objectif visé est de déterminer la commande appliquée à chaque instant à partir d'une séquence O/R et du temps mis entre deux changements consécutifs d'ordre et/ou de compte rendu et obtenir une séquence O/p(C) pour l'identification de la partie "préactionneurs". L'obtention, à chaque instant de l'observation, d'un vecteur de probabilité sur les commandes nécessite de déterminer, au cours d'une phase d'apprentissage préalable, le modèle de comportement inverse de la partie actionneurs/capteurs (Fig.IV.10).

A partir d'une séquence de comptes rendus observée sur la partie opérative, la connaissance de la trajectoire suivie par la grandeur mesurée permettra de déterminer le sens de la commande appliquée entre chaque changement de compte rendu.

La classification des temps de parcours d'un même point singulier doit permettre de déterminer le nombre de commandes différentes en sens et en module.

Figure IV.10: Détermination du modèle inverse de la partie actionneurs/capteurs.



3.2. Apprentissage des classes de commande à partir des mesures des temps d'évolution

3.2.1. Formulation du problème

Nous distinguons quatre types de parcours possibles d'un point singulier en fonction du sens (positif, négatif) et du changement ou non de sens par rapport au parcours précédent pour tenir compte de la forme de la came et des capteurs.

Etant donnée une séquence S ordres/comptes rendus observée sur la partie opérative saine telle que o(t)<>o(t+1) et/ou r(t)<>r(t+1) ainsi que les mesures de temps entre deux changements consécutifs d'ordre et/ou de compte rendu, nous nous proposons dans une première étape de réaliser un apprentissage des classes de commandes.

Nous utilisons le modèle de la trajectoire identifiée précédemment sans fusionnement pour classer les mesures de temps de parcours de chaque point singulier en quatre groupes en fonction du sens d'évolution actuel et du sens du parcours précédent.

Les mesures dont le groupe n'est pas déterminé ne sont pas prises en compte. Ceci peut arriver dans les deux cas suivants:

- En phase d'initialisation et pour la première action élémentaire observée où le sens du parcours précédent n'est pas connu.

- Dans le cas d'un palindrome si on observe une évolution ayant pour origine et extrémité le sommet pivot sans sortir du palindrome. Le groupe n'est pas déterminé non plus pour l'action élémentaire qui suit cette évolution.

Le modèle de trajectoire utilisé peut être non déterministe. Le sens de la commande n'est pas immédiatement connu,à chaque changement de CR. La procédure utilisée mémorise les alternatives de transition possibles tant que l'indéterminisme subsiste.
En particulier pour les évolutions à partir du sommet pivot d'un palindrome, le sens ne peut être connu qu'à la sortie du palindrome.

Ce choix est motivé par le fait que le fusionnement effectué pour obtenir un automate déterministe risque de faire une confusion entre PS de dimensions différentes. Cette confusion complique la phase d'apprentissage des commandes.

3.2.2.Problèmes posés par les temps d'arrêt:

Les temps de traversée des PS ne correspondent pas directement à ceux enregistrés dans la séquence. Si on cumule les temps pendant la présence d'un même compte rendu on inclue également les temps d'arrêt.

Ces derniers posent alors un problème difficile à résoudre. Nous distinguons deux types d'arrêts: les arrêts commandés et les arrêts par blocage de l'évolution. Ce blocage peut être dû à l'action d'une autre sous PO ou à l'actionneur lui-même "en fin de course".

Hypothèse 4

Nous supposons pour simplifier qu'il n'existe pas d'interaction entre trajectoires de type blocage d'évolution.

hypothèse 5

Nous supposons qu'au cours de la séquence échantillon le parcours des points singuliers se fait sur une commande maintenue. Les changements de commande ne sont alors admis qu'au début des PS (mais les ordres peuvent changer).

En fonction de la séquence d'ordre observée pendant la présence d'un même CR, nous pouvons avoir les situations suivantes:

a) Changements de compte rendu observés sans changement d'ordre ou avec un seul changement. La sous séquence est de la forme:

...	o_1	o_2	o_3	...	$r_j <> r_k$, $r_k <> r_l$
...	r_j	r_k	r_l	...	
...	t_{j1}	t_{k2}	t_{l3}	...	

Le temps t_{k2} de persistance de r_k en présence de o_2 correspond bien au temps de traversée du PS auquel est associé ce compte rendu. Il n'y a donc pas d'arrêt commandé.

b) Il existe plusieurs changements d'ordres au cours de la présence d'un même CR. La séquence est de la forme:

...	o_1	o_2	...	o_n	o_m	...	$o_2 <> o_3$, $o_{n-1} <> o_n$
...	r_k	r_j	...	r_j	r_l	...	$r_k <> r_j$, $r_l <> r_j$
...	t_{k1}	t_{j2}	...	t_{jn}	t_{jm}	...	

- Pendant le temps t_{jn} de présence de r_j et de o_n il y a présence d'une commande non nulle puisqu'on observe un changement de CR. Mais on ne connait pas parmi les ordres $o_1, \ldots o_n$ celui ayant provoqué le déplacement.
- L'incertitude quant à la commande présente pendant les temps t_{jn-1}, ..., t_{j2} est partagée entre la commande Stop ou nulle et la commande ayant provoqué l'évolution au cours de t_{jn}. Il peut y avoir en effet des changements d'ordres (au cours de l'évolution) sans changement de commande.

3.2.3.Influence sur l'apprentissage des classes de commande:

L'existence de plusieurs changements d'ordres pendant la présence d'un même CR peut avoir un effet négligeable dans l'apprentissage des classes de commande si la somme des temps cumulés au cours de ces changements est très faible devant le temps où il y a effectivement évolution:

$$(t_{j2} + \ldots + t_{jn-1}) \ll t_{jn}$$

Si les temps cumulés ci-dessus ne sont pas négligeables leur conséquence sur l'apprentissage des commandes peut être néfaste. Nous étudions leurs effets au §3.2.6.

Remarque:

Si les changements d'ordres n'interviennent qu'au début des points singuliers, les temps d'arrêt sont facilement mis en évidence.

Nous supposons dans une première approche que les temps d'arrêt sont négligeables devant les temps de traversée des points singuliers. Nous prenons, initialement les temps où il y a effectivement évolution (t_{jn}) pour réaliser l'apprentissage.

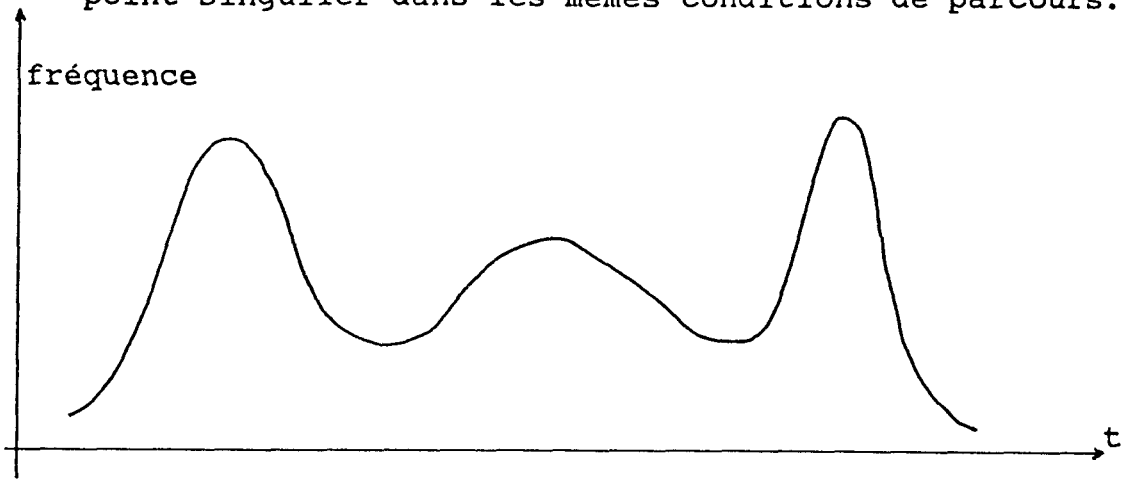
3.2.4.Décomposition d'une distribution en ses composantes gaussiennes

Les temps de traversée d'un point singulier dans les mêmes conditions de parcours sont distribués suivant une certaine fonction. La figure IV.11 représente un exemple d'une telle distribution.

Pour que l'apprentissage soit possible il faut que chaque point singulier soit parcouru avec toutes les vitesses possibles, avec et sans changement de sens par rapport au parcours précédent et un nombre de fois suffisant.

Si nous supposons que les temps de traversée d'un point singulier sous l'action d'une commande maintenue sont distribués suivant une loi normale, le problème d'apprentissage des classes de commandes revient à identifier les composantes gaussiennes de l'histogramme.

Figure IV.11 : Distribution des temps de traversée d'un point singulier dans les mêmes conditions de parcours.



Le problème de décomposition d'une distribution en ses composantes gaussiennes, d'une grande utilité pratique, a fait l'objet d'une abondante littérature [61] -- [66].

Celui-ci peut être formulé de la façon suivante [62], [63]. Soit X une variable aléatoire continue de densité f(X). On suppose que celle-ci représente une population formée de k sous populations. Chaque sous population i suit une distribution gaussienne de moyenne m_i, d'écart type σ_i et intervient dans la population globale avec une proportion $p(C_i)$ ou probabilité a priori de la classe C_i. La fonction f(X) peut se mettre sous la forme:

$$\left.\begin{array}{l} f(X) = \sum \{p(C_i).p(X|C_i) \quad \text{pour } i=1,\ k\} \\ \text{avec} \quad p(X|C_i) = (1/\sqrt{2\pi}\,\sigma_i).\exp(-(X-m_i)^2/2\sigma_i^2)) \\ \sum p(C_i) = 1 \text{ pour } i=1,\ k \end{array}\right| \qquad (1)$$

$p(X|C_i)$ est la fonction de densité conditionnelle normale.

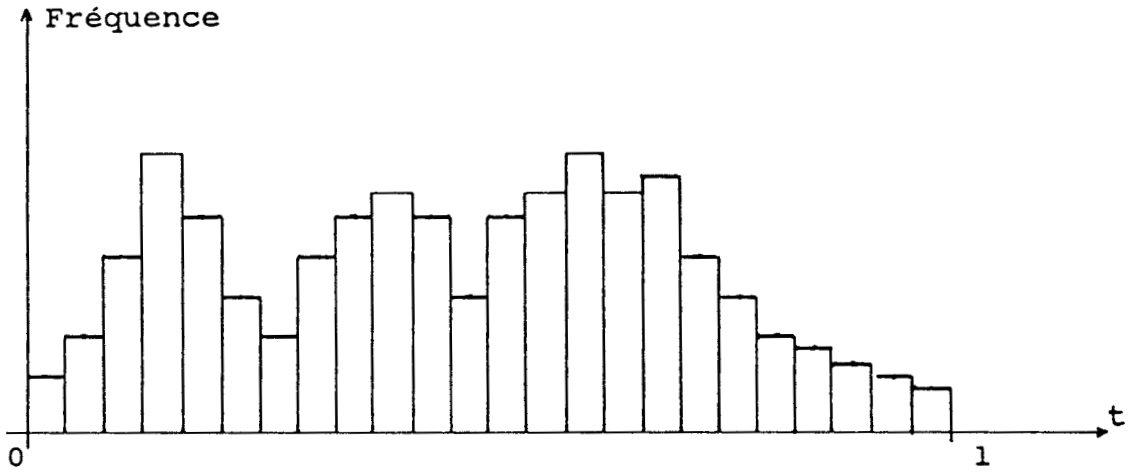
L'identification des composantes gaussiennes du mélange consiste à rechercher le système de nombres:

$\{ k,\ (p(C_i), m_i, \sigma_i)$ pour $i=1,\ k\}$ vérifiant (1).

Dans la pratique, la fonction f(X) n'est connue que par un échantillon comportant un certain nombre N de réalisations indépendantes.

La distribution est alors représentée par un histogramme comportant un certain nombre n de classes fonction du pas d'échantillonnage. Les mesures sont, en général, préalablement normalisées sur l'intervalle [0,1]. La figure IV.12. montre un exemple d'histogramme.

Figure IV.12:Histogramme de distribution des temps de parcours.



3.2.5. Choix d'une méthode de résolution

Certaines méthodes d'analyse supposent la connaissance a priori du nombre k de classes ou l'égalité des écarts types des composantes. Plusieurs principes ont été utilisés pour identifier un mélange gaussien parmi lesquels nous citerons:

a)- Les méthodes graphiques [64],
b)- La déconvolution successive [65],
c)- Maximum de vraissemblance [62];
d)- Les méthodes stochastiques [66],
e)- L'analyse des domaines de convexité [61], [63].

Certaines de ces méthodes sont relativement complexes, comme la méthode du maximum de vraissemblance, ou exigent un nombre très important d'observations comme les méthodes d'estimation stochastique. D'autres méthodes plus simples et plus opérationnelles comme les méthodes graphiques donnent de bons résultats quand les composantes sont bien séparées.

Dans le problème de détermination des commandes, qui nous préoccupe, le nombre de classes n'est pas connu a priori. De plus il est souhaitable de réduire au maximum la période d'observation et donc le nombre de mesures nécessaires à l'apprentissage.

Devant le grand nombre de mesures à mémoriser et d'histogrammes à analyser pour une partie opérative donnée, il serait préférable d'utiliser une méthode d'apprentissage séquentielle qui évite de disposer en mémoire de l'ensemble des mesures pour faire l'analyse.

Nous supposons dans la suite que l'analyse des distributions a été faite suivant l'une des méthodes citées ci-dessus.

3.2.6. Analyse des résultats de l'apprentissage

Les hypothèses faites précédemment, en particulier en ce qui concerne la distribution normale des temps de parcours, peut se révéler inexacte dans la réalité à cause des arrêts mais aussi d'autres facteurs. Suivant qu'il y a ou non arrêt, par exemple, au début d'un point singulier, le nombre de classes détectées peut être différent suivant l'importance de l'inertie de démarrage.

Théoriquement le nombre de classes détectées dans les différents groupes et pour tous les points singuliers doit être le même si les hypothèses admises ont été respectées. Si tel n'est pas le cas, les causes des anomalies doivent être recherchées en effectuant éventuellement une nouvelle fois l'apprentissage pour les cas où il y a problème. Les causes possibles peuvent être:

a) L'absence d'une commande pour certains points singuliers. (le PS n'est jamais parcouru avec une vitesse donnée pour des raisons d'exploitation par exemple).

b) La confusion de deux classes voisines. Si les modules des vitesses sont réellement très proches, cette confusion peut être constatée pour la plupart des PS ou même pour tous. Dans ce dernier cas il n'existe malheureusement pas de solution.

c) La création de classes supplémentaires lorsque des changements d'ordres se produisent à des dates aléatoires au cours de l'évolution ou à cause de l'inertie.

Nous étudions ci-après quelques solutions possibles pour résoudre ces problèmes d'apprentissage.

Améliorations de l'apprentissage

Lorsque plusieurs changements d'ordre sont observés pendant la présence d'un même CR l'analyse est systématiquement reprise afin d'identifier l'ordre le plus probable ayant provoqué le déplacement et par suite les temps d'arrêt éventuels et le temps réel de parcours du PS. La séquence est ici de la forme:

...	o_1	o_2	...	o_n	o_m	...	$o_2 <> o_3$, $o_{n-1} <> o_n$
...	r_k	r_j	...	r_j	r_1	...	$r_k <> r_j$, $r1 <> r_j$
...	t_{k1}	t_{j2}	...	t_{jn}	t_{jm}	...	

Nous distinguons les situations suivantes:

a) La somme des temps cumulés pendant lesquels il y a possibilité d'arrêt ($s_j=t_{j2}+ \ldots +t_{jn-1}$) est négligeable devant le temps de déplacement t_{jn} et ceci pendant toute la durée d'observation et pour l'histogramme considéré.
L'importance relative des temps s_j par rapport aux temps t_{jn} est mieux représentée par le rapport:

$$\beta = \frac{\sum s_j}{\sum t_{jn}}$$

L'effet produit sur l'apprentissage reste faible si les cas où s_j n'est pas négligeable sont relativement rares.

Si donc $\beta << 1$ l'effet sur l'apprentissage est négligeable mais une incertitude totale subsiste quant à la commande présente pendant les temps $t_{j2}, \ldots, t_{jn-1}$.

b) Si les temps s_j sont importants l'apprentissage est refait en prenant comme mesures de temps les durées de présence du CR: $t_j=tj_2+ \ldots +t_{jn}$. La comparaison des nouvelles classes identifiées aux précédentes peut permettre d'appréhender les arrêts dans certains cas mais le grand nombre de situations possibles ne permet pas d'avoir une solution générale.

Ce problème reste à approfondir. Nous ferons simplement les remarques suivantes:

- Si les arrêts sont globalement constants et s'il n'y a pas de changement d'ordre au cours de l'évolution l'effet sur les classes identifiées se manifestera par une translation ou la création de nouvelles classes suivant qu'ils sont ou non systématiques.

- Si les temps d'arrêts ont une grande dispersion leurs effets peuvent se manifester par une translation des classes si leur présence est systématique.
- Si les temps d'arrêt sont négligeables et les changements d'ordre observés se présentent au cours de l'évolution, les nouvelles classes identifiées doivent se rapprocher des commandes réelles. En particulier, le nombre de classes identifié est conforme à la réalité.

L'utilisation des distances relatives des points singuliers et la comparaison des classes identifiées dans chaque groupe pour un PS donné peuvent aider à résoudre certaines situations.

3.3. Identification des commandes par classement

Nous avons fait l'apprentissage des classes de commande. Pour chaque histogramme nous avons déduit les valeurs estimées des moyennes $\hat{m}_i$, des écarts types $\hat{\sigma}_i$ et des probabilités a priori $\hat{p}(C_i)$ de chaque classe.

Il s'agit maintenant de faire le classement des temps de parcours observés sur une séquence O/R/t pour déterminer les commandes admissibles ou un vecteur de probabilité sur ces commandes. Notre but est d'obtenir une séquence O/p(C) afin d'identifier la machine séquentielle constituée par la partie "préactionneurs". Il suffit donc de calculer simplement la probabilité a postériori $p(C_i|X)$ de chaque classe :

$$P(C_i|X) = \frac{p(X|C_i).p(C_i)}{f(X)} \qquad (2)$$

Ces probabilités sont telles que $\sum p(C_i|X) = 1$ pour i=1, k dans l'histogramme considéré. Il faut de plus que la somme des composantes du vecteur de probabilité obtenu sur l'ensemble des commandes soit égale à 1.

3.3.1. Réalisation de la correspondance entre classes identifiées dans les différents histogrammes:

Pour réaliser la correspondance entre les différents modules de vitesses identifiés entre points singuliers (correspondance entre classes des histogrammes pour différents PS), nous utilisons la notion de distance relative des PS. Ce problème revient à étiqueter les classes des histogrammes par les commandes C_{P1}, C_{P2}, ..., C_{Pr}, C_{N1}, C_{N2}, ..., C_{Nk}.

