N° d'ordre : 894

THÈSE

présentée à

L'UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE TROISIEME CYCLE

Spécialité "Automatique"

par

Jean-Pierre PARSY

"DISPOSITIF INTERACTIF D'AIDE A LA CONCEPTION DES PROGRAMMES DE COMMANDE DE PROCESSUS LOGIQUES INDUSTRIELS".

Soutenue le 1er juin 1981 devant la Commission d'Examen

Membres du Jury	MM.	P. VIDAL	Président
		J.M. TOULOTTE	Rapporteur
		G. MANESSE	Examinateur
		M. BLANCHARD	Invité
		J. LIONET	Invité

Le travail exposé dans ce mémoire a été réalisé au Laboratoire d'Automatique de l'Université de Lille. Avant d'en entreprendre l'exposé je tiens à exprimer ma reconnaissance àtoutes les personnes qui m'ont aidé dans ce travail.

Je remercie le Professeur Vidal Directeur du Laboratoire d'Automatique d'avoir bien voulu accepter la présidence du jury.

Monsieur Toulotte Professeur au Laboratoire d'Automatique a été responsable scientifique et rapporteur de cette thèse; qu'il trouve ici le témoignage de ma gratitude pour ses conseils et l'aide constante qu'il m'a apportée.

Je remercie également Monsieur Manesse Maître Assistant au Laboratoire d'Electrotechnique de Lille, Monsieur Blanchard animateur du groupe "systèmes logiques" de l'A.F.C.E.T et Directeur de la société Valoris, ainsi que Monsieur Lionet de la société Merlin Gérin pour leur participation au jury.

CHAPITRE I LE GRAFCET

CHAPITRE II DESCRIPTION DU SYSTEME D'AIDE A LA REALISATION D'ENSEMBLES LOGIQUES

INTRODUCTION

Depuis plusieurs années, deux grandes révolution sont apparues dans le domaine de la logique industrielle. La première est l'apparition sur le marché de dispositifs programmés performants et bon marché. La seconde est la définition d'un outil de description des systèmes logiques. Ce dernier appelé Grafcet est clair, précis, puissant et contrairement aux méthodes précédentes facile à utiliser. La mise en oeuvre d'un Grafcet n'est cependant pas sans poser de problèmes aux concepteurs peu familiarisés avec les sytèmes programmés. Pour les aider nous leur proposons un système qui permet de vérifier certains points de la validité de la description du cahier des charges donnée sous forme d'un Grafcet et assure l'implantation automatique pour un grand nombre de machines programmées.

Dans le premier chapitre nous rappelons la définition de l'outil Grafcet. Nous donnons ensuite dans le second chapitre la structure du système étudié d'aide à la conception par ordinateur. Le troisième chapitre présente une méthode de décomposition en graphes d'état utile pour certaines méthodes d'implantation. Celles-ci sont enfin présentées dans le cadre du système de traduction.

CHAPITRE I

L E G R A F C E T

Nous présentons dans ce chapitre l'outil de description des systèmes logiques appelé Grafcet défini par le groupe de Travail "systèmes logiques" de l'AFCET, en insistant sur son interprétation point fondamental pour sa réalisation.

Le Grafcet comporte trois parties :

. un graphe
. une situation initiale
. une interprétation

Le Grafcet obéit à des règles d'évolution qu'il importe de bien connaitre pour étudier la méthodologie d'implantation.

I. 1 LE GRAPHE (1) (5) (22)

Le graphe est composé de deux types de noeuds appelés étapes et transitions et de liaisons orientées ou arcs. Un arc relie toujours une étape à une transition ou une transition à une étape.