Le nombre de commandes non nulles à considérer dans chaque sens d'évolution est déterminé pendant la phase d'apprentissage précédente.

Soient m_{1ij}, m_{2ij}, ..., m_{kij} les moyennes des classes identifiées dans un histogramme d'un groupe i et pour un PS_j donnés correspondant à des vitesses non nulles. Ces valeurs sont ordonnées: $m_{1ij} < m_{2ij} < \ldots < m_{kij}$.

Si le nombre de classes identifiées pour deux points singuliers PS_1 et PS_2 est le même nous devons avoir l'égalité approximative des rapports:

$$\frac{m_{1i1}}{m_{1i2}} \approx \frac{m_{2i1}}{m_{2i2}} \approx \ldots \approx \frac{m_{ki1}}{m_{ki2}}$$

Si cette égalité est vérifiée à un facteur près les classes homologues sont étiquetées par la même commande. Si la différence des rapports est importante l'apprentissage doit être reconsidéré.

3.3.2. Procédure de classement et d'obtention de la séquence ordres/commandes

Le traitement de la séquence O/R/t, pour obtenir une séquence O/p(C) est conduite de la façon suivante. Soit une sous séquence de la forme:

...	o_1	o_2	...	o_n	o_m	...	$o_2 <> o_3$, $o_{n-1} <> o_n$
...	r_k	r_j	...	r_j	r_l	...	$r_k <> r_j$, $r1 <> r_j$
...	t_{k1}	t_{j2}	...	t_{jn}	t_{jm}	...	

Etape 1: détermination du sens des commandes entre chaque changement d'ordre et/ou de compte rendu:

L'utilisation de l'automate correspondant au modèle de la trajectoire permet de déterminer le sens des commandes non nulles (entre chaque changement de compte rendu).

Lorsque le sens n'est pas déterminé ou le sens du parcours précédent, l'incertitude sur la commande présente peut être totale ou partielle suivant qu'il y a ou non possibilité d'arrêt.

Etape 2: Classement et calcul du vecteur de probabilité sur l'ensemble des commandes.

2.a) Aux endroits de la séquence où le sens de l'évolution est connu ainsi que celui de l'évolution précédente, le temps t_{jn} est classé dans le groupe correspondant. On obtient les valeurs des probabilités a postériori de chaque classe. Le vecteur de probabilité sur l'ensemble des commandes est de la forme:

$$[p(C_0|X)\ p(C_{N1}|X)\ ---\ p(C_{Nr}|X)\ p(C_{P1}|X)\ ---\ p(C_{Pk}|X)]^T$$

$$\sum p(C_i|X) = 1 \text{ pour } i=0,N1 \text{ à } Nr,\ P1 \text{ à } Pk$$

où seules les probabilités des commandes correspondant au sens de déplacement ne sont pas nulles.

2.b) Lorsqu'il y a un arrêt éventuel (pendant les temps $t_{j2},\ldots t_{jn-1}$) et si celui-ci n'est pas identifié l'incertitude est partagée entre C_0 et les commandes C_N ou (exclusif) C_P suivant le sens d'évolution en cours. La probabilié d'avoir C_0 est telle que $p(C_0|X)=Max\{p(C_{Ni+Pi}|X)\}$.

3.4. Remarques et conclusion

Nous avons supposé implicitement qu'il existe une commande d'arrêt, donc que la machine séquentielle constituée par les préactionneurs, possède au moins un état donnant cette sortie. Ce n'est pas toujours le cas. L'arrêt peut être simplement provoqué par un blocage de l'évolution comme, par exemple, pour les vérins.

Nous avons pu identifié le modèle de comportement inverse de la partie actionneurs/capteurs mais celui-ci reste à affiner. Le problème d'identification des temps d'arrêt n'a pas été entièrement résolu.

Certains effets des erreurs d'apprentissage comme la confusion de deux classes ou commandes dans le même sens peuvent avoir des conséquences négligeables sur l'efficacité du test. Il n'en est pas de même sur l'identification de la partie "préactionneurs".

Nous abordons dans le chapitre 5 qui suit le problème d'identification des machines séquentielles en particulier dans le cas où il y a une incertitude sur les sorties comme pour les préactionneurs.

CHAPITRE V: IDENTIFICATION DE MACHINES SEQUENTIELLES

CHAPITRE V

IDENTIFICATION DE MACHINES SEQUENTIELLES

1.INTRODUCTION

L'identification de processus industriels modélisés par des automates est un domaine de recherche qui a été très exploré ces dernières années. L'obtention du modèle d'un processus permet de concevoir sa commande et/ou son diagnostic.

Le problème de détermination d'un modèle à partir d'un ensemble d'observations certaines possède en général une solution. Plusieurs algorithmes à vocations différentes ont été proposés jusqu'à ce jour.

Lorsque les résultats d'une reconnaissance des formes sont exploités en vue d'une quelconque action de commande ou d'identification, des difficultés peuvent surgir du fait de l'imprécision en général contenue dans ces résultats; spécialement lorsque des méthodes statistiques de décision sont mises en oeuvre. A l'éxception de quelques cas particuliers, il est en effet impossible de garantir un taux de bonne classification à postériori de 100%.

La solution, la plus généralement adoptée dans ce cas, consiste en une modélisation de l'ensemble processus/système d'information par des modèles approximatifs ou probabilistes, en l'occurence les automates stochastiques, prenant en compte le caractère incertain de l'information. Ces modèles basés sur un "taux de transition" sont bien adaptés pour des processus fondamentalement non déterministes.

Un seuillage sur les valeurs des probabilités de transition permet dans certains cas d'obtenir un modèle "déterministe", mais le caractère statistique de cette décision minimisant la probabilité d'erreur ne tient pratiquement plus compte des évènements rares.

Lorsque le processus physique est entièrement déterministe et que seul le système d'information est probabiliste à causes de difficultés d'accès à l'information réelle, d'autres solutions doivent être recherchées. Les modèles stochastiques sont trop approximatifs pour être utilisés dans des problèmes comme le diagnostic de pannes.

Ce chapitre discute de ces problèmes nouveaux et des solutions apportées. Nous traiterons successivement des points suivants:

- Position du problème d'identification et principe des méthodes existantes.
- Identification d'une machine séquentielle par la contradiction dans le cas d'observations certaines.
- Identification à partir d'informations incertaines.
- Application à l'identification de la partie "préactionneurs" d'une partie opérative.

L'annexe 1 regroupe les définitions et théorèmes relatifs aux machines séquentielles.

2. IDENTIFICATION A PARTIR D'OBSERVATIONS CERTAINES:

2.1.Position du problème

Le problème d'identification d'une machine séquentielle M_o dont la structure interne est inconnue (boite noire) et où seules les entrées et sorties sont accessibles à l'observation, peut être formulé, dans une première approche, de la manière suivante:

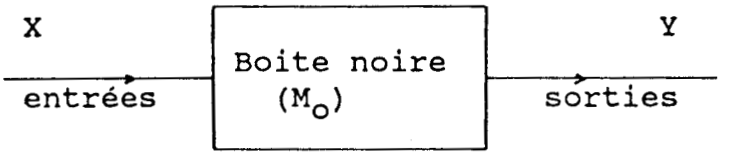


Figure V.1: Problème d'identification.

Etant donné un ensemble d'observations $S=\{(\underline{x},\underline{y}) \mid (\underline{x},\underline{y}) \in (XxY)^*\}$ avec X et Y les alphabets d'entrée et de sortie respectivement, trouver une machine séquentielle M_i réduite ayant le même comportement entrée/sortie que M_o.

On montre [7] que ce problème d'identification ne peut avoir de solution que si les alphabets d'entrée/sortie X et Y sont connus d'avance .

Plusieurs types d'expérimentations peuvent être envisagés suivant que l'on peut ou non agir sur les entrées et suivant que l'on peut ou non revenir à un état déterminé de M_o dit état initial.

2.2.Principe des méthodes d'identification existantes:

a)Méthodes utilisées en inférence grammaticale

L'utilisation de la relation d'invariant droit ou k-finale introduite par NERODE [26] fournit plusieurs algorithmes utilisés principalement dans les problèmes d'inférence grammaticale [19], [22], [23].

D'une manière générale, ces méthodes partent d'un ensemble échantillon

$$E = \{ \underline{z} \mid \underline{z} \in (X^* \times Y)^* \}$$

ou $E = \{ \underline{z} \mid \underline{z} \in (X \times Y)^* \}$

ou $E = \{ \underline{z} \mid \underline{z} \in V_T^* \}$ avec V_T l'alphabet terminal pour un langage. $\{\underline{z}\}$ est un sous ensemble du comportement d'un certain état initial correspondant par exemple à l'axiome d'une grammaire.

Des procédures de manipulation de l'échantillon basées sur la relation de k-finale et k-équivalence permettent d'obtenir un modèle d'automate réduit [3],[4]. Ces méthodes sont mal adaptées dans le cas qui nous préoccupe. En effet pour un automatisme industriel, il est pratiquement impossible de revenir à un état initial préspécifié pour obtenir un échantillon qui est le comportement de celui-ci.

b)Méthode par simplification

Une autre méthode d'identification toujours basée sur des procédures de simplification proposée par KELLA [2], part d'une séquence d'entrée/ sortie échantillon considérée comme une machine séquentielle avec un nombre d'états égal à la longueur de la séquence. L'utilisation des procédures classiques de simplification (états compatibles, etc...) adaptées à l'occasion à ce type particulier de machine permet d'obtenir un modèle à nombre d'états réduit.

Bien qu'il existe maintenant des algorithmes de simplification performants [25], [5], [7], la méthode reste très lourde à manipuler dès que la séquence échantillon devient un peut trop longue à cause de son caractère combinatoire.

c)Identification par la contradiction

La méthode décrite par NARANJO [1] utilise le principe de la contradiction processus-modèle et une procédure d'ajustement séquentielle du modèle jusqu'à obtenir un automate réduit acceptant la séquence échantillon.

Malheureusement, la non mémorisation du passé nécessite des séquences d'entrée/sortie souvent fort longues. De plus il est pratiquement impossible d'étendre l'algorithme à l'identification de processus à partir d'observations incertaines. Le principe de cette méthode qui sera adoptée içi est décrit au paragraphe 3 ci-après.

3.PRINCIPE DE L'IDENTIFICATION PAR LA CONTRADICTION

3.1. Différentiation des états, séquences contradictoires:

Soit une machine séquentielle de type Moore définie par: $A = \{ X,Q,Y,\delta,\lambda \}$ avec X et Y les alphabets d'entrée et de sortie respectivement. Q est l'ensemble des états internes de la machine; δ : QxX—>Q la fonction état suivant ou de transition et λ :Q—>Y la fonction de sortie.

Le comportement d'un état $q \in Q$ est défini par l'ensemble B(q) des séquences d'entrée/sortie $(\underline{x},\underline{y})$ telles que $\lambda(q,\underline{x})=\underline{y}$

$$B(q) = \{ (\underline{x},\underline{y}) \mid \lambda(q,\underline{x}) = \underline{y}\}$$

Deux états q_1 et q_2 sont équivalents si leurs comportements $B(q_1)$ et $B(q_2)$ sont identiques. En d'autres termes si on applique une séquence d'entrée $\underline{x}$ à partir de l'état q_1 et q_2 on obtient la même séquence de sortie.

Inversement, si deux états q_1, q_2 ne sont pas équivalents alors il existe une séquence d'entrée $\underline{x}$ de longueur $|\underline{x}|\leq n-1$, ($n = |Q|$) telle que $\lambda(q_1,\underline{x}) <> \lambda(q_2,\underline{x})$. On considère les deux séquences telles que:

$$\begin{array}{l|l} \lambda(q_1,\underline{x}) = y^1y^2\ldots y^ky^i & y^i <> y^j \\ \lambda(q_2,\underline{x}) = y^1y^2\ldots y^ky^j & \lambda(q_1) = \lambda(q_2) = y^1 \end{array}$$

Ces deux séquences de sortie sont dites <u>contradictoires</u>. L'existence de séquences contradictoires observables dans le comportement externe d'un automate est donc un indice de présence d'états non équivalents pouvant donc être différentiés.

<u>Exemple</u>:

Soit l'automate simple de la figure V.2, les deux états q_1 et q_3 tels que $\lambda(q_1) = \lambda(q_2) = y_1$ peuvent être différentiés par la séquence d'entrée $\underline{x} = b$ de longueur 1.

$$\begin{array}{l} \lambda(q_1,b) = y_1y_1 \\ \lambda(q_3,b) = y_1y_2 \end{array}$$

Les deux séquences d'entrée/sortie contradictoires observables dans le comportement externe de l'automate et pouvant différentier q_1 et q_3 peuvent s'écrirent:

$$\begin{cases} S_{1,1} = y_1,b,y_1 \\ S_{1,2} = y_1,b,y_2 \end{cases}$$

Si on extrait toutes les séquences d'entrée/sortie contradictoires observables dans le comportement externe d'un automate, celles-ci peuvent être réparties en K sous ensembles, K = |Y|, chaque sous ensemble { S_i } correspondant à une sortie y_i origine de la séquence.

Figure V.2: Automate A1.

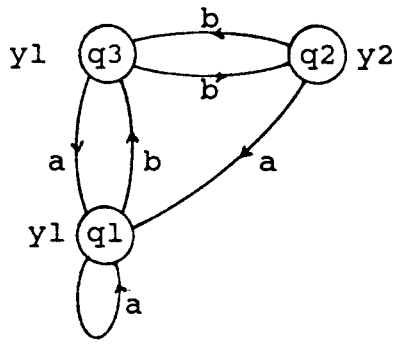
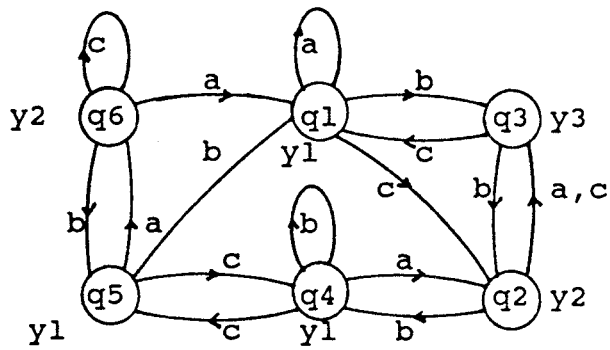


Figure V.3: Automate A2.



Lemme V.1: Un automate réduit A = (X,Q,Y,δ,λ) de type Moore possède un seul état q_i tel que $\lambda(q_i) = y_i$ si et seulement si l'ensemble { S_i } de séquences d'entrée/sortie contradictoires associées à y_i est vide.

Autrement dit, si aucune paire de séquences contradictoires associée à une sortie y_i n'est observable dans le comportement externe de l'automate, alors on est assuré que celui-ci ne possède qu'un seul état donnant la sortie y_i. Ce lemme découle de la définition de l'équivalence entre états.

En corrolaire de ce lemme, dès que { S_i } n'est plus vide, on est certain qu'il existe plusieurs états q non équivalents tels que $\lambda(q)=y_i$.

3.2.Propriétés des états associés aux séquences contradictoires

L'existence d'une paire de séquences contradictoires $S_{i,k}$, $S_{i,l}$, d'après le lemme, implique qu'il existe au moins deux états q1,q2 non équivalents donnant la même sortie, cependant à chaque séquence S_i peut correspondre plusieurs états.

Dans l'exemple de la figure V.3, on peut observer les deux séquences contradictoires suivantes et leurs états associés:

$$\begin{cases} S_{1,1} = y1,b,y1,a,y1 \\ S_{1,2} = y1,b,y1,a,y2 \end{cases} \quad ; \quad \begin{cases} SQ_{1,1} = q5,b,q1,a,q1 \\ SQ_{1,2} = q4,b,q4,a,q2 \end{cases}$$

A la séquence $S_{1,1}$ correspond l'ensemble des états $Q_{1,1}=\{q1\}$ et à la séquence $S_{1,2}$ correspond l'ensemble $Q_{1,2} = \{ q4,q2 \}$.

Un sous ensemble S_i de séquences contradictoires contient des éléments de longueurs différentes et éventuellement infinies. Dans le comportement externe de l'automate A2 on peut observer les séquences contradictoires suivantes correspondantes à la sortie y1:

Séquences de longueur 1:

y1,a,y1	y1,b,y3	y1,c,y2
y1,a,y2	y1,b,y1	y1,c,y1

Séquences de longueur 2:

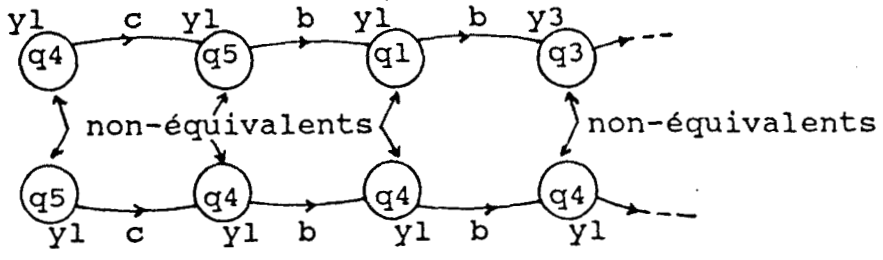
y1,b,y1,a,y1	y1,b,y1,b,y3	y1,a,y1,b,y1
y1,b,y1,a,y2	y1,b,y1,b,y1	y1,a,y1,b,y3
y1,b,y1,c,y1	y1,a,y2,a,y3	y1,a,y2,c,y2
y1,b,y1,c,y2	y1,a,y2,a,y1	y1,a,y2,c,y3

Séquences de longueur supérieure à 2:

y1,c,y1,b,y1,b,y3	y1,c,y1,c,y1,...,c,y1,b,y1,b,y1
y1,c,y1,b,y1,b,y1	y1,c,y1,c,y1,...,c,y1,b,y1,b,y3

Cette dernière séquence d'une longueur éventuellement infinie constitue un cas particulier. Ceci se produit lorsqu'il existe deux états q1, q2 tels que $\lambda(q1)=\lambda(q2)$ et $\delta(q1,x)=q2$, $\delta(q2,x)=q1$ pour une certaine entrée x.

Lemme V.2: Pour une paire de séquences contradictoires donnée, les états homologues entre les deux séquences ne sont pas équivalents, en particulier les deux états origines.



En effet, si deux états homologues sont équivalents alors les séquences de sortie obtenues à partir de ces états seraient identiques, il n'y aurait donc pas de contradiction.

3.3.Longueur des séquences contradictoires:

Si deux états q1, q2 sont k-équivalents mais ne sont pas (k+1)-équivalents alors on est assuré de trouver dans l'ensemble (S_i) au moins deux séquences contradictoires $S_{i,m}$, $S_{i,n}$ de longueur k+1. Cependant, l'existance de séquences contradictoires de longueur k+1 ne signifie pas qu'il existe necessairement des états k-équivalents.

Lemme V.3: L'existence de séquences contradictoires de longueur k implique l'existence de séquences contradictoires de longueur $l \leq k$.

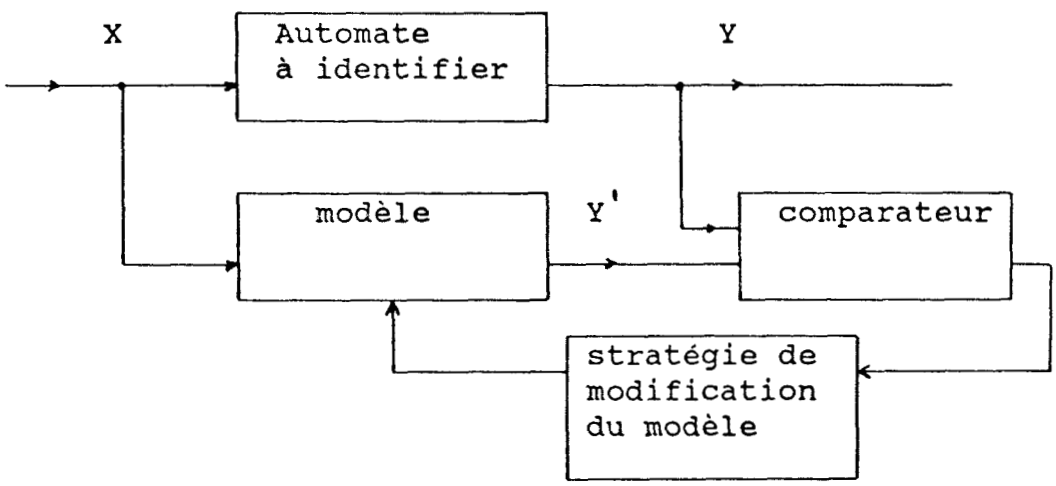
Ce lemme [1] indique que les séquences contradictoires observables dans le comportement entrée/sortie d'un automate ont des longueurs éventuellement croissantes et que cet accroissement, à chaque fois, se fait par pas unitaire. Toutes ces remarques seront très utiles pour l'identification de l'automate.

4. APPRENTISSAGE D'UN MODELE D'AUTOMATE PAR LA CONTRADICTION

4.1. Principe général:

Le diagramme de la figure V.4 ci-dessous montre la procédure générale d'identification d'un modèle d'automate par la contradiction.

Figure V.4 :identification d'un modèle d'automate par la contradiction.



Le modèle peut être initialisé avec un nombre d'états égal au cardinal de l'alphabet de sortie Y, chaque sortie ayant donc un seul état associé. Le résultat de l'identification sera alors l'obtention d'un modèle de type Moore.

Au cours de l'évolution du processus, le modèle est complété par l'établissement des transitions en fonction des entrées de façon à ce que le modèle accepte la séquence d'entrée/sortie.

Lorsque, étant dans un état déterminé, à une entrée x qui se présente correspond déjà une transition établie antérieurement, le modèle évolue suivant cette transition vers un état q donnant une sortie y'. Le processus, à ce moment donne une sortie y.

Si y=y' le modèle est pour l'instant correct. Si au contraire y<>y'alors il y a une contradiction entre le processus et le modèle. Ce dernier doit donc être modifié. Tout le problème consiste à trouver ce qui doit être changé dans le modèle pour éliminer les différentes contradictions qui se présentent.

4.2.Stratégies de réajustement du modèle

Plusieurs stratégies de réajustement du modèle peuvent être envisagées NARANJO dans [1], propose deux algorithmes basés sur une représentation du modèle par un ensemble de règles de production et de règles de transition. Le premier algorithme se donne comme contrainte la non mémorisation du passé donc de la séquence d'identification. Ceci a pour effet de rendre la procédure de modification du modèle quelque peu problématique. La séquence nécessaire à l'obtention d'un modèle complet est souvent fort longue, l'information qui y est contenue n'étant pas entièrement exploitée. Cette procédure est pratiquement impossible à étendre à l'identification d'un processus à partir d'informations incertaines.

Le deuxième algorithme proposé manipule une séquence échantillon enregistrée pour en extraire toutes les séquences contradictoires. L'ensemble des règles de production est alors déterminé. Il reste à établir les règles de transition, ce qui est fait par des essais successifs. La première étape de cet algorithme est manifestement trop longue.

Pour toutes les raisons évoquées ci-dessus et compte tenu de notre objectif, nous avons développé une autre procédure que nous exposons ci-après.

5. ALGORITHME D'IDENTIFICATION:

5.1.Description générale

La représentation adoptée pour le modèle est la table de transition classique à laquelle nous avons adjoint la séquence des états du modèle associée à la séquence d'entrée/sortie du processus, ce qui constitue un historique utile pour résoudre les contradictions qui se présentent.

La procédure, schématisée par le diagramme général de la figure V.5 est donc la suivante.

Au départ le modèle est initialisé avec un nombre d'états égal à celui des sorties: $|Q|=|Y|$; chaque sortie étant associée à un état différent. Tant que la séquence d'entrée/sortie n'est pas finie on réalise les opérations suivantes :

a) Tant qu'il n'y a pas de contradition entre modèle et échantillon on parcourt la séquence et on établit éventuellement les transitions entre états. Lorsqu'une transition peut être établie vers plusieurs états possibles on en choisit un et on garde en mémoire les autres alternatives.

b) Lorsqu'une contradiction se présente on procède éventuellement aux essais des alternatives existantes pour la résoudre. En effet celle-ci peut avoir deux causes différentes:

=> Une ou plusieurs transitions établies antérieurement sont fausses.

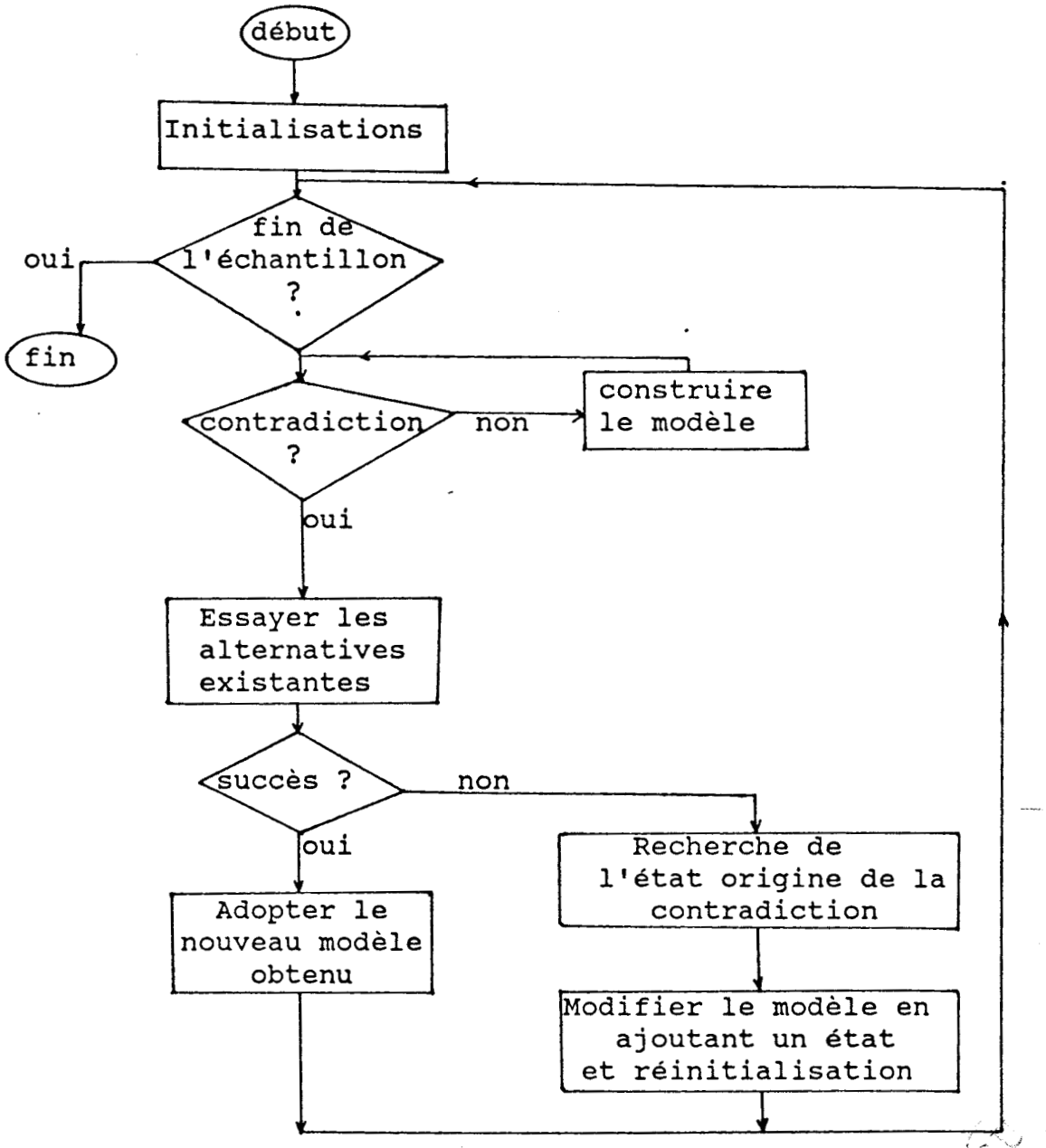
=> Le nombre d'états du modèle est incomplet. (deux états non équivalents ont été confondus.)

c) Deux cas de modification sont à envisager suivant le résultat de ces essais:

1) On obtient un modèle d'essai permettant de lever la contradiction actuelle. Celui-ci est alors adopté comme nouveau modèle et on continue la construction en a).

2) Toutes les possibilités de transitions ont été épuisées sans avoir résolu la contradiction. Une procédure de détermination de l'état origine de la contradiction est alors lancée. Le modèle se voit ainsi ajouter un état supplémentaire ayant comme sortie associée celle de l'état origine de la contradiction. Dans ce cas le modèle est réinitialisé en supprimant toutes les relations de transition. On reprend donc la construction du modèle à partir de l'instant initial mais avec un état en plus.

Figure V.5 : Procédure générale de construction du modèle:



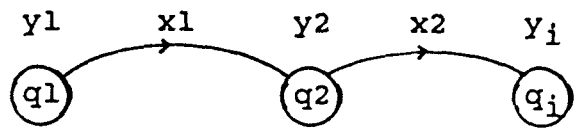
5.2. Propriétés du modèle obtenu à chaque étape:

Pour que l'algorithme soit effectif il faut qu'à chaque étape de la construction le modèle obtenu permette:

1) De détecter toute différence entre la séquence générée par le modèle et la séquence échantillon par le biais des séquences contradictoires.

2) De résoudre cette contradiction c-à-d déterminer l'état origine de la contradiction et les modifications à faire subir au modèle pour qu'il accepte la séquence d'entrée/sortie échantillon.

Dans le modèle initial, tous les états q_i tels que $\lambda(q_i)=y_i$ ont été supposés équivalents et donc confondus en un seul état. Supposons que dans le processus il existe deux états q1, q1' non équivalents donnant la même sortie y1. Il existe alors une séquence d'entrée $\underline{x}$ telle que appliquée à partir de q1 et q1' donne deux séquences de sortie contradictoires $\underline{y}$ et $\underline{y}'$.



Si la séquence échantillon comporte les sous séquences $(\underline{x},\underline{y})$ et $(\underline{x},\underline{y}')$ dans cet ordre, le modèle sera initialement construit de telle façon qu'à partir de q1, par application de $\underline{x}$ produise la séquence de sortie $\underline{y}$. Lorsque la deuxième sous séquence $(\underline{x},\underline{y}')$ se présente, le modèle déjà construit va alors générer normalement la séquence $(\underline{x},\underline{y})$ en contradiction avec $(\underline{x},\underline{y}')$. On peut montrer, à partir de l'équivalence entre automates que toute différence de comportement entrée/sortie entre modèle et processus est ainsi détectée.

5.3.Procédure de résolution des contradictions:

La procédure de lever des contradictions qui est le coeur de l'algorithme est la plus délicate. Deux problèmes se posent à chaque fois qu'une contradiction se présente:

1) Quel est l'état origine de la contradiction? ou en d'autres termes quel est (ou quels sont) le(s) état(s) à dupliquer (procédure d'éclatement des états) pour résoudre la contradiction?

2) De quelle manière faut-il modifier le modèle pour qu'il accepte la séquence ou se rapproche d'une certaine manière du processus?

On ne traite ici que les contradictions qui n'ont pas pu être résolues par un choix judicieux des transitions entre états. Le modèle obtenu alors est le "meilleur" au sens où il accepte la plus grande séquence relativement à ses états actuel.

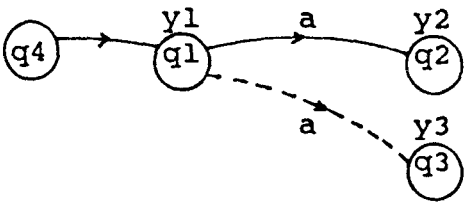
Si donc une contradiction subsiste c'est que le nombre d'états du modèle est insuffisant, toutes les possibilités de transition entre états ayant été essayées.

Les contradictions qui seront détectées au cours du parcours de l'échantillon se présentent d'une manière aléatoire, et leur résolution se faisant à chaque étape, ne sont pas indépendantes.

EXEMPLE

A une étape de l'algorithme on rencontre la contradiction suivante:

| y1,a,y2
| y1,a,y3

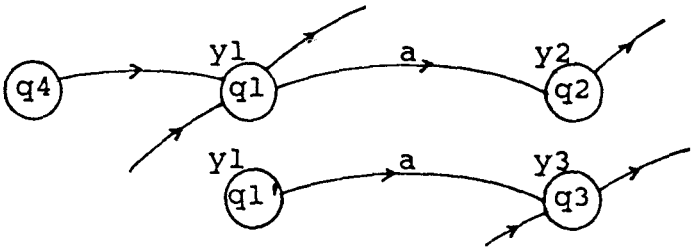


Si dans le modèle, q1 est le seul état donnant y1 alors on est sûr que q1 est l'origine de la contradiction et que la création d'un état q1' homologue de q1 permettrait de résoudre la contradiction à cette étape.

S'il existe déjà dans le modèle deux états q1, q1' donnant la même sortie y1 la contradiction peut provenir soit de q1 lui même ou d'un état antérieur à q1 dans la séquence (q4, etc...).

Pour pouvoir donc résoudre les contradictions à chaque étape de l'algorithme il faut mémoriser celles déjà résolues aux étapes antérieures.

La modification du modèle consiste donc à créer un état supplémentaire (q1' dans l'exemple) et lui associer la même sortie que l'état origine de la contradiction (λ(q1)).



L'état q1 possède des relations de transition établies antérieurement dont certaines (au moins une vers q1) doivent être attribuées à q1'. Mais, à ce stade, il est pratiquement impossible de déterminer lesquelles. De plus le modèle, tel que construit,accepte la séquence échantillon jusqu'à l'instant t-1 précédant la contradiction, mais du fait que celui-ci est encore incomplet il accepte aussi d'autres séquences provenant de relations de transition fausses. Avec le formalisme d'inclusion d'automates on peut exprimer ceci de la façon suivante:

Le modèle $M_{k,i}$ obtenu à une étape k est un élément de l'ensemble $M_k = \{M_{k,1}, M_{k,2}, \ldots M_{k,n}\}$ des automates acceptant la séquence échantillon jusqu'à l'instant k soit S_k.
$S_k \subseteq B(M_{k,i})$ mais l'inverse n'est pas vrai, ceci explique que le modèle puisse générer des séquences de sortie non incluses dans S_k lorsqu'une nouvelle séquence d'entrée lui est appliquée.

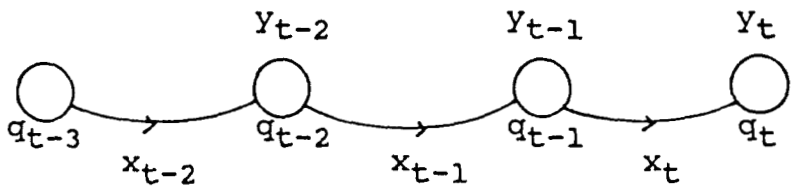
5.4. Procédure récursive de recherche de l'état de contradiction:

Compte tenu des remarques précédentes, la recherche de l'état origine de la contradiction se fait d'une manière récursive de la façon suivante:

Soient $S_e = (y_{t-3}, x_{t-2}, y_{t-2}, x_{t-1}, y_{t-1}, x_t, y_t^e)$ la séquence

donnée par l'échantillon en contradiction à l'instant t avec
$S_m = (y_{t-3}, x_{t-2}, y_{t-2}, x_{t-1}, y_{t-2}, x_t, y_t^m)$ la séquence

générée par le modèle. Celui-ci a donc été construit de la manière suivante:



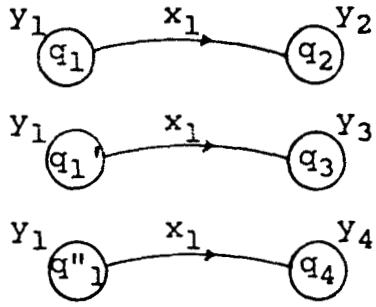
Etape 0: On initialise sur l'état à l'instant t-1 soit $q_{k-1}=q_{t-1}$;

Etape 1: L'état origine de la contradiction est q_{k-1} dans les trois cas suivants:

a) q_{k-1} n'a pas d'homologue;
b) la contradiction ($\lambda(q_{k-1})$ -> y_t^e et $\lambda(q_{k-1})$ ->y_t^m) n'a pas encore été résolue.
c) la séquence $y_{k-2}, x_{k-1}, y_{k-1}, \ldots, x_t, y_t^m$ donnée par le modèle n'a pas été observée précédemment dans l'échantillon.

Etape 2: si aucune des trois conditions n'est vraie remonter vers l'état précédent (k=k-1) et aller à l'étape 1, sinon l'état origine de la contradiction est q_{k-1}.

La condition (a) est évidente, du moins lorsque $q_{k-1} = q_{t-1}$.
La condition (b) représente la situation suivante :

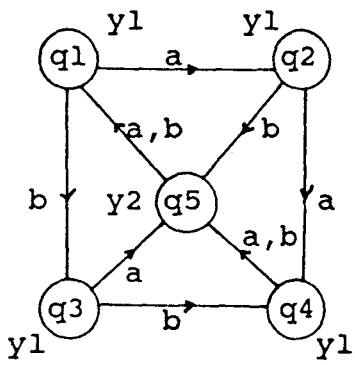


On rencontre une première fois la contradiction ($y_1 \rightarrow y_2$; $y_1 \rightarrow y_3$) on crée alors l'état q_1'. Une deuxième fois on rencontre par exemple la contradiction ($y_1 \rightarrow y_2$; $y_1 \rightarrow y_4$) ou ($y_1 \rightarrow y_3$; $y_1 \rightarrow y_4$) ces dernières n'ayant pas été résolues on crée donc maintenant un deuxième état q''_1 homologue de q_1 et q_1'.