Le graphe G est donc défini par le quadruplet $G = (E, T, \alpha, \beta)$, où E et T sont deux ensembles finis non vides constitués des étapes et des transitions.

$$E = \{ e_1, e_2, \ldots, e_n \}$$

$$T = \{ t_1, t_2, \ldots, t_m \}$$

Dans la représentation originale (notée dans la suite Go) les étapes sont représentées par des cercles; dans la

.../...

notation normalisée proposée par l'ADEPA (notée dans la suite GN) elles sont représentées par un carré ou un rectangle. Le numéro i de l'étape e_i est notée à l'intérieur du symbole (Fig I 1)

Fig I.1

α et β sont des applications de E x T dans $\{0,1\}$ appelées respectivement fonction d'entrée ou de sortie des transitions définies comme suit :

$\alpha (e_j, t_i) = 1$ si il existe un arc qui relie l'étape e_j à la transition t_i (Fig I.2.- a)

$\beta (e_k, t_l) = 1$ si il existe un arc qui relie la transition t_l à l'étape e_k (Fig I.2.- b)

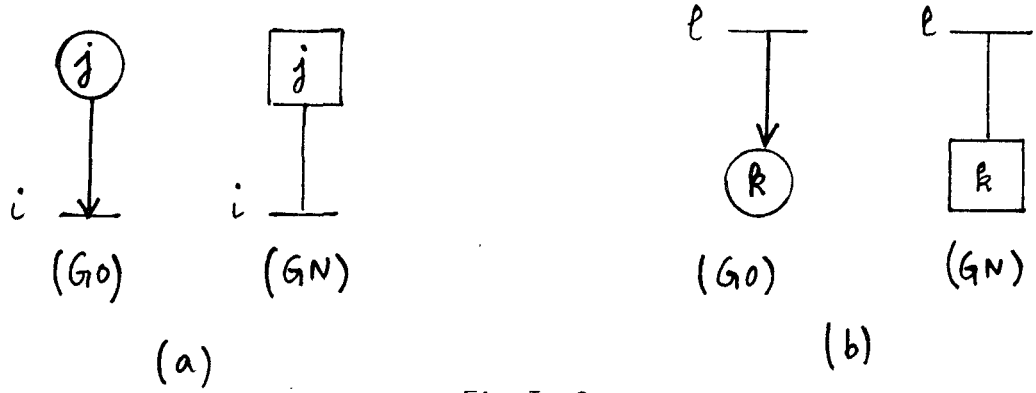


Fig I. 2

L'étape e_j est dite entrée de la transition t_j, e_k est dite sortie de t_l.

.../...

La transition t_l est dite entrée de l'étape e_k, t_i est dite sortie de e_j.

Nous noterons $^\bullet t$ (resp $t^\bullet$) l'ensemble des étapes d'entrée (resp de sortie) de la transition t, et $^\bullet e$ (resp $e^\bullet$) l'ensemble des transitions d'entrée (resp de sortie) de l'étape e

$$^\bullet t = \{ e \in E \mid \alpha(e,t) = 1 \}$$
$$t^\bullet = \{ e \in E \mid \beta(e,t) = 1 \}$$
$$^\bullet e = \{ t \in T \mid \beta(e,t) = 1 \}$$
$$e^\bullet = \{ t \in T \mid \alpha(e,t) = 1 \}$$

I.2 SITUATION INITIALE

Une étape est soit ACTIVE soit INACTIVE.

On appelle situation à un instant donné l'ensemble des étapes actives à cet instant.

Une situation est matérialisée en plaçant un point à l'intérieur du symbole des étapes actives.

Nous associons à chaque étape une variable binaire m_e qui représente son activité :

$$m_e = \begin{cases} 1 & \text{si l'étape est active} \\ 0 & \text{si l'étape est inactive} \end{cases}$$

Ces variables seront appelées dans la suite variables d'étapes.

La situation d'un Grafcet peut être représentée par le sous ensemble de E constiutée des étapes actives à cet instant ou ce qui est équivalent par le n-uple des variables d'étapes $(m_{e_1}, m_{e_2}, \ldots, m_{e_\ell})$

Le concepteur doit obligatoirement préciser la situation initiale S_0, c'est à dire donner la liste des étapes initialement actives. Ces dernières sont repérées en doublant les contours de leur représentation. (Fig. I.3).