La condition (c) élimine l'erreur pouvant être introduite par le fait que le modèle génère une séquence non observée dans l'échantillon.

Exemple:

Soit le processus suivant et la séquence échantillon correspondante:

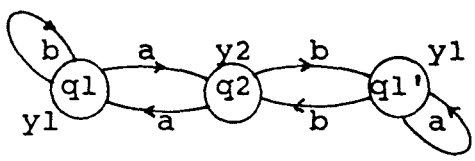


Séquence:

1	2	3	4	5	6	7	8	9	10
b	a	a	b	a	b	a	a	b	b
y1	y2	y1	y1	y2	y1	y1	y1	y2	y1

11	12	13	14	15	16	17	18	19	20
a	b	a	b	b	a	b	b		
y1	y2	y1	y1	y1	y2	y1	y1		

A l'instant 18 il y a contradiction entre modèle et échantillon.le modèle obtenu alors est le suivant:

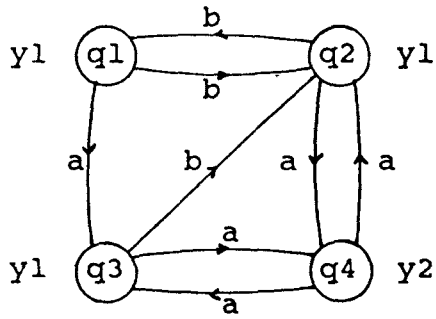


contradictions résolues:
{ y1 —>y2 ; y1 —>y1 }

Le modèle donne alors la séquence S_m = ... y_2,b,y1,b,y_2
en contradiction avec S_e = ... y_2,b,y_1,b,y_1
Nous pouvons vérifier que la séquence S_m n'a pas été observée dans l'échantillon.

5.5.Exemple d'identification:

Soit le modèle réel ci-dessous tiré de [1] et la séquence échantillon.



Echantillon:

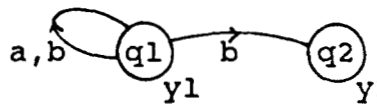
it	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
e	a	b	b	b	a	a	b	b	a	a	a	a	b	b	b
s	y2	y1	y1	y1	y1	y2	y1	y1	y2	y1	y2	y1	y1	y1	y1

it	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
e	a	a	b	b	a	a	a	a	a	b	a	b	b	a	a
s	y1	y2	y1	y1	y2	y1	y2	y1	y2	y1	y2	y1	y1	y2	y1

it	31	32	33	34	35	36	37	38	39	40	41	42	43	44
e	b	a	a	a	b	b	a	a	b	a	b	a	b	a
s	y1	y1	y2	y1	y1	y1	y2	y1	y1	y1	y1	y2	y1	y2

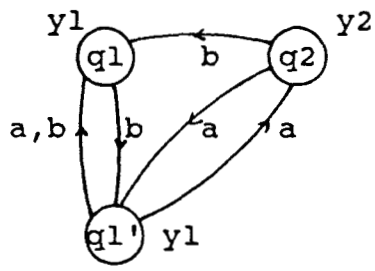
On initialise donc le modèle avec deux états q1 et q2 donnant respectivement les sorties y1 et y2.

La première contradiction se présente à l'instant 6; le modèle est alors le suivant:



modèle	q1	a	q1
	y1	a	y1
échantillon	y1	a	y2

La contradiction est résolue par la création d'un état q1' homologue de q1. Une deuxième contradiction se présente à l'instant 26:



modèle	q2	b	q1	a	q1'
	y2	b	y1	a	y1
échantillon	y2	b	y1	a	y2

Celle-ci est résolue par la création d'un état q1" homologue de q1,q1'. En effet la séquence présentée par le modèle n'existe pas dans l'échantillon. On obtient finalement un modèle réduit et équivalent au processus réel.

5.6.Propriétés des modèles obtenus par l'algorithme:

L'algorithme d'identification tel que décrit précédemment va générer à chaque instant k un modèle M_k réduit acceptant la séquence échantillon jusqu'à l'instant k soit S_k. S_k est alors un sous ensemble du comportement d'un état de M_k :

$S_k \subseteq M_k$. La condition de l'échantillon complet [23] telle que définie pour les problèmes d'inférence grammaticale est necessaire mais non suffisante pour l'obtention du modèle d'une machine séquentielle. En effet, le modèle M_i d'une machine séquentielle M_o fortement connexe doit être tel que $M_i \equiv M_o$ ou M_i quasi-équivalente à M_o.

Pour satisfaire cette exigence nous ajouterons la condition nécessaire suivante pour la complétude de l'échantillon:

- Il faut que l'échantillon contienne toutes les paires de séquences contradictoires observables dans le comportement externe de la machine séquentielle M_o à identifier.

VEELENTURF dans [3] donne comme condition pour la complétude que S contienne toutes les séquences d'entrée/sortie de longueur 2n-1 ($n=|Q|$) à partir d'un certain état initial d'une machine de Mealy.
Cette condition est peut être suffisante mais non nécessaire et oblige à disposer d'une séquence échantillon souvent fort longue.

Moyennant la condition ajoutée pour la complétude de l'échantillon l'algorithme permet d'obtenir un modèle M_i réduit et équivalent au processus si celui-ci est entièrement défini ou quasi-équivalent s'il est imcomplètement spécifié.

5.7.Quelques améliorations de l'algorithme:

La méthode d'identification utilise une procédure d'essai exaustive de toutes les transitions possibles entre états. le nombre d'itérations ainsi nécessaires devient vite très grand. Pour améliorer la vitesse de traitement, deux solutions sont possibles:

a) Filtrage des alternatives de transition à essayer:

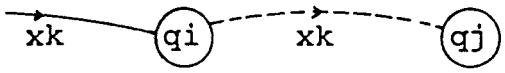
Certaines alternatives de transitions n'ont pas à être essayées. En effet celles-ci mènent vers des modèles isomorphiques à celui déjà obtenu à une permutation des états homologues près. Le filtrage de ces alternatives consiste simplement à tester si les états homologues vers l'un desquels il faut établir une transition sont isolés. Si c'est le cas le choix de l'un ou l'autre de ces états abouti à des modèles isomorphiques.
Cette condition n'est pas suffisante pour éliminer tous les modèles isomorphiques mais permet un gain de temps non négligeable.

b) Construction orientée:

Lorsque le modèle obtenu à une étape de la construction est remis en cause celui-ci n'est pas entièrement faux. Plutôt que de réinitialiser entièrement le modèle sans tenir compte des transitions établies antérieurement, on réalise une construction orientée en choisissant de préférence les transitions déjà construites aux étapes précédentes.

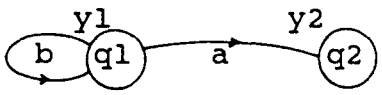
5.8.Cas d'un échantillon stable:

Lorsque le processus à identifier n'évolue que d'une situation stable à une autre, ce qui est souvent le cas pour les automatismes industriels, plutôt que de faire un enregistrement de séquence à chaque top d'horloge, on n'enregistre que lorsqu'il y a un changement des entrées ou sorties. Ceci permet d'obtenir un échantillon de longueur raisonnable. Le résultat de l'identification sera donc un modèle stable.Dans ce cas, un autre type de "contradiction" apparaît lorsqu'on ne peut réaliser une transition vers un des états sans violer la propriété de stabilité.
La situation suivante est en effet interdite si l'échantillon est stable:



$qi \neq qj$

Exemple:
Soit le modèle suivant obtenu à une étape de l'algorithme à partir d'un échantillon stable.



A un moment de la construction une transition (q2,a,q1) doit être établie mais il existe déjà une transition (q1,a,q2) ce qui risque de rendre le modèle instable. On est alors devant une contradiction dite de stabilité.
Celle-ci est résolue à peu près de la même manière que les autres en dupliquant l'état origine de la contradiction (q1 dans l'exemple) sauf que celle-ci n'est pas mémorisée. Ceci risque alors de fausser le résultat de la procédure de recherche de l'état de contradiction aux étapes ultérieures. Une solution consiste à rechercher, avant la résolution de la contradiction suivante, la paire de séquences contradictoires à mémoriser.

5.9.Test d'arrêt de l'algorithme:

Lorsqu'on arrive à la fin de l'échantillon, il peut subsister des alternatives de transition entre états non encore essayées. Si ces choix mènent tous vers une contradiction alors le modèle obtenu est le seul parmi tous ceux ayant le même nombre d'états avec les sorties associées qui accepte l'échantillon, mais peut ne pas être complet. Si au contraire l'utilisation de ces choix restants permet d'obtenir un autre modèle non isomorphique au précédent alors on peut affirmer que l'échantillon pris pour l'identification est incomplet si le processus est entièrement spécifié.

6. IDENTIFICATION D'UNE MACHINE SEQUENTIELLE A PARTIR D'OBSERVATIONS INCERTAINES

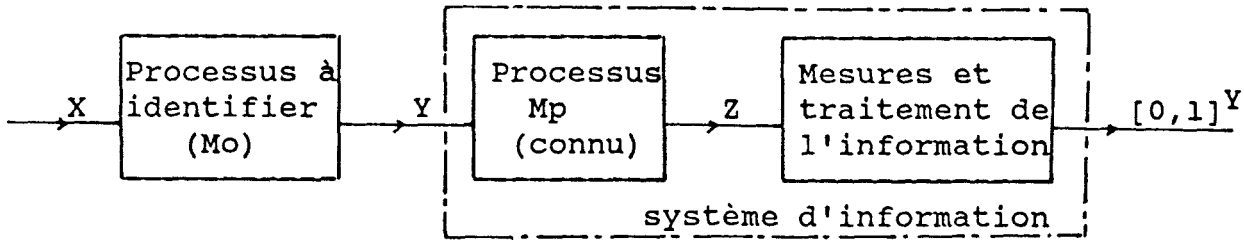
6.1.Position du problème d'identification:

Soit un automatisme industriel (noté Mo sur la figure V.6) dont le comportement entrée/sortie est entièrement déterministe.

Ses sorties Y ne sont pas accessibles à l'observation mais constituent les entrées d'un autre système Mp qui lui est observable en sortie. De plus sa structure interne est supposée connue mais n'est pas inversible pour pouvoir déterminer avec certitude ses entrées Y à partir de l'observation de ses sorties Z.

Le système d'information, à chaque instant k ne peut donner alors qu'un vecteur de probabilité Pk=[p(y1),p(y2), ... ,p(yn)] sur les sorties Y ou l'ensemble des sorties possibles.

Figure V.6: Identification d'une machine séquentielle non directement observable en sortie



Nous nous proposons d'identifier la machine séquentielle Mo à partir d'observations probabilistes constituées d'une séquence S = ($\underline{x}$,P($\underline{y}$)) ou incertaine S = ($\underline{x}$,($\underline{y}$)) avec $\underline{x} \in X^*$, $\underline{y} \in Y^*$ où X et Y sont respectivement l'alphabet d'entrée et de sortie.

6.2.Méthodes d'identification existantes:

Il n'existe pas, à notre connaissance, de procédure d'identification permettant d'obtenir le modèle déterministe d'une machine séquentielle à partir d'observations probabilistes ou incertaines.

NARANJO dans [1] propose deux cas de filtrage:

a) filtrage d'une indétermination pour éviter une contradiction.
b) filtrage d'une indétermination par utilisation d'une relation de transition établie antérieurement.

Malheureusement l'algorithme d'identification proposé dans [1] dans le cas d'observations certaines ne peut être généralisé pour prendre en compte des incertitudes sur l'échantillon.

Les autres méthodes existantes modélisent l'ensemble processus-système d'information par des automates stochastiques [6], [8], [9], [13], [11], [12] ou des automates flous [10]. Le caractère statistique de ces modèles permet d'approcher le comportement de processus fondamentalement non déterministes. Mais ils ne peuvent être utilisés ici pour le problème qui nous préoccupe. On ne peut, en effet, exclure les cas rares; le système de diagnostic risque de générer de fausses alertes ou alors il ne pourra donner qu'une probabilité qu'une erreur soit présente.

6.3. Qualité d'un échantillon:

On ne peut parler ici de la complétude de l'échantillon de la même manière que dans le cas d'observations certaines. On peut calculer des indices permettant d'estimer la qualité d'un échantillon.

a) La pauvreté d'un échantillon de longueur T sur un alphabet Y de cardinal n mesurée par la somme des incertitudes à chaque instant k:

$$I = \sum_{k=1}^{T} \sum_{i=1}^{n} pki.Log_2(pki)$$

donne une idée sur l'efficacité du système d'information. Le meilleur échantillon (certain) aura une incertitude nulle, alors que le plus mauvais aura une incertitude maximale:

$$Imax = Log_2(n)$$

b) On peut calculer aussi un facteur de séparabilité (ou de confusion) entres les différentes sorties par la somme des carrés de la différence à chaque instant k entre la probabilité pki d'une sortie yi et celle pkj de la sortie yj:

$$Cij = \sum_{k=1}^{T} (pki-pkj)^2$$

L'échantillon sera d'autant meilleur que l'indice de séparabilité Cij est grand entre toutes les sorties yi et yj.

Nous supposerons ici que le système d'information est "sérieux" : la probabilité allouée à la sortie réellement présente est toujours supérieure ou égale à celle des autres. Cette condition est en générale satisfaite lorsque le système d'information comporte une procédure de reconnaissance des formes.

6.4. Distances modèle/échantillon:

GAINES, dans [9], [11], [12] propose de calculer une distance entre un échantillon S = (x,y) incertain (sans les valeurs de probabilité) et le modèle stochastique pour évaluer leur adéquation. Ces distances sont les suivantes:

a)
$$d1 = \sum_{k=1}^{T} \sum_{i=1}^{n} (peki - pmki)^2$$

Soit la somme des carrés des différences entre les probabilités pmki données par le modèle et celles peki données par l'échantillon. peki est égal à zéro ou un.

b)
$$d2 = -\sum_{k=1}^{T} \sum_{i=1}^{n} peki.Log_2(pmki)$$

Soit la somme de moins le logarithme à base 2 des probabilités données par le modèle correspondantes aux composantes non nulles de l'échantillon.

c)
$$d3 = \sum_{k=1}^{T} (1 - pk(yi))$$

pk(yi) est la probabilité qu'alloue le modèle à la sortie yi présente dans l'échantillon.

Exemple: Soit la séquence de sortie suivante:

	instant	1	2	3	4	5	6	7	8	9
	sortie	y1	y2	y3	y1	y1	y2	y2	y3	y3
	échantillon									
Modèle	p(y1)	0.1	0.1	0.2	1.0	0.1	0.4	0.4	1.0	0.5
	p(y2)	0.2	0.4	0.1	0.0	0.2	0.6	0.6	0.0	0.5
	p(y3)	0.7	0.5	0.7	0.0	0.7	0.0	0.0	0.0	0.0

$d1 = (0.9^2+0.2^2+0.7^2)+0.62+0.14+0.0+1.34+0.32+2.0+1.5$
$= 7.26$

$d2 = -Log2(0.1)-Log2(0.4)+0.51+0.0+3.32+0.74+0.74+ \infty + \infty$
$= \infty$

$d3 = 0.9 +0.6 +0.3 +0.0 +0.9 +0.4 +0.4 +1.0 + 1.0$
$= 5.5$

La distance d2 ne tolère pas la situation où la probabilité associée à une sortie est nulle alors que cette sortie survient dans l'échantillon.

Ces distances sont telles qu'une valeur faible indique une meilleure adéquation entre modèle et échantillon. Elles peuvent être utilisées dans le cas où l'échantillon est probabiliste et le modèle déterministe.

7. ALGORITHME D'IDENTIFICATION

7.1. Description générale de l'algorithme

L'algorithme proposé pour l'identification d'une machine séquentielle à partir d'informations probabilistes ou floues généralise celui décrit précédemment pour un échantillon certain.

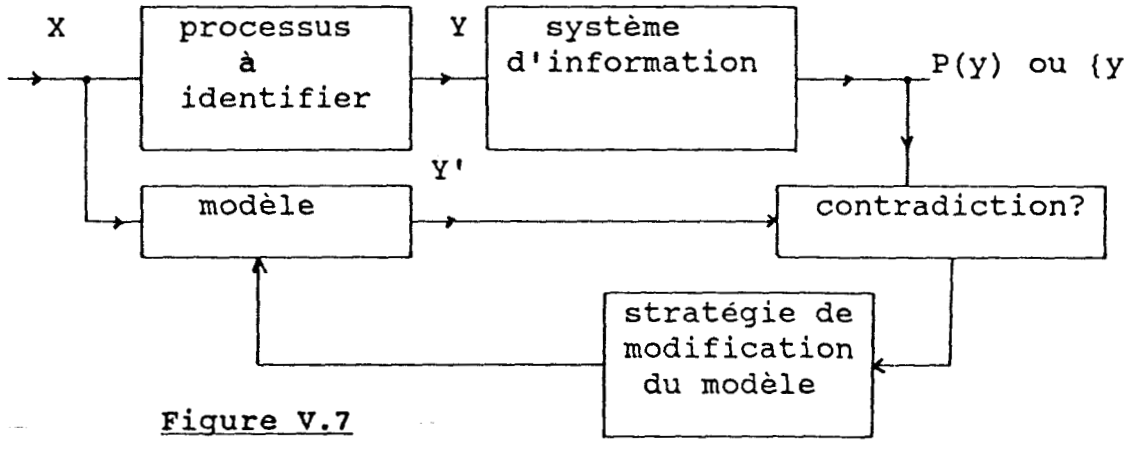


Figure V.7

Le sens de la contradiction entre modèle et échantillon est généralisé de la manière suivante:

«Il y a contradiction à un instant k entre le modèle déterministe donnant la sortie ym et l'échantillon donnant une probabilité correspondante p(ym) si celle-ci n'est pas supérieure ou égale aux probabilités des autres sorties».

Pour éviter les erreurs pouvant être dûes à la précision avec laquelle les probabilités sont données, on introduit un facteur de précision fp pouvant être fixé par l'utilisateur. Dans ce cas, on ne considère qu'il y a contradition que si p(ym) est supérieure à pmax-fp.

Exemple:

	modèle	y1	y1	y3
	p(y1)	0.4	0.3	0.6
échantillon	p(y2)	0.4	0.4	0.2
	p(y3)	0.2	0.3	0.2

contradiction

pas de contradiction

pas de contradiction si fp>=0.1

Le facteur fp peut être aussi considéré comme un facteur de confiance, mais ne sera pas utilisé dans ce sens. Nous avons supposé que le système d'information est sérieux. Notre objectif est alors d'obtenir un modèle déterministe ayant une adéquation maximum avec l'échantillon (distances d1, d2, d3 minimum) ainsi qu'un nombre d'états minimal.

L'algorithme d'identification à partir d'un échantillon probabiliste est réalisé de la façon suivante:

Etape:0 le modèle est initialisé avec un nombre d'états égal au nombre de sorties; chaque sortie étant associée à un état différent. L'état initial est choisi au premier instant ki de l'échantillon tel qu'une seule des sorties ait une probabilité supérieure aux autres de fp:

$$pk(yi) > pk(yj) + fp \quad ; \forall\, i<>j$$

Ceci évite de remettre en cause l'état pris initialement.

Etape:1 Tant que la séquence échantillon n'est pas épuisée on réalise les opérations suivantes:

a) Tant qu'il n'y a pas de contradictions entre modèle et échantillon on parcourt la séquence et on établit éventuellement les transitions entres états.
L'établissement d'une transition est réalisée vers un des états donnant une sortie ayant une probabilité maximum. Les autres alternatives mises en mémoires sont constituées par toutes les transitions possibles vers les états donnant des sorties yz telles que:

$$pk(yz) >= pkmax-fp$$

b) Lorsqu'une contradiction se présente, on essaye de la résoudre par les alternatives qui ont été mémorisées. Chaque alternative testée est systématiquement éliminée de la liste. Deux cas de modification sont à envisager suivant le résultat de ces essais:

1) On obtient un modèle qui permet de lever la contradition actuelle. Celui-ci est alors adopté comme nouveau modèle et on continue la construction à l'étape 1.

2) Toutes les possibilités de transition ont été épuisées sans avoir résolu la contradiction. Une procédure de recherche de l'état à dupliquer est alors lancée. Le modèle se voit alors ajouter un état supplémentaire homologue de celui créant la contradiction.

On réinitialise le modèle en supprimant toutes les transitions entre états. La construction du modèle est alors reprise à partir de l'instant initial mais avec un nombre d'états plus grand.

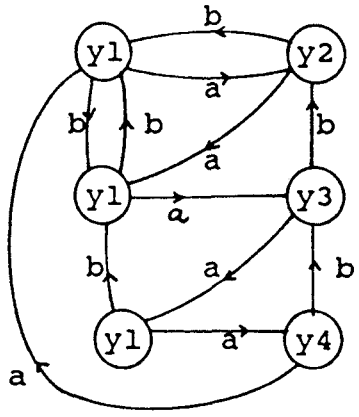
Etape:2 Vérification des alternatives restantes:
Lorsqu'on arrive à la fin de la séquence échantillon, des alternatives de transitions non encore testées peuvent subsister. L'essai de ces alternatives peut donner d'autres modèles acceptant l'échantillon et non isomorphiques au premier obtenu.

7.2. Procédure de résolution des contradictions

La recherche de l'état origine de la contradiction est réalisée de la même manière que dans le cas d'observations certaines. Mais des erreurs peuvent alors apparaître puisqu'on utilise un modèle non necessairement juste au niveau des sorties générées.

Exemple:

Soit l'automate réel suivant et un échantillon probabiliste:



instant	1	2	3	4	5	6	7	8	9	10
entrée	b	a	b	a	a	a	a	a	a	b
p(y1)	0.2	0.1	0.5	0.0	1.0	0.0	0.3	0.2	1.0	1.0
p(y2)	0.2	0.9	0.5	1.0	0.0	0.2	0.3	0.2	0.0	0.0
p(y3)	0.2	0.0	0.0	0.0	0.0	0.4	0.3	0.2	0.0	0.0
p(y4)	0.2	0.0	0.0	0.0	0.0	0.4	0.1	0.2	0.0	0.0

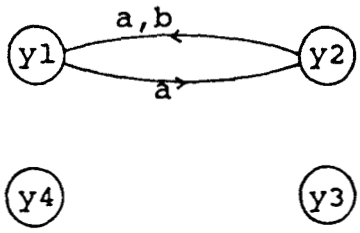
instant	11	12	13	14	15	16	17	18	19	20
entrée	b	a	b	b	b	a	a	a	b	a
p(y1)	1.0	0.0	0.9	0.2	0.5	0.0	1.0	0.0	0.0	1.0
p(y2)	0.0	0.8	0.1	0.2	0.5	1.0	0.0	0.0	1.0	0.0
p(y3)	0.0	0.2	0.0	0.2	0.0	0.0	0.0	1.0	0.0	0.0
p(y4)	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0

instant	21	22	23	24	25	26	27	28	29	30
entrée	a	a	a	b	b	b	b	a	a	a
p(y1)	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0
p(y2)	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
p(y3)	1.0	0.0	0.5	1.0	0.0	0.0	0.0	1.0	0.0	0.0
p(y4)	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.1

L'algorithme détecte une première contradiction à l'instant 6

```
modèle:          y1 -a-> y2      ; (q1 --> q2)
échantillon:     y1 -a-> y3,y4
```

Le modèle obtenu alors est le suivant:



Le résultat de la recherche de l'état à dupliquer donne q1 (ce qui n'est pas faux) mais la contradiction résolue qui sera mémorisée est l'une des suivantes:

```
1 ) y1 --> y2          2 ) y1 --> y2
    y1 --> y3              y1 --> y4
```

Cette incertitude risque de fausser les résultats de la procédure de recherche aux étapes ultérieures.

Dans cet exemple on obtient finalement, parmi les modèles possibles, le suivant, avec un état en plus non utilisé (q6).

		a	b
y1	q1	q2	q5
y2	q2	q7	q1
y3	q3	q1	q5
y4	q4	q1	q3
y1	q5	q4	q1
y2	q6	--	--
y1	q7	q3	q1

7.3.Conséquences d'une erreur faite sur l'état à dupliquer

La reprise de la construction du modèle après la duplication d'un état aboutit à deux conclusions possibles suivant que la précédente contradiction a été ou non résolue.

a) Dans le cas où la contradiction précédente n'a pas été résolue après l'ajout d'un état suplémentaire c'est que celui-ci ne constitue pas la solution. La sortie associée à cet état n'est pas la bonne. On cherche en effet à résoudre une contradiction à un instant donné de l'échantillon. L'erreur a été ici mise en évidence. La correction consisterait donc à changer la sortie associée à l'état ajouté et reprendre la construction du modèle.

b) La dernière modification effectuée a permis de résoudre la contradiction. Deux hypothèses sont alors possibles:

1) Cette modification constitue réellement la solution.

2) La modification est toujours erronée mais l'incertitude sur l'échantillon a permis de passer outre la contradiction. On ne possède malheureusement à ce stade de la construction aucun moyen de distinction entre ces deux hypothèses.

Nous proposons ci-après trois solutions possibles correspondant à trois niveau de décision quant au modèle à adopter.

a) Procédure A : Correction des erreurs:

On conserve la même procédure de recherche et de modification que dans le cas d'observations certaines à laquelle on ajoute une deuxième possibilité de modification pour corriger les erreurs. A chaque fois qu'une contradiction se présente on regarde si le dernier état ajouté a été utilisé (non isolé). Si cela n'est pas le cas la correction consiste à affecter à cet état une autre sortie, celle qui vient d'être trouvée par la procédure de recherche et reprendre la construction du modèle.

Cette façon de faire élimine les erreurs "immédiatement détectables". Suivant l'échantillon certaines erreurs ne peuvent être mises en évidences que bien plus tard dans la séquence. De plus une suite d'erreurs systématiques n'est pas exclue.

b) Procédure B: essais successifs par changement de la sortie associée au dernier état ajouté et correction des erreurs.

Dans cette procédure il n'est plus tenu compte des contradictions résolues précédemment; il n'y a donc plus de procédure de recherche de l'état à dupliquer. On essaye systématiquement tous les modèles possibles obtenus en affectant toutes les sorties au dernier état ajouté. Le modèle retenu est choisi parmis ceux acceptant la plus grande séquence échantillon et utilisant le moins d'états possible. Ce dernier critère de selection permet de mettre en évidence les erreurs qui sont toujours possibles.

Le temps de calcul necessaire est ici nettement plus important mais reste raisonnable.

c) procédure C: Essai exaustif de tous les modèles possibles avec un nombre d'états croissants.

Dans la méthode précédente on décide du modèle à adopter à chaque fois qu'un état est ajouté. Une solution extrême consiste à garder à chaque étape de l'algorithme l'ensemble des modèles à nombre d'états minimal acceptant une plus grande séquence échantillon. La décision de choix d'un modèle est alors rejetée à une étape extrême où celui-ci accepte toute la séquence échantillon. Cet algorithme a été développé en prenant une représentation des modèles sous la forme de machines de Mealy pour éviter l'élaboration des différentes combinaisons de sorties à affecter aux états ajoutés.

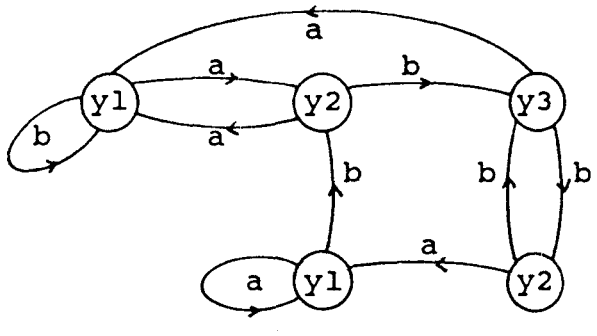
Dans ce cas le modèle est initialisé avec un seul état. On essaye alors de construire un modèle de façon à lui faire accepter la plus grande séquence échantillon. Tant que toute la séquence n'est pas acceptée on ajoute les états un par un et on recommence les essais.

Le temps de calcul devient alors très important. Des exemples à 4 ou 5 états peuvent necessiter facilement plus d'une heure sur un micro-ordinateur IBM PC! Ce sera donc l'algorithme B précédent qui sera finalement adopté ici.

On donne ci-après quelques exemples d'identification où les trois algorithmes précédents sont mis en oeuvre.

7.4.Exemples d'identification

a) Exemple 1: Le processus réel est le suivant.



L'échantillon n'est pas stable et est donné avec un facteur de précision fp de 0.1.

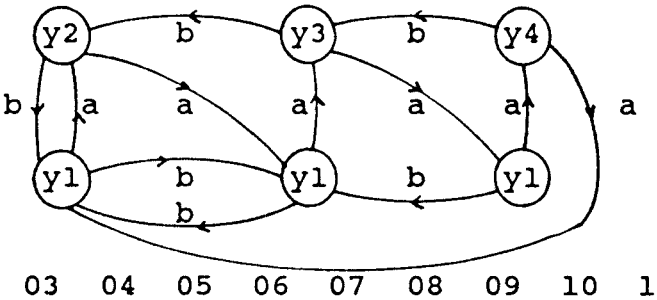
instant	1	2	3	4	5	6	7	8	9	10	11	12	13	14
entrée	b	b	b	a	a	a	b	a	a	b	b	b	b	b
p(y1)	0.3	0.9	0.3	0.5	0.5	0.0	0.3	0.3	0.0	0.0	0.0	0.1	0.1	0.0
p(y2)	0.3	0.1	0.4	0.5	0.0	1.0	0.3	0.4	1.0	0.0	1.0	0.1	0.8	0.0
p(y3)	0.4	0.0	0.3	0.0	0.5	0.0	0.4	0.3	0.0	1.0	0.0	0.8	0.1	1.0

instant	15	16	17	18	19	20	21	22	23	24	25	26	27	28
entrée	b	a	a	a	a	b	b	b	a	b	a	a	b	a
p(y1)	0.3	0.8	0.8	1.0	1.0	0.0	0.5	0.0	0.4	0.0	0.9	0.0	0.0	1.0
p(y2)	0.4	0.0	0.1	0.0	0.0	0.9	0.0	0.5	0.3	0.8	0.1	1.0	0.0	0.0
p(y3)	0.3	0.2	0.1	0.0	0.0	0.1	0.5	0.5	0.3	0.2	0.0	0.0	1.0	0.0

instant	29	30	31	32	33	34	35	36	37	38	39	40	41	42
entrée	b	b	b	a	a	a	a	a	b	b	a	a		
p(y1)	1.0	1.0	0.4	0.5	0.5	0.0	0.6	0.2	0.0	0.1	0.5	1.0		
p(y2)	0.0	0.0	0.3	0.5	0.0	0.8	0.2	0.6	0.4	0.7	0.3	0.0		
p(y3)	0.0	0.0	0.3	0.0	0.5	0.2	0.2	0.2	0.6	0.2	0.2	0.0		

L'échantillon est ici complet; l'algorithme donne un modèle réduit identique au processus.

b) Exemple 2: Le processus réel est le suivant. L'échantillon est donné avec un facteur de précision: fp=0.0

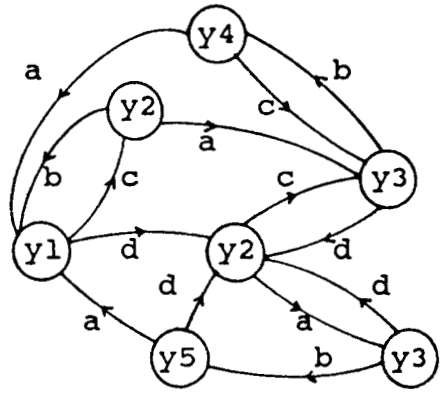


instant	01	02	03	04	05	06	07	08	09	10	11	12	13	14
entrée	b	a	b	a	a	a	a	a	a	b	b	a	b	b
p(y1)	0.4	0.1	0.5	0.0	1.0	0.0	0.5	1/4	1.0	1.0	1.0	0.0	0.9	1/4
p(y2)	0.3	0.9	0.5	1.0	0.0	0.2	0.5	1/4	0.0	0.0	0.0	0.8	0.1	1/4
p(y3)	0.3	0.0	0.0	0.0	0.0	0.6	0.0	1/4	0.0	0.0	0.0	0.2	0.0	1/4
p(y4)	0.0	0.0	0.0	0.0	0.0	0.2	0.0	1/4	0.0	0.0	0.0	0.0	0.0	1/4

instant	15	16	17	18	19	20	21	22	23	24	25	26	27	28
entrée	b	a	a	a	b	a	a	a	a	b	b	b		
p(y1)	0.5	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0		
p(y2)	0.5	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0		
p(y3)	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.5	1.0	0.0	0.0		
p(y4)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0		

L'algorithme donne ici un modèle identique au processus. La qualité de l'échantillon est assez bonne (0.261).

c) Exemple 3: modèle stable. Les transitions de stabilisation de la forme (yi,x,yi) sont implicites dans le processus réel représenté ci-dessous. fp=0.0



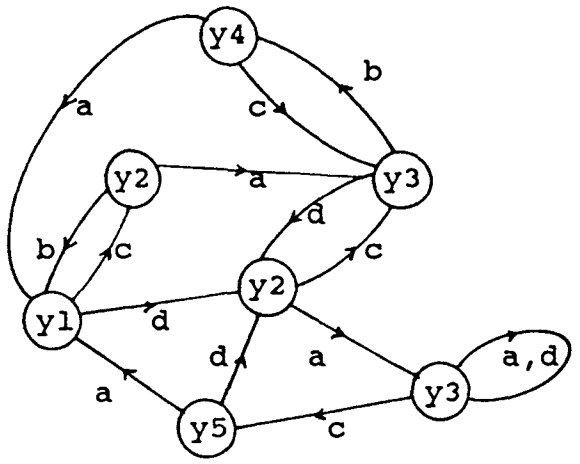
instant	01	02	03	04	05	06	07	08	09	10	11	12	13	14
entrée	a	c	b	d	a	b	d	c	d	c	b	a	c	a
p(y1)	0.2	0.2	0.5	0.0	0.1	0.2	0.1	0.3	0.0	0.0	0.0	0.5	0.1	0.0
p(y2)	0.2	0.3	0.2	1.0	0.1	0.2	0.3	0.3	1.0	0.0	0.0	0.5	0.6	0.0
p(y3)	0.2	0.1	0.1	0.0	0.8	0.1	0.2	0.3	0.0	1.0	0.5	0.0	0.1	1.0
p(y4)	0.2	0.2	0.1	0.0	0.0	0.1	0.2	0.1	0.0	0.0	0.5	0.0	0.1	0.0
p(y5)	0.2	0.2	0.1	0.0	0.0	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.1	0.0

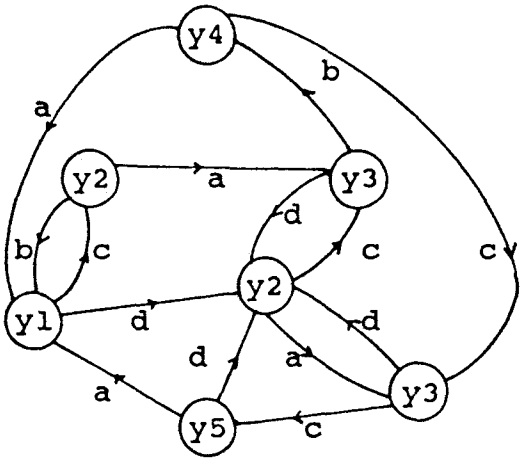
instant	15	16	17	18	19	20	21	22	23	24	25	26	27	28
entrée	b	c	d	a	d	a	b	a	d	a	b	a	d	c
p(y1)	0.1	0.2	0.2	0.2	0.2	0.2	0.0	1.0	0.1	0.2	0.0	1.0	0.0	0.2
p(y2)	0.2	0.2	0.2	0.2	0.2	0.2	0.0	0.0	0.3	0.2	0.0	0.0	1.0	0.2
p(y3)	0.3	0.2	0.2	0.2	0.2	0.6	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.2
p(y4)	0.3	0.2	0.2	0.2	0.2	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	0.2
p(y5)	0.1	0.2	0.2	0.2	0.2	0.0	1.0	0.0	0.2	0.2	1.0	0.0	0.0	0.2

instant	29	30	31	32	33	34	35	36	37	38	39	40	41	42
entrée	d	a	b	a	d	c	b	c	d	c	b	a	c	a
p(y1)	0.1	0.0	0.1	0.3	0.0	0.0	0.0	0.5	0.0	0.0	0.5	0.5	0.0	0.0
p(y2)	0.9	0.0	0.1	0.2	1.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	1.0	0.0
p(y3)	0.0	0.8	0.1	0.1	0.0	1.0	0.0	0.5	0.0	0.5	0.0	0.0	0.0	1.0
p(y4)	0.0	0.1	0.1	0.2	0.0	0.0	0.8	0.0	0.5	0.0	0.5	0.0	0.0	0.0
p(y5)	0.0	0.1	0.6	0.2	0.0	0.0	0.2	0.0	0.0	0.5	0.0	0.5	0.0	0.0

instant	43	44	45	46	47	48	49	50	51	52	53	54	55	56
entrée	b	a	d	a	b	d	a	b	a	c	b	c	a	b
p(y1)	0.0	0.2	0.2	0.1	0.0	0.1	0.0	0.1	1.0	0.0	1.0	0.0	0.0	0.0
p(y2)	0.0	0.2	0.2	0.1	0.0	0.8	0.1	0.1	0.0	1.0	0.0	1.0	0.0	0.0
p(y3)	0.0	0.2	0.2	0.6	0.0	0.1	0.7	0.1	0.0	0.0	0.0	0.0	1.0	0.0
p(y4)	1.0	0.2	0.2	0.1	0.1	0.0	0.1	0.2	0.0	0.0	0.0	0.0	0.0	1.0
p(y5)	0.0	0.2	0.2	0.1	0.9	0.0	0.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0