Fig. I. 3

L'initialisation qui place le Grafcet dans sa situation initiale S_0, en activant les étapes initialement actives et en désactivant les autres peut être définie implicitement ou explicitement.

Dans le premier cas, elle se fait automatiquement à la mise sous tension. Dans le second cas, le concepteur doit définir une commande d'initialisation I, cette dernière peut se réduire à une simple variable d'entrée ou au contraire être une expression complexe faisant intervenir par exemple des conditions de sécurité et des variables d'étapes en cas de reprise après erreur.

I.3. INTERPRETATION D'UN GRAFCET

Le Grafcet est un outil qui permet de décrire un système logique à deux niveaux :

- le premier précise les spécifications fonctionnelles
- le second doit en plus prendre en compte des contraintes technologiques.

L'interprétation donnée au Grafcet dépend bien sûr du niveau où se place le concepteur. Pour réaliser un système logique, ce dernier doit définir, puis mettre en oeuvre un Grafcet de second niveau; l'établissement du Grafcet de niveau I n'étant qu'une étape intermédiaire, mais indispensable.

Nous allons préciser l'interprétation d'un Grafcet de niveau II.

I.3. 1. RECEPTIVITE

A chaque transition est associée une condition logique appelée réceptivité et notée r_t. Cette dernière fait intervenir l'état ou le changement d'état de variables d'entrée, le caractère actif ou inactif de certaines étapes, le temps matérialisé par une variable fin de temporisation et des prédicats qui permettent de comparer des mots.

Remarquons qu'à ce stade de la description tous les mots et variables sont désignés par un identificateur indépendant du matériel choisi pour la réalisation. Ce dernier est une suite d'au plus 6 caractères alphanumériques commençant par une lettre.

Le passage de 0 à 1 (resp 1 à-0) de la variable d'entree a est noté a↑ (resp a↓). Une temporisation lancée par la variable T de durée a sa sortie FT qui passe dans l'état 1 au bout du temps. τ (Figure I.4). Nous nous limitons ici volontairement à une

.../....

présentation très simple des temporisations, nous les étudierons plus en détail dans un chapitre suivant :

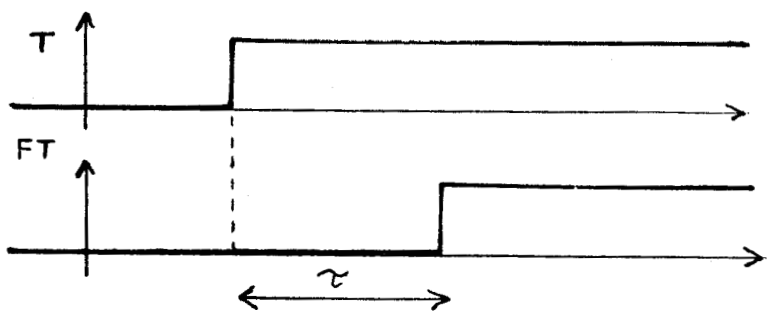


Fig. I.4

Les prédicats permettent de comparer deux mots entre eux, ou un mot et une valeur numérique ; leurs formes sont alors les suivantes :

[id α id]
[id α v]

où id désigne l'identificateur d'un mot, α un opérateur de relation, et v une valeur décimale. Les opérateurs de relations possibles sont ceux du langage Basic, à savoir :

=	pour égal
>	pour strictement plus grand
<	pour strictement plus petit
>=	pour plus grand ou égal
<=	pour inférieur ou égal
<>	pour différent

Il n'est pas admis ici d'effectuer des opérations à l'intérieur des prédicats (par exemple A + B < 5). Toutefois ces dernières peuvent être effectuées auparavant, et il suffit alors de les remplacer par l'identificateur du résultat.