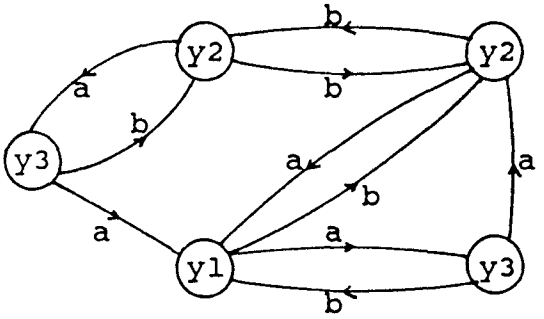
L'échantillon est ici encore imcomplet; l'algorithme donne, en plus d'un modèle identique au processus, les 2 modèles suivants:



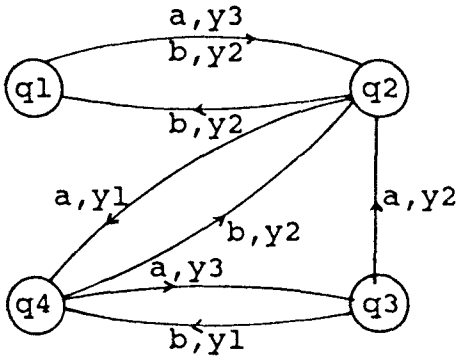


d) Exemple 4: Le processus réel est représenté ci-dessous sous forme d'une machine de Moore et son équivalence sous forme de modèle de Mealy. fp=0.0

Modèle de Moore



Modèle de Mealy



instant	01	02	03	04	05	06	07	08	09	10	11	12	13	14
entrée	a	b	b	b	a	a	a	a	a	a	a	b	b	a
p(y1)	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
p(y2)	0.5	0.0	0.5	0.5	0.5	0.0	0.5	0.5	0.0	0.5	0.5	0.5	1.0	0.0
p(y3)	0.5	0.0	0.5	0.5	0.5	0.0	0.5	0.5	0.0	0.5	0.5	0.5	0.0	0.0

instant	15	16	17	18	19	20	21	22	23	24	25	26	27	28
entrée	b	b	a	b	a	a	a	a	b	a	b	b	a	a
p(y1)	0.0	0.5	0.5	0.5	0.5	0.5	0.5	0.0	0.0	0.0	0.0	0.0	1.0	0.0
p(y2)	1.0	0.5	0.0	0.5	0.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0
p(y3)	0.0	0.0	0.5	0.0	0.5	0.5	0.5	0.0	0.0	1.0	0.0	0.0	0.0	1.0

instant	29	30	31	32	33	34	35	36	37	38	39	40	41	42
entrée	b	b	b	a	a	a	a	a	a	a	b	b	a	b
p(y1)	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
p(y2)	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0
p(y3)	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

instant	43	44	45	46	47	48
entrée	b	a	b	a	a	a
p(y1)	0.0	0.0	0.0	0.0	1.0	0.0
p(y2)	1.0	0.0	1.0	0.0	0.0	0.0
p(y3)	0.0	1.0	0.0	1.0	0.0	1.0

Cet échantillon est particulier. L'algorithme A fait une erreur et donne plusieurs modèles avec un état en plus dont le processus réel où l'état en trop n'est pas utilisé.

Les résultats de ces quatre exemples sont regroupés dans le tableau suivant pour une meilleure comparaison.

n°	nbe	X/Y	éch.	Qualité	Proc. A	Proc. B	Proc. C
Ex 1	5	2/3	40	0.468	18 s	20 s	30 mn
Ex 2	6	2/4	26	0.261	8 s	17 s	> 1h30
Ex 3	7	4/5	56	0.457	9 s	18 s	> 2 h
Ex 4	5	2/3	48	0.197	erreur	3mn25	2 mn 10

On peut remarquer les différences de temps de calcul entre les procédures A,B d'une part et C d'autre part.

7.5.Complexité en temps de calcul:

Le temps de calcul dépend de cinq facteurs qui sont par ordre d'importance:

- Le nombre d'états ns;
- La qualité de l'échantillon I;
- Le nombre de sorties ny;
- La longueur de l'échantillon l ;
- Le nombre d'entrées nx.

Ce temps est difficilement chiffrable. Les algorithmes A et B sont beaucoup plus sensibles au nombre d'états ajoutés (ns-ny) et à la qualité de l'échantillon alors que l'algorithme C est essentiellement sensible au nombre d'états total ns et à la qualité de l'échantillon.

7.6. Quelques amélioration des algorithmes:

Le nombre d'itérations nécessaires à l'obtention d'un modèle étant important un filtrage des alternatives de transitions à essayer est réalisé de la manière suivante:

L'alternative qi —> qj en remplacement de qi —> qk n'est pas considérée si qj et qk ont même sortie associée et si les deux états sont isolés à ce moment là (pas de transition partant de qj ni de qk).

La construction orientée utilisée dans l'algorithme d'identification à partir d'un échantillon certain n'est pas nécessaire ici puisque de toute façon toutes les possibilités de transition doivent être essayées.

La prise en compte de la stabilité de l'échantillon est réalisée de la même manière que dans le premier algorithme avec des observations certaines. Ceci constitue alors une autre source de contradictions et un filtrage des alternatives de transitions à essayer en ne conservant que celles préservant la stabilité du modèle.

8. APPLICATION A L'IDENTIFICATION DE LA PARTIE PREACTIONNEURS D'UNE PARTIE OPERATIVE

Dans le chapitre IV précédent nous avons vu comment il était possible d'identifier les commandes à l'aide d'une procédure de classification automatique et obtenir une séquence d'entrée/ sortie (O/p(C)) probabiliste de la partie "préactionneurs" pour son identification.

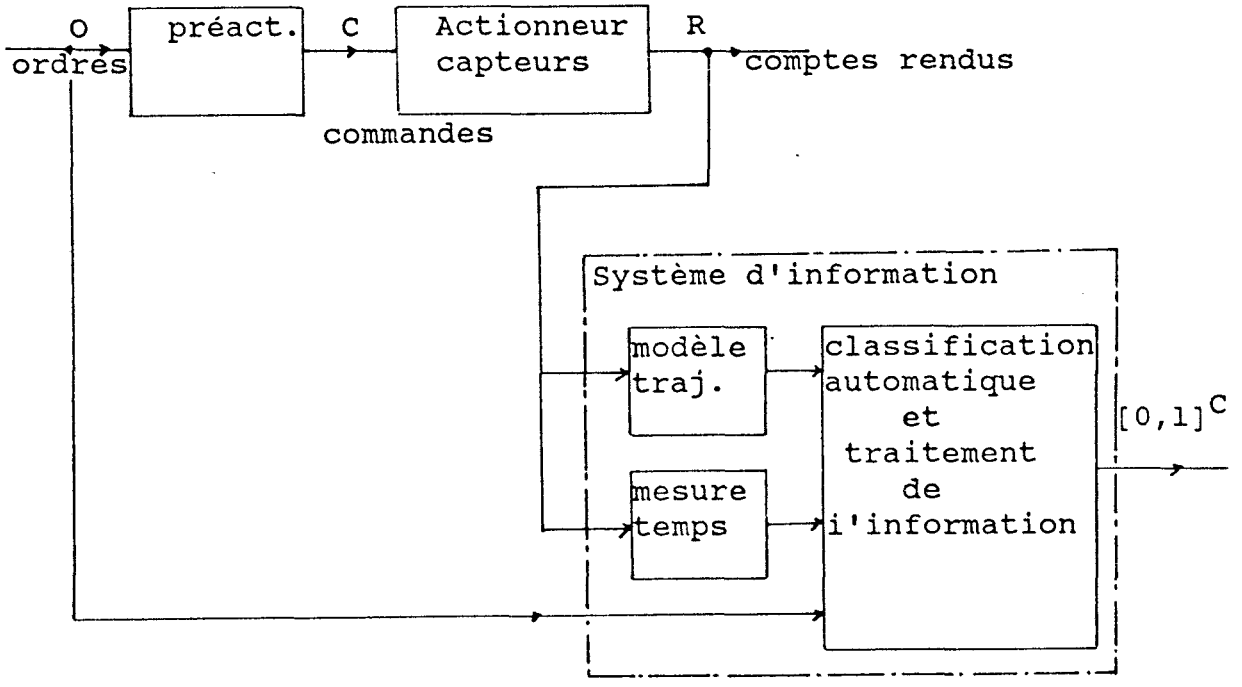


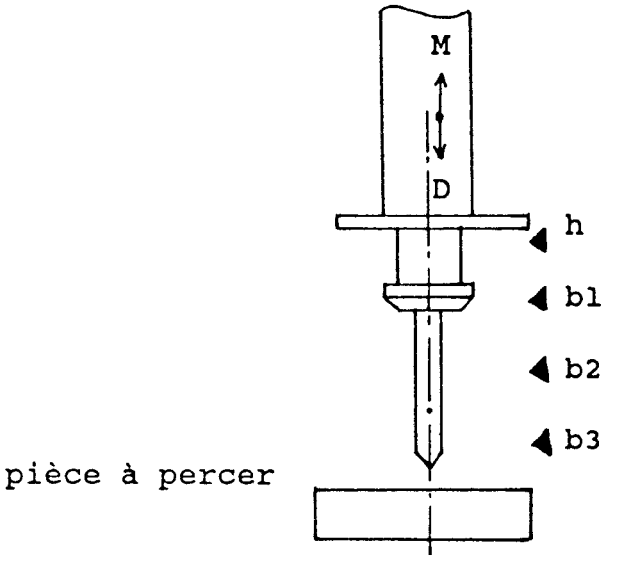
Figure V.8 Détermination d'une séquence o/p(c) pour l'identification de la partie 'préactionneurs' d'une sous partie opérative.

Nous allons voir maintenant dans quelle mesure on peut obtenir le modèle de cette partie préactionneurs pour compléter l'apprentissage automatique du modèle de la partie opérative.

Les avantages et inconvéniants du modèle obtenu sur l'efficacité du diagnostic ont été discutés au chapitre IV.

Pour l'illustration nous prendrons l'exemple d'une sous partie opérative décrite à la figure V.9 à défaut d'un exemple réel.

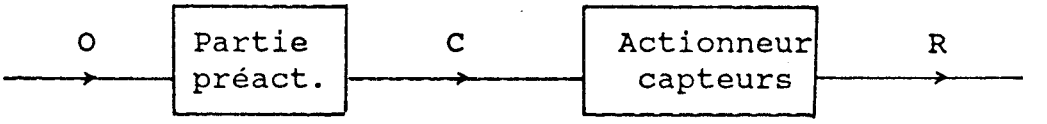
Figure V.9 Exemple de sous partie opérative



Une tête de perçage coulissante dans un batis est commandée en montée-descente. La position de la tête est repérée par quatre capteurs (h, b_1, b_2, b_3) de type tout ou rien.

Les alphabets O et R sont supposés connus et sont donnés à la figure V.10 ci-dessous.

Figure V.10



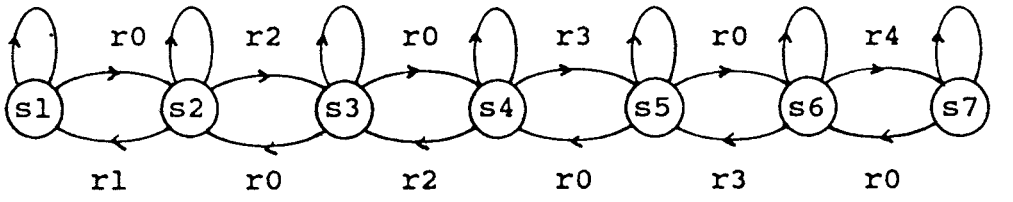
O={a,m,G,P,M,D} R={r1,r0,r2,r3,r4}

avec $r1=h.\overline{b1}.\overline{b2}.\overline{b3}$; $r0=\overline{h}.\overline{b1}.\overline{b2}.\overline{b3}$; $r2=\overline{h}.b1.\overline{b2}.\overline{b3}$

$r3=\overline{h}.\overline{b1}.b2.\overline{b3}$; $r4=\overline{h}.\overline{b1}.\overline{b2}.b3$

La procédure d'identification de la trajectoire permet d'obtenir le modèle de trajectoire de la figure V.11.

Figure V.11



Ce graphe est complété par un sommet initial (In) pour permettre l'initialisation du modèle de la trajectoire qui va être utilisé au cours de la phase de classification (figure V.12).

Les sommets déterministes du graphes sont ceux n'ayant pas r0 comme compte rendu.

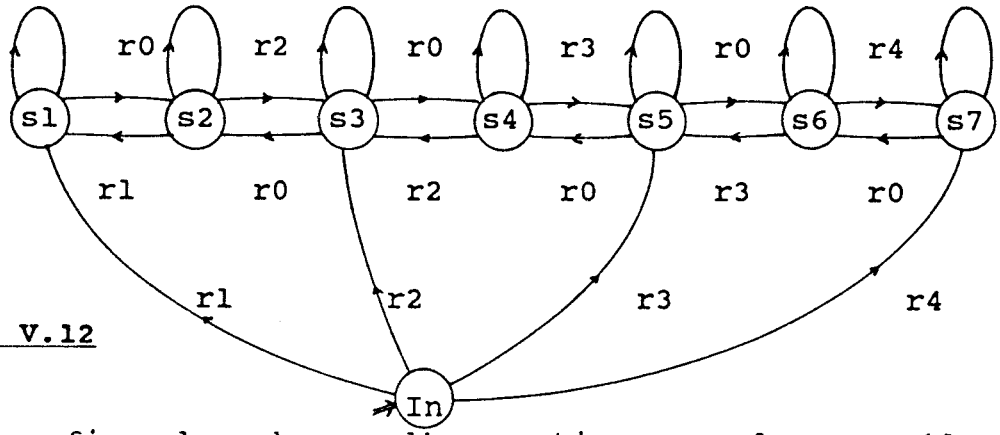


Figure V.12

En fin de phase d'apprentissage, la procédure de classification à permis de déterminer les commandes ou vitesses:

C = { Stop, Mgv, Mpv, Dgv, Dpv }

Le modèle de la trajectoire est alors complété par les étiquettes associées aux arcs soit les modules de vitesse correspondants et les temps de couverture (avec et sans changement de sens).

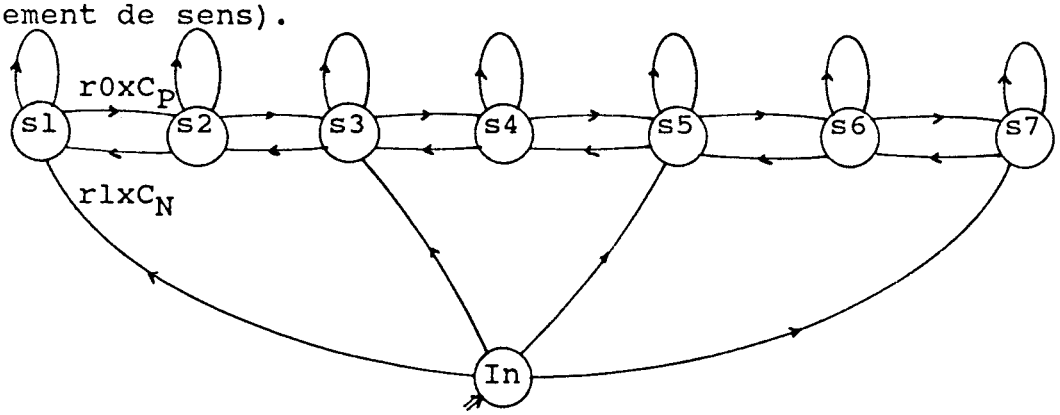


Figure V.13 C_P = {Mgv, Mpv}; C_N = {Dgv, Dpv}

L'utilisation de ce modèle de trajectoire ainsi complété et la procédure de classification va nous permettre, à partir d'une séquence O/R/t d'entrée/sortie de la partie opérative, d'obtenir une séquence O/p(C) probabiliste qui servira à l'identification de la partie "préactionneurs"; celle-ci étant considérée comme une machine séquentielle.