Une réceptivité toujours vérifiée est notée = 1

I.3 - 2 ACTIONS

Une action est toujours associée à une étape, elle peut être :

. une commande externe

. une commande d'un dispositif particulier tel qu'un temporisateur ou un compteur

. une instruction de traitement de mot

. un appel de sous-programme

Les possibilités de traitement de mot et d'appel de sous-programme seront étudiées dans les paragraphes : I.3-2- 7 et I. 3-2-8.

Nous nous limitons pour l'instant aux commandes continues (ou à niveau) externes ou de dispositifs particuliers. Une telle action peut être contrôlée par l'activité d'une ou plusieurs étapes ou bien lancée (resp. arrêtée) dès l'activation d'une ou plusieurs étapes. Elle peut en plus être soumise à une condition logique, qui comme la réceptivité est une expression logique complexe

I. 3 - 2 - 1 ACTIONS CONTINUES INCONDITIONNELLES CONTROLEES PAR L'ACTIVITE D'UNE ETAPE :

A chaque étape e est associé un ensemble A_e d'actions (ou sorties) continues, dites contrôlées sans condition par l'activité de celle-ci.

.../...

Les actions de A_e sont exécutées, c'est-à-dire passent dans l'état 1 lorsque l'étape correspondante e est active, elles cessent d'être exécutées c'est-à-dire reviennent à l'état o lorsque l'étape e cesse d'être active.

Dans la représentation originale ces actions de A_e sont notées à côté du symbole de l'étape correspondante e, dans la représentation normalisée elles doivent être placées à l'intérieur de rectangles reliées à la partie droite du symbole de l'étape e (Fig. I.5)