Un exemple de séquence est donné ci-après:

instant	01	02	03	04	05	06	07	08	09	10	11	12	13	14
entrée	a	m	a	M	m	P	a	D	m	G	a	M	m	P
p(Stop)	0.2	0.0	0.4	0.2	0.0	0.0	0.6	0.3	0.0	0.0	0.2	0.6	0.0	0.0
p(Mgv)	0.2	0.0	0.1	0.2	0.8	0.2	0.1	0.2	0.0	0.0	0.2	0.1	0.7	0.4
p(Mpv)	0.2	0.0	0.2	0.2	0.2	0.8	0.1	0.2	0.0	0.0	0.2	0.1	0.3	0.6
p(Dgv)	0.2	0.9	0.1	0.2	0.0	0.0	0.1	0.2	0.3	0.9	0.2	0.1	0.0	0.0
p(Dpv)	0.2	0.1	0.2	0.2	0.0	0.0	0.1	0.1	0.7	0.1	0.2	0.1	0.0	0.0

instant	15	16	17	18	19	20	21	22	23	24	25	26	27	28
entrée	D	M	G	a	D	m	P	a	m	M	D	M	a	G
p(Stop)	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.6	0.2
p(Mgv)	0.0	0.2	0.6	0.2	0.2	0.0	0.0	0.2	0.0	0.2	0.0	0.2	0.1	0.2
p(Mpv)	0.0	0.8	0.4	0.2	0.2	0.0	0.0	0.2	0.0	0.8	0.0	0.8	0.1	0.2
p(Dgv)	0.1	0.0	0.0	0.2	0.2	0.9	0.4	0.1	0.3	0.0	0.3	0.0	0.1	0.2
p(Dpv)	0.9	0.0	0.0	0.2	0.2	0.1	0.6	0.1	0.7	0.0	0.7	0.0	0.1	0.2

instant	29	30	31	32	33	34	35	36	37	38	39	40	41	42
entrée	m	D	M	a	P	m	D	a	M	G	m	a	D	m
p(Stop)	0.0	0.0	0.0	0.5	0.6	0.0	0.0	0.5	0.3	0.4	0.0	0.5	0.4	0.0
p(Mgv)	0.9	0.0	0.8	0.1	0.1	0.3	0.0	0.1	0.2	0.1	0.6	0.2	0.1	0.0
p(Mpv)	0.1	0.0	0.2	0.1	0.1	0.7	0.0	0.2	0.2	0.2	0.3	0.1	0.2	0.0
p(Dgv)	0.0	0.7	0.0	0.1	0.1	0.0	0.4	0.1	0.1	0.1	0.0	0.1	0.1	0.6
p(Dpv)	0.0	0.3	0.0	0.2	0.1	0.0	0.6	0.1	0.2	0.2	0.0	0.1	0.2	0.4

instant	43	44	45	46	47	48	49	50	51	52	53	54	55	56
entrée	M	D	P	M	D	M	G	P	D	a	G	M	m	D
p(Stop)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.6	0.6	0.0	0.0
p(Mgv)	0.6	0.0	0.0	0.1	0.0	0.2	0.9	0.4	0.0	0.0	0.1	0.1	0.8	0.0
p(Mpv)	0.4	0.0	0.0	0.9	0.0	0.8	0.1	0.6	0.0	0.1	0.1	0.1	0.2	0.0
p(Dgv)	0.0	0.7	0.4	0.0	0.4	0.0	0.0	0.0	0.3	0.0	0.1	0.1	0.0	0.8
p(Dpv)	0.0	0.3	0.6	0.0	0.6	0.0	0.0	0.0	0.7	0.0	0.1	0.1	0.0	0.2

instant	57	58	59	60	61	62	63	64	65	66	67	68	69	70
entrée	a	P	m	a	G	m	a	P	m	G	a	P	m	a
p(Stop)	0.4	0.5	0.0	0.4	0.6	0.0	0.7	0.5	0.0	0.0	0.4	0.4	0.0	0.4
p(Mgv)	0.2	0.2	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.1	0.1	0.0	0.1
p(Mpv)	0.2	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.2
p(Dgv)	0.1	0.1	0.3	0.3	0.2	0.8	0.2	0.1	0.4	0.8	0.2	0.2	0.4	0.1
p(Dpv)	0.1	0.1	0.7	0.3	0.2	0.2	0.1	0.1	0.6	0.2	0.2	0.2	0.6	0.2

instant	71	72	73	74	75	76	77	78	79	80	81	82	83	84
entrée	M	m	a	G	m	a	D	P	m	G				
p(Stop)	0.4	0.0	0.4	0.4	0.0	0.5	0.5	0.5	0.0	0.0				
p(Mgv)	0.1	0.2	0.2	0.2	0.7	0.2	0.2	0.2	0.0	0.0				
p(Mpv)	0.2	0.8	0.2	0.2	0.3	0.1	0.1	0.1	0.0	0.0				
p(Dgv)	0.1	0.0	0.1	0.1	0.0	0.1	0.1	0.1	0.4	0.6				
p(Dpv)	0.2	0.0	0.1	0.1	0.0	0.1	0.1	0.1	0.6	0.4				

Qualité de l'échantillon: I = 0.55; L'échantillon est considéré comme stable. Le modèle obtenu est représenté figure V.14 ci-dessus.

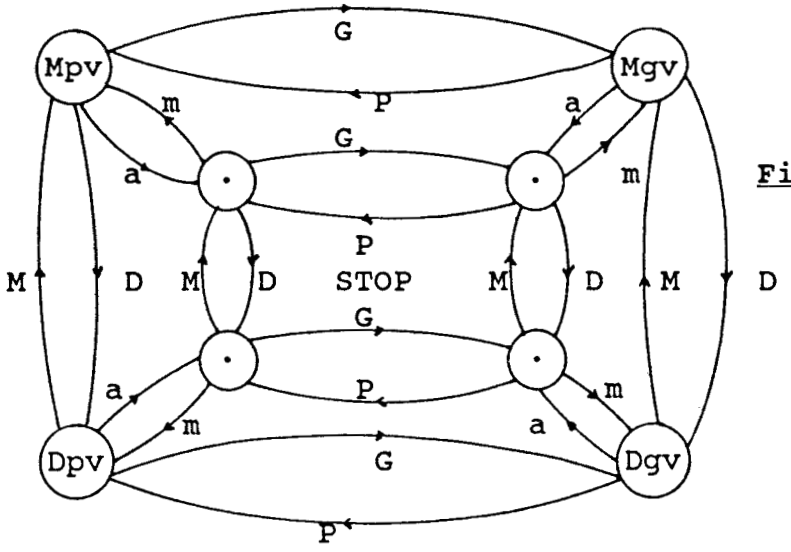


Figure V.14

On peut faire les remarques suivantes:

a) La commande Stop est souvent confondue avec les autres, ce qui traduit le problème de détermination des temps d'arrêt évoqué dans le chapitre précédent. Dans ce cas l'incertitude est souvent importante.

On pourrait croire que l'on pourrait obtenir le modèle de la partie préactionneurs en laissant une incertitude complète aux endroits où il existe un problème de temps d'arrêt à déterminer. Ceci n'est pas toujours le cas.

b) Comportement en cas d'erreurs du système d'information: Si les probabilités données par l'échantillon et donc par le système d'information sont erronnées le comportement du processus risque d'être non déterministe. L'algorithme d'identification boucle alors indéfiniment, une première solution consiste à réajuster le facteur fp et recommencer l'identification. Le résultat serait un modèle ayant une distance plus grande avec l'échantillon. La perte d'informations ainsi occasionnée risque d'être très nuisible.

Une deuxième solution consiste à rechercher l'instant de contradiction où il y a non déterminisme et modifier l'échantillon à cet instant par une incertitude complète.

CONCLUSION GENERALE

La modélisation du comportement de la partie opérative telle que décrite dans le chapitre II nous a permis de proposer plusieurs procédures de test en ligne de plus en plus complexes mais aussi plus efficaces.

La dernière procédure étudiée, intègrant le maximum d'informations sur la PO, permet une importante couverture de détection des défaillances affectant les actionneurs, préactionneurs et leurs interfaces.

L'architecture d'implantation du test proposée constitue une solution parmis d'autres. Celle-ci a l'avantage d'éviter la duplication des interfaces PC/PO et de réaliser ainsi un compromis sécurité/fiabilité. Parmis les autres solutions possibles, le choix d'une architecture informatique multiprocesseurs tolérante aux fautes avec intégration du test peut améliorer la sûreté de fonctionnement de la structure.

Dans les chapitres 4 et 5, nous avons étudié les possibilités d'apprentissage automatique du modèle de comportement de la partie opérative dans le but, en particulier, de créer automatiquement la structure de données nécessaire au test.

L'apprentissage des classes de commande peut être rendu difficile par la présence de classes d'arrêt ou changement d'ordres au cours de l'évolution. Ce problème reste posé dans le cas général.

Nous avons proposé dans le chapitre 5 une procédure d'identification des machines séquentielles dans le cas déterministe. Nous l'avons généralisée afin de prendre en compte des incertitudes sur les sorties lorsque celles-ci sont observées à travers un système d'information probabiliste. Ceci permet d'obtenir un modèle déterministe malgré la non observabilité des commandes et les imperfections de la classification réalisée au chapitre 4.

L'obtention du modèle de comportement réel de la PO peut être très utile au concepteur en particulier pour la validation par simulation des logiciels de commande, l'étude des modes de défaillance et l'affinement des stratégies de conduite.

ANNEXE

RAPPELS SUR LES MACHINES SEQUENTIELLES [1], [7], [15], [16]

Définition A1: Une machine séquentielle est définie par le quintuplé: $M = (X,Q,Y,\delta,\lambda)$ avec:
$X=\{x1,x2,...,xu\}$ un ensemble non vide appelé **alphabet d'entrée.**
$Y=\{y1,y2,...,yv\}$ est **l'alphabet de sortie.**
$Q=\{q1,q2,...,qk\}$ est l'ensemble fini des **états internes.**
δ: QxX -> Q est la **fonction état suivant ou de transition.**
λ: QxX -> Y (pour une machine de mealy) ou λ: Q -> Y (pour une machine de Moore) constitue la **fonction de sortie.**

Le graphe de fluence et la table de transition constituent des représentations graphiques habituelles des machines séquentielles.

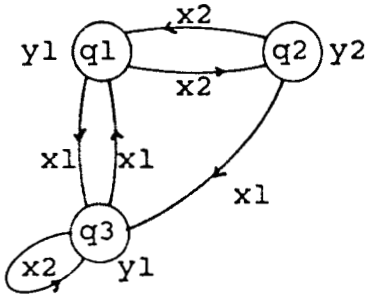
Le graphe de fluence comporte un ensemble de sommets, chacun d'eux étiqueté par un état interne de l'automate et un ensemble d'arcs orientés, appelés également transitions, reliant les états entre eux. Dans une machine de Mealy, les sorties complètent les étiquettes associées aux transitions. Dans une machine de Moore, les sorties complètent les étiquettes associées aux sommets ou états.

La table de transition constitue donc une représentation tabulaire du graphe de fluence. Ce graphe peut être représenté d'une autre manière par une matrice de transitions QxQ dont les éléments représentent les fonctions δ et λ.

Exemple:

Représentation d'une machine de Moore $M=(X,Q,Y,\delta,\lambda)$ où
$X=\{x1,x2\}$; $Y=\{y1,y2\}$; $Q=\{q1,q2,q3\}$

Graphe de fluence **Table de transition** **Matrice de transition**

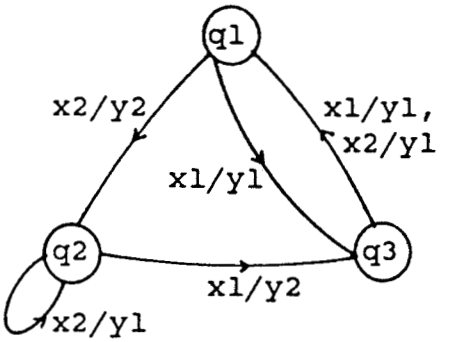


	x1	x2	
q1	q3	q2	y1
q2	q3	q1	y2
q3	q1	q3	y1

	q1	q2	q3
q1	(-----)	(x2,y2)	(x1,y1)
q2	(x2,y1)	(-----)	(x1,y1)
q3	(x1,y1)	(-----)	(x2,y1)

Représentation d'une machine de Mealy $M=(X,Q,Y,\delta,\lambda)$

Graphe de fluence Table de transition Matrice de transition



	x1	x2
q1	q3/y1	q2/y2
q2	q3/y2	q2/y1
q3	q1/y1	q1/y1

$$\begin{bmatrix} (-----) & (x2,y2) & (x1,y1) \\ (-----) & (x2,y1) & (x1,y2) \\ (x1,y1) & (-----) & (-----) \\ +(x2,y1) & & \end{bmatrix}$$

Machine séquentielle fortement connexe:

Définition A2: Une machine séquentielle $M=(X,Q,Y,\delta,\lambda)$ est dite fortement connexe si pour tout qi, qj $\in$ Q, il existe une séquence d'entrée $\underline{x}$ qui appliquée à partir de qi permet d'atteindre qj soit: $\delta(qi,\underline{x}) = qj$

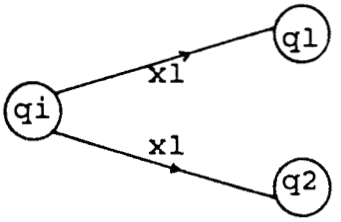
Dans la pratique cette propriété est en général vérifiée pour tout automatisme industriel. Une machine séquentielle fortement connexe ne possède donc ni état source ni état puits ni sous ensemble d'états isolés (ou sous machine).

Machine séquentielle déterministe:

Définition A3: Une machine séquentielle est dite non déterministe si sa fonction de transition est une application multivoque.

prend alors ses valeurs dans P(Q), l'ensemble des parties de Q; à un couple (qi,xi) $\in$ QxX peut correspondre un ensemble d'états, ce qui s'exprime graphiquement par:

$\delta(qi,x1) = \{q1,q2\}$



Machine séquentielle incomplètement spécifiée:

Dans le cas où la fonction n'est pas définie pour tout couple (q,x) $\in$ QxX l'automate est dit incomplètement spécifié ou incomplet.

Signification de la matrice de transition d'ordre supérieur:

Soit $T^2=TxT$ la matrice d'ordre 2 d'une machine séquentielle, ses éléments Tij représentent l'ensemble des séquences d'entrée/sortie $(\underline{x},\underline{y})$ de longueur 2 que l'on peut observer lorsqu'on passe de l'état qi à l'état qj.

De même la matrice de transition d'ordre n permettra de déterminer toutes les séquences d'entrée/sortie de longueur n générées par la machine en passant d'un état qi à un état qj.

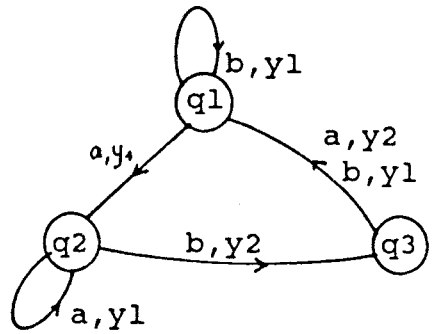
La matrice produit est obtenue de la manière suivante:
Soient A et B deux matrices txt;
C = AxB est une matrice txt dont l'élément c_{ij} s'écrit:

$$c_{ij} = a_{i1}.b_{1j} + a_{i2}.b_{2j} + \ldots + a_{it}.b_{tj}$$

$$\text{soit } c_{ij} = \sum_{k=1}^{t} a_{ik}.b_{ki}$$

Exemple

$$T = \begin{bmatrix} (b,y1) & (a,y1) & \emptyset \\ \emptyset & (a,y1) & (b,y2) \\ (a,y2) & \emptyset & \emptyset \\ +(b,y1) & & \end{bmatrix}$$



$$T^2 = \begin{bmatrix} (bb,y1y1) & (aa,y1y1) & (ab,y1y2) \\ & +(ba,y1y1) & \\ \hline (bb,y2y1) & (aa,y1y1) & (ab,y1y2) \\ +(ba,y2y2) & & \\ \hline (bb,y1y1) & (ba,y1y1) & \emptyset \\ +(ab,y2y1) & +(aa,y2y1) & \end{bmatrix}$$

ETATS EQUIVALENTS:

Définition A4: Le comportement d'un état q Q est défini par l'ensemble B(q) des séquences d'entrée/sortie ($\underline{x}$,$\underline{y}$), $\underline{x} \in X^*$, $\underline{y} \in Y^*$, telles que $\lambda(q,\underline{x})=\underline{y}$:

$$B(q) = \{ (\underline{x},\underline{y}) \mid \lambda(q,\underline{x}) = \underline{y} \}$$

Définition A5: Deux états q1, q2 sont dits équivalents notés $q1 \equiv q2$, si en appliquant une séquence d'entrée $\underline{x}$ à partir de q1 et q2 on obtient la même séquence de sortie $\underline{y}$.

$$\lambda(q1,\underline{x}) = \lambda(q2,\underline{x}) = \underline{y} \; ; \; \forall \underline{x} \in X^* \quad ===> \quad q1 \equiv q2$$

En d'autres termes, deux états q1 et q2 sont équivalents s'ils ont le même comportement B(q1)=B(q2). Ainsi deux états sont équivalents s'il n'existe aucun moyen de les distinguer par observation des séquences d'entrée/sortie de la machine.

Deux états non équivalents sont dits différentiables. Deux états q1, q2 sont donc différentiables si et seulement si il existe une séquence d'entrée $\underline{x}$ telle que, appliquée à partir de q1 et q2 donne deux séquences de sortie différentes.

L'équivalence entre états est une vraie relation d'équivalence au sens mathématique. Par contre la relation d'états différentiables n'est pas une relation d'équivalence.

Chaque relation d'équivalence induit une partition sur l'ensemble des objets sur lesquels elle s'applique. Dans notre cas une famille de classes d'états d'équivalence. On note l'ensemble des états équivalents à un état q par [q]

$$[q] = \{ qi \mid qi \underline{=} q \}$$

Corrollaire A1: $[qi] = [qj] \iff qi \underline{=} qj$
Si deux états qi, qj ont deux classes d'équivalence identiques alors ils sont équivalents et inversement.

ETATS k-EQUIVALENTS:

Définition A6: Deux états qi, qj sont k-équivalents notés $qi \underline{=}^k qj$ si $\lambda(qi,\underline{x}) = \lambda(qj,\underline{x}) = \underline{y}$ pour toute séquence d'entrée $\underline{x}$ de longueur $|\underline{x}| \leq k$.
Deux états non k-équivalents sont dits k-différentiables. il existe alors une séquence d'entrée $\underline{x}$ de longueur k telle que $\lambda(qi,\underline{x}) <> \lambda(qj,\underline{x})$

Lemme A1: Si deux états qi, qj sont k-équivalents alors ils sont aussi 1-équivalents pour tout $1 \leq k$. De même, si deux états sont k-différentiables, ils sont aussi 1-différentiables pour tout $1 \geq k$.

Lemme A2: Si $qi \underline{=} qj$ alors $qi \underline{=}^k qj \quad \forall k$
Un automate réduit comporte des états au maximum r-équivalents avec $r < (t-1)$ dans une machine de Mealy et $r < (t-2)$ dans une machine de Moore. Cet automate est dit "d'ordre r".
Deux états qi, qj d'une machine séquentielle sont donc équivalents s'ils sont:
(t-1)-équivalents dans une machine de Mealy ou
(t-2)-équivalents dans une machine de Moore, avec t=card(Q).
Ils sont différentiables s'ils sont t-différentiables dans une machine de Mealy ou (t-1)-différentiables dans une machine de Moore.

Pour deux états non équivalents on peut donc trouver une séquence d'entrée $\underline{x}$ de longueur maximum t = card(Q) qui appliquée à partir des deux états donne deux séquences de sortie différentes.

Lemme A3: Si une machine séquentielle M contient deux états différentiables qui sont k-équivalents, alors elle doit contenir aussi deux états k-équivalents mais (k+1)-différentiables.

Définition A7: La relation de k-équivalence détermine une partition sur l'ensemble des états de Q notée $[\underline{=}^k]$ ou d'une façon équivalente:

$$[\underline{=}^k] = \bigcup_{q \in Q} [q]_k$$

Théorème A1: Pour tout entier k, $[\underline{=}^k] = [\underline{=}]$ si et seulement si $[\underline{=}^{k+1}] = [\underline{=}^k]$

En d'autres termes si la partition déterminée par la (k+1)-équivalence est identique à celle obtenue pour la k-équivalence alors cette partition constitue la partition d'équivalence.

Théorème A2: Dans une machine séquentielle si tous les états sont tels que $qi \underline{=}^1 qj$ alors $qi \underline{=} qj \quad \forall qi,qj$

Machines séquentielles réduites:

Définition A8: une machine séquentielle est dite réduite s'il n'existe pas d'états équivalents dans sa définition. En d'autres termes $qi \underline{=} qj \implies qi = qj$. Chaque état n'est équivalent qu'à lui même.