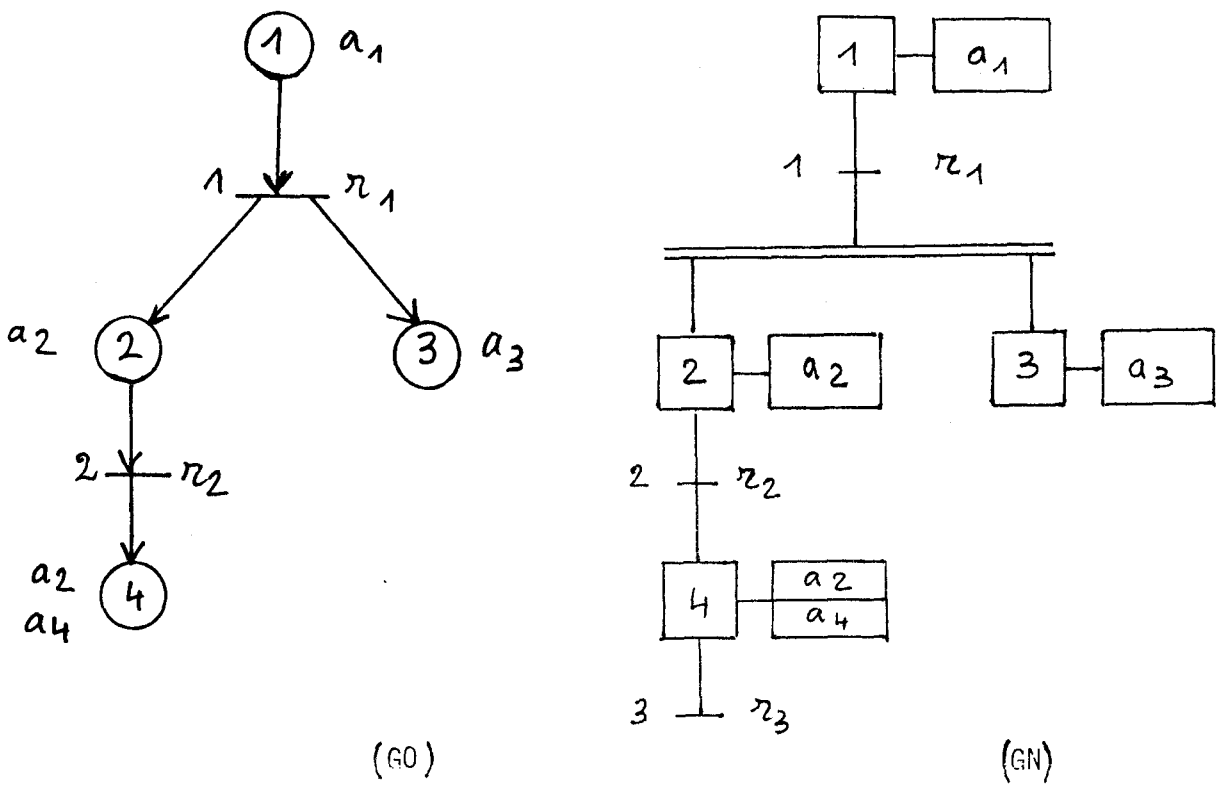


Fig. I.5

$A_{e_1} = \{ a_1 \}$ $\quad A_{e_2} = \{ a_2 \}$

$A_{e_3} = \{ a_3 \}$ $\quad A_{e_4} = \{ a_2 , a_4 \}$

.../...

Le tableau (Fig I.6) précise l'état des actions pour différentes situations

Situation	Etat de l'action			
	a_1	a_2	a_3	a_4
e_1	1	0	0	0
e_2 e_3	0	1	1	0
e_3 e_4	0	1	1	1

Fig. I. 6

Dans une situation donnée, l'état de l'action a est la somme logique des variables m_e associées aux étapes, dont l'ensemble A_e contient a

$$a = \sum_{\{e \in E \mid a\ Ae\}} m_e$$

Ainsi pour l'exemple de la Fig I.5 nous avons

$$a_1 = m_{e_1}$$
$$a_2 = m_{e_2} + m_{e_4}$$
$$a = m_{e_3}$$
$$a = m_{e_4}$$

.../...

I. 3 - 2 - 2 ACTIONS CONTINUES CONDITONNELLES CONTROLEES PAR L'ACTIVITE D'UNE ETAPE

Une action a contrôlée par l'activité de l'étape e peut être conditionnelle, c'est à dire soumise à une condition logique. Cette condition peut être une expression complexe portant entre autre sur des variables d'entrée, des variables d'étapes et des prédicats numériques.

Dans ce cas l'action a est effective si à la fois l'étape est active et la condition est vérifiée.

L'action a soumise à la condition c est notée :

si c : a

ou en omettant le si :

c : a

Notons c(a,e) la condition à laquelle est soumise l'action a associée à l'étape e.

Remarquons qu'une action inconditionnelle n'est qu'un cas particulier d'action conditionnelle :

La condition qui dans ce cas est toujours vérifiée est égale à 1

A_e désignera dans la suite de l'exposé les actions contrôlées par l'étape e conditionnelles ou non

Ainsi pour l'exemple de la figure 7 nous avons :

$$A_{e_1} = \{a_1\} \; , \; A_{e_2} = \{a_2 \; , \; a_4\} , A_{e_3} = \{a_4\}$$

.../...

$$c(a_1, e_1) = 1$$
$$c(a_2, e_2) = c_1$$
$$c(a_4, e_2) = c_2$$
$$c(a_4, e_3) = c_1$$