Machines séquentielles équivalentes:

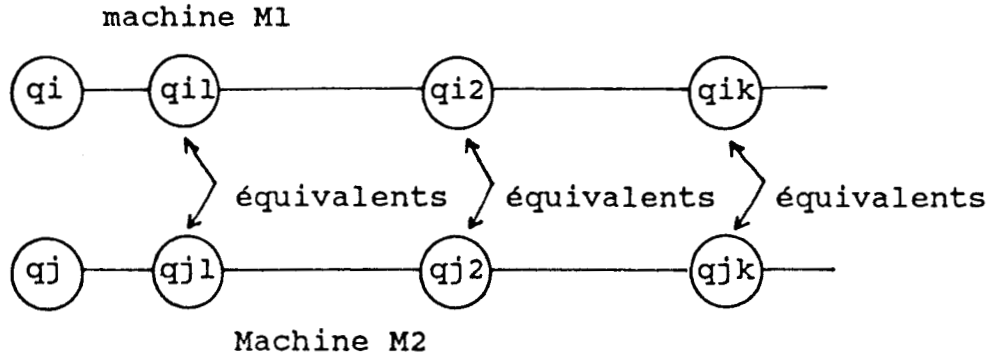
Définition A9: Soient deux automates M1 et M2 ayant les mêmes alphabets d'entrée et de sortie:

- M1 est équivalent et inclu dans M2 noté ($M1 \underline{C} M2$) si pour tout état de M1 il existe un état équivalent de M2.
- M1 et M2 sont équivalents ($M1 \underline{=} M2$) si $M1 \underline{C} M2$ et $M2 \underline{C} M1$. Ils sont dits différentiables dans le cas contraire.

Deux automates équivalents et réduits ont le même nombre d'états mais peuvent présenter des descriptions différentes du seul fait de leur numérotation. De tels automates sont donc isomorphiques à la numérotation des états près.

Lemme A4: Si M1 et M2 sont deux machines séquentielles fortement connexes et différentiables alors aucun état de M1 n'est équivalent à un état de M2.

Théorème A3: Si deux états qi, qj d'une machine séquentielle sont k-équivalents et si les états successeurs associés à toute séquence d'entrée de longueur k sont équivalents alors qi $\underline{=}$ qj.



Théorème A4: Si M_r est la forme réduite d'une machine M alors:

a) M_r est unique à un isomorphisme près.
b) M_r est équivalente à M; $M_r \underline{=} M$.
c) Il n'éxiste pas d'états équivalents dans M_r.
d) Aucune autre machine équivalente n'est plus petite (en nombre d'états) que M_r.

Théorème A5: Dans une machine séquentielle à t états pour chaque sous ensemble de r états ($2 \underline{\leq} r \underline{\leq} t$) il existe au moins deux états qui sont (t-r+1)-différentiables.

ETATS COMPATIBLES:

Lorsqu'on s'interesse aux machines séquentielles incomplètement spécifiées la notion d'équivalence cède la place à celle d'états compatibles. On dit dans ce cas qu'une séquence d'entrée est acceptable pour un état q si les transitions empreintées au cours de cette séquence sont toutes définies.

Définition A10: Deux états q1, q2 d'une machine séquentielle incomplètement spécifiée sont dits compatibles si l'application d'une séquence d'entrée quelconque mais acceptable à partir de q1 et q2 donne les mêmes séquences de sortie.
q1 et q2 sont incompatibles s'il existe une séquence d'entrée acceptable pour q1 et q2 telle que appliquée à partir de q1 et q2 donne des séquences de sortie différentes.

Machines séquentielles quasi-équivalentes:

Définition A11: Un état q1 est dit quasi-équivalent à q2 si toute séquence d'entrée acceptable à partir de q2 donne la même séquence de sortie lorsqu'elle est appliquée à partir de q1 et q2.

Définition A12: Une machine séquentielle M1 est dite quasi-équivalente à une machine M2 si pour chaque état q1 de M1 il correspond au moins un état q2 de M2 tel que q2 soit quasi-équivalent à q1.

M1 est quasi-équivalente à M2 s'il n'y a aucun moyen de les distinguer par leur comportement entrée/sortie par l'utilisation des seules séquences acceptables par les états de M2. A noter que la quasi-équivalence n'est pas une relation symétrique.

DETERMINATION DE L'ETAT D'UNE MACHINE SEQUENTIELLE: [1]

La détermination de l'état d'une machine séquentielle connue à partir de séquences d'observation entrée/sortie est un problème couramment rencontré. L'analyse des matrices de transition d'ordre supérieur permet de déterminer les séquences caractéristiques recherchées.

Lemme A6: Lorsque plusieurs colonnes d'une matrice de transition d'ordre r comportent les mêmes suites d'entrées correspondant à des suites de sorties différentes, ces suites d'entrées permettent d'apprendre l'état de l'automate. Elles sont appelées **"séquences d'apprentissage de l'état de l'automate"**.

Lemme A7: Dans toute machine séquentielle fortement connexe et d'ordre r, il existe des séquences d'apprentissage de l'état ayant une longueur La $\leq (r+1)(t-1)$ avec $t = card(Q)$.

Lemme A8: Lorsque toutes les lignes d'une matrice de transition d'ordre Ls comportent une même suite d'entrées rangée dans la même colonne j, cette suite d'entrées permet de synchroniser la machine séquentielle à l'état qj. Elle est appelée **"séquence de synchronie de l'automate"**.

Lemme A9: S'il existe une séquence de synchronie dans une machine séquentielle à t états une borne supérieure de sa longueur Ls est : $2^t - (t+1)$.

Identification de l'état de départ:

Lemme A10: Dans une matrice de transitions d'ordre Li, lorsqu'à une suite d'entrées apparaissant dans toutes les lignes, il correspond des suites de sorties différentes deux à deux, cette suite d'entrées permet d'identifier l'état de départ. Elle est appelée **"séquence d'identification de l'état de départ"**.

Lemme A11: Dans toute machine séquentielle fortement connexe et d'ordre r, s'il existe une séquence d'identification de l'état de départ, sa longueur est: $Li \leq (r+1)(t-1)$ avec $t=|Q|$.

BIBLIOGRAPHIE

[1] **M.NARANDJO** «Commande et identification d'automates à partir d'informations incertaines.» thèse Doc. es-sciences, Univ.CLERMONT II, juin 1984.

[2] **J.KELLA** «Sequential machine identification». IEEE Tr. on Comp. pp332-338, march 1971.

[3] **L.P.J.VEELENTURF** «Inference of sequential machines from sample computations». IEEE Tr. on computers, vol.C27, pp167-170, 1978.

[4] **A.W.BIERMAN, J.A.FELDMAN** «On the synthesis of finite state machines from samples of their behavior». IEEE Tr. on comp. pp592-596, 1972.

[5] **A.DUNWORTH, H.V.HARTOG** «An efficient state minimization algorithm for some special classes of incompletely specified machines». IEEE Tr. on comp. vol.C28, No7, pp531-535, 1979.

[6] **F.BANCILHON, M.DEPEYROT** «Identification des automates non fiables». Revue RAIRO juin 1974, pp127-153.

[7] **A.GILL** «Introduction of the theory of finite state machines». Editions Mc Graw Hill, 1962.

[8] **R.GERARDY** «Probabilistic finite state system identification». Int. J. of General systems vol.8, pp229-242, 1982.

[9] **B.R.GAINES** «System identification, approximation and complexity». Int. J. of General Systems vol.3, 1977.

[10] **B.R.GAINES** «Sequential fuzzy system identification». Fuzzy sets and Systems vol.2, No1, pp15-24, 1979.

[11] **B.R.GAINES** «Approximate identification of automata». Electronic Letters vol.11, No18, pp444-445, 1975.

[12] **B.R.GAINES** «Behavior/structure transformations under uncertainty». Int. J. of M.M. studies vol.8, pp337-365, 1976.

[13] **R.GERARDY** «Experiments with some methods for the identification of finite state systems». Int. J. of General Systems, vol.9, pp197-203, 1983.

[14] **J.PEARL** «State complexity of imprecise causal models». IEEE Tr. on SMC vol.6, No9, pp652-655, 1976.

[15] **R.J.NELSON** «Introduction to automata». John Willey & Sons inc 1968.

[16] **J.ZAHD** «Machines séquentielles». Traité d'électricité vol.XI, Editions Georgi, 1980.

[17] **E.M.GOLD** «Complexity of automation identification from given data». Information and Control, No20, pp302-320, 1978.

[18] **M.RICHETIN, M.NARANDJO, G.RIVES, M.BERTHAUD** «Controle assisté par ordinateur d'un train continu à fil». Revue Le Nl Automatisme No37, Avril 1983,pp60-64.

[19] **L.MICLET** «Regular inference with a tail clustering method». IEEE Tr on SMC vol.10, Noll, pp737-743, 1980.

[20] **A.VAN DER MUDE, A.WALKER** «On the inference of stochastic regular grammars». Inform. and control, vol.38, pp310-329, 1978.

[21] **R.M.WHARTON** «Approximate language identification». Information and control, vol.26, pp236-255, 1974.

[22] **R.M.WHARTON** «Grammar enumeration and inference». Information and Control, No33, pp253-272, 1977.

[23] **K.S.FU, T.L.BOOTH** «Grammatical Inference, introduction and survey». IEEE Tr on SMC,
part.1 : SMC.5, Nol, pp95-lll, jan. 1975,
part.2 : SMC.5, No7, pp409-423, jully 1975.

[24] **L.MICLET** «Méthodes structurelles pour la reconnaissance des formes». Editions Eyrolles 1984.

[25] **J.LAGASSE, M.COURVOISIER, J.P.RICHARD** «Logique séquentielle». Dunod Université Paris 1976.

[26] **A.NERODE** «Linear automaton transformations». Proc. of Amer. Math. Soc. vol.9, pp541-544, 1958.

[27] **T.L.BOOTH, R.A.THOMPSON** «Applying probability measures to abstract languages». IEEE Tr on Computers vol.C22, pp442-450, May 1973.

[28] Groupe Systèmes Logiques de l'**AFCET** «GRAFCET, interpréta--tions algèbriques et algorithmiques et temporisations». Le Nl Automatisme sept.1983, pp60-65.

[29] Groupe Systèmes logiques de l'AFCET «GRAFCET, les actions associés aux étapes».Le N1 Automatisme Avril 1985, pp71-74.

[30] Norme NF C03-190 de juin 1982 «Diagramme Fonctionnel «GRAFCET» pour la description des systèmes logiques de commande». Editée par l'UTE.

[31] M.MOALLA, R.DAVID «Extensions du GRAFCET pour la représentation des systèmes temps réels complexes». Revue RAIRO Automatique, vol15, No2, pp159-192, 1981.

[32] J.M.TOULOTTE, J.P. PARSY «A method for decomposing interpreted Petri nets and its utilization». Digital Process No5, pp223-234, 1979.

[33] M.MOALLA, J.SIFAKIS, M.SILVA «A la recherche d'une méthodologie de conception sûre des automatismes logiques basée sur l'utilisation des réseaux de Pétri». «Sécurité et disponibilité des systèmes informatiques». Monographies de l'AFCET, 1980.

[34] J.DEFRENNE, J.M.TOULOTTE, «Sur l'accroissement de la sécurité des systèmes logiques». 4eme journées sci. et techniques de la production automatisée, juin 1983.

[35] A.NAIFI «Contribution à l'étude des systèmes logiques. Fonction combinatoire et théorie de l'information. Sécurité et simulation de la partie opérative». Thèse 3^{e} cycle Automatique USTLille 1983.

[36] C.KUBIAK, L.ETESSE, A.FRIGENT, J.L.RAINARD «La sûreté de fonctionnement dans les systèmes complexes». Note Technique NT/RCI/PLC/3 du CNET Lanion, 1978.

[37] J.DEFRENNE «Implantation de réseaux de Pétri sur automate biprocesseur à haute sûreté de fonctionnement». Thèse de 3^{e} cycle Automatique USTLille, 1979.

[38] S.THELLIEZ, J.M.TOULOTTE «GRAFCET et logique industrielle programmée». Editions EYROLLES 1982.

[39] S.THELLIEZ, J.M.TOULOTTE «Applications industrielles du GRAFCET». Editions EYROLLES 1983.

[40] J.DEFRENNE «Amélioration de la sécurité et de la maintenabilité des machines séquentielles». 4^{e} colloque fiabilité, maintenabilité, Mai 1984.

[41] J.M.TOULOTTE, J.DEFRENNE, R.HACHEMANI «Etude et réalisation d'un automate à sécurité intégrée». Rapport de contrat ADI no83/351, 1984.

[42] M.BLANCHARD «Comprendre, maîtriser et appliquer le GRAFCET». Cepadues Editions 1982.

[43] M.DIAZ «Conception de systèmes totalement autotestables et à pannes non dangereuses».
Thèse Doc. es-sciences, P. Sabatier, Toulouse 1974.

[44] D.DEI-SVALDI, M.COLLIER, J.P.VAUTRIN «Machine commandée par automate programmable, Amélioration de la sécurité». Revue Nl Automatisme, pp44-48, Mai 1983.

[45] J.G.GEFFROY «Test en ligne et sécurité des systèmes logiques». Thèse Doc. d'état es-sciences, Univ. P.Sabatier, Toulouse 1976.

[46] A.PAGES, M.GONDRAN «Fiabilité des systèmes».
Editions Eyrolles Paris 1980.

[47] J.M.TOULOTTE «Réseaux de Petri et automates programmables». Nl Automatisme, no6-7, juillet-août 1978.

[48] A.KAUFMANN «Introduction à la théorie des sous ensembles flous». Editions Masson Paris 1975.

[49] J.DEFRENNE, J.M.TOULOTTE, R.HACHEMANI «Validation des logiciels de commande des systèmes industriels à évolutions simultannées».
Convention Automatique Productique, Paris, Mai 1986.

[50] J.DEFRENNE, J.M.TOULOTTE, R.HACHEMANI «Présentation d'un outil interactif de validation des commandes des systèmes industriels». IMACS Lille 1986.

[51] J.DEFRENNE «Modélisation de la partie opérative, impact sur la sécurité et la maintenance des systèmes à évolution séquentielle».
Thèse d'état Automatique USTLille, 1986.

[52] J.C.LAPRIE «Prévision de la sûreté de fonctionnement et architecture de structures numériques temps réel réparables». Thèse d'état, Univ.P.Sabatier Toulouse 1975.

[53] J.M.COLIN, J.MARTINEZ, M.SILVA «Packages for validating discrete production systems modeled with Petri nets».
IMACS Lille 1986.

[54] E.CASTELAIN, D.CORBEEL, J.C.GENTINA «Simulation de la commande des systèmes de production flexibles».
IMACS symposium, Lille 1986.

[55] **H.DENEUX, R.DAVID** «Spécification et mise en oeuvre d'automates interconnectés à l'aide du grafcet» RAIRO Automatique vol17, no4, pp339-358, 1983.

[56] **R.HACHEMANI** «Modélisation et diagnostic de la partie opérative». Mémoire de DEA, USTLille, juin 1984.

[57] **J.MARIN** «Sur le test en ligne des machines séquentielles réalisées à partir des réseaux de Pétri» Thèse Doct. 3e cycle, Nice 1975.

[58] **J.P.BACONNET, B.GIRARD, S.NATKIN** «Introduction à la sûreté de fonctionnement des systèmes informatiques». Monographie de l'AFCET 1980.

[59] **K.S. FU** «Syntactic methods in pattern recognition» Academic Press, 1974

[60] **C. BERGE** «Graphes et hypergraphes» Editions Dunod, 1970.

[61] **J.G. POSTAIRE, C.P.A. VASSEUR** «An approximate solution to Normal Mixture Identification with application to Unsupervised Pattern Classification». IEEE Tr on PAMI, vol.3, No2, pp163-1nn, Mars 1981.

[62] **P. CASES** «Décomposition d'un histogramme en composantes gaussiennes». Revue de statistique appliquée Vol.XXIV, No1, pp63-82, 1976.

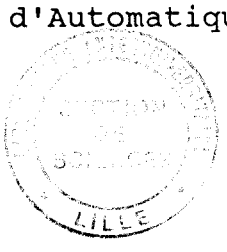
[63] **J.G. POSTAIRE** «Optimisation du processus de classification automatique par analyse de la convexité des fonctions de densité de probabilité» These Doc. es-sciences, Université de Lille, 1981.

[64] **BHATTACHARIA** «A simple method of resolution of a distribution into gaussian components» pp115-135, BIOMETRICS, Mars 1967.

[65] **J.P. BENZECRI** «La regression» (Publication du laboratoire de statistique mathématique), 1970

[66] **T.Y. YOUNG, C. CORALUPPI** «Stochastic estimation of a mixture of normal density functions using an information criterion» IEEE Tr on IT, vol13, No3, pp258-263, 1970

[67] **R. HACHEMANI** «Implantation d'une méthode de test en ligne de la partie opérative sur micro-ordinateur» Rapport interne, Centre d'Automatique de Lille, 1986.



RESUME

La présente étude est une contribution à l'amélioration de la sûreté de fonctionnement, en particulier de la sécurité des systèmes de production automatisés à évolution séquentielle. La détection automatique des défaillances de tels systèmes est nécessaire afin d'assurer la mise en sécurité et/ou la reconfiguration en présence de défaillances.

L'étude des taux de défaillance des automatismes industriels et des techniques généralement utilisées pour améliorer leur sûreté de fonctionnement, en particulier leur sécurité, fait apparaître la nécessité d'une surveillance accrue des éléments de la partie opérative.

Le chapitre 2 étudie la modélisation du comportement de la partie opérative vis à vis de la commande. Cette modélisation est basée sur l'observation de l'évolution d'un certain nombre de grandeurs physiques à travers les capteurs sous l'action des commandes appliquées. Le modèle obtenu est utilisé pour réaliser le test en ligne de la partie opérative. Ce test prend en compte le temps normalement imparti à l'éxécution des actions élémentaires.

Le chapitre 3 présente une solution pour l'implantation du test. Une description par Grafcet est utilisée. Un language de spécification du comportement de la PO permet de générer la structure de données qui sera utilisée en temps réel.

Dans les chapitres 4 et 5 nous étudions les possibilités d'apprentissage automatique du modèle de la partie opérative en vue de générer automatiquement la structure de données. L'analyse des séquences ordres/comptes rendus observées sur la PO permet de retrouver "l'agencement" de ces derniers. La classification des durées des actions élémentaires permet de déterminer les vitesses d'évolution ou commandes appliquées. Les résultats de la classification sont utilisés pour l'identification de la machine séquentielle constituée par la partie "préactionneurs" qui élabore les commandes à partir des ordres appliqués.