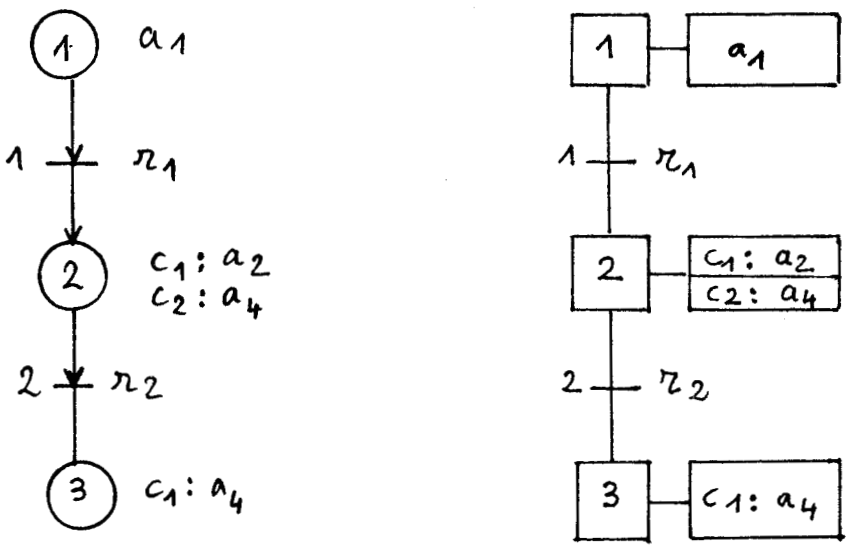


Fig I.7

(GO) (GN)

L'expression donnant l'état de l'action a est alors égale à la somme logique des produits des variables associées m_e associées aux étapes e, dont l'ensemble A_e contient a, par la condition c(a,e).

$$a = \sum_{\{e \in E \mid a \in A_e\}} c(a,e)\, m_e$$

Pour l'exemple (Fig I.7) nous avons

$$a_1 = m_{e_1}$$
$$a_2 = c_1 \cdot m_{e_2}$$
$$a_4 = c_2 \cdot m_{e_2} + c_1\, m_{e_3}$$

Notons c(a,e) la condition à laquelle est soumise l'action a associée à l'étape e .

.../....

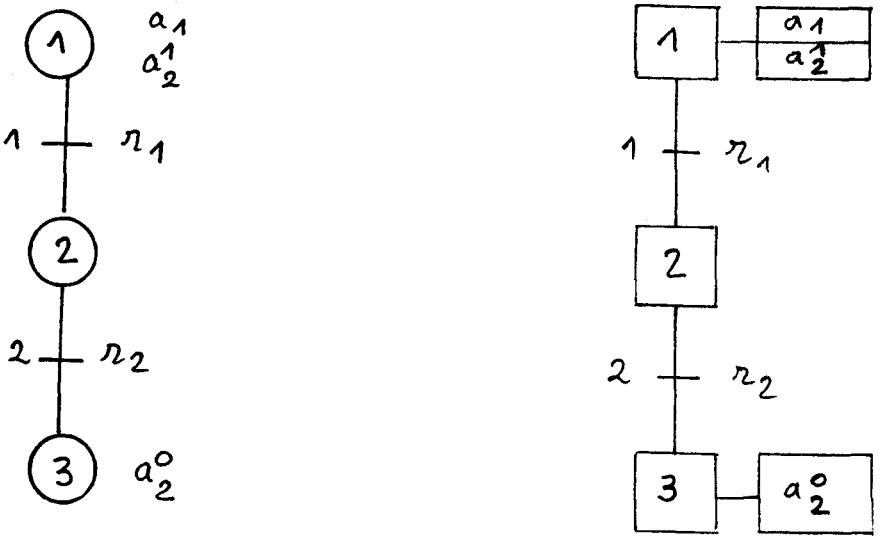
I. 3 - 2 - 3 ACTIONS CONTINUES INCONDITIONNELLES LANCEES OU ARRETEES DES L'ACTIVATION D'UNE ETAPE :

A chaque étape e nous associons également deux ensembles notés A_e^1 et A_e^0

Le premier est constitué des actions à lancer, c'est à dire à mettre à l'état 1 dès que l'étape e est activée, le second des actions à arrêter c'est à dire à mettre à l'état 0 également dès que l'étape e est activée.

Les actions de A_e^1 et A_e^0 contrairement à celles de A_e , une fois démarrées ou stoppées par l'activation de l'étape e, ne sont plus contrôlées par l'activité de cette dernière.

Pour distinguer les différentes actions associées à une étape, nous indicerons par 1 ou 0 les actions de A_e^1 et A_e^0 (Fig. I.8)



(GO) Fig I.8 (GN)

.../...

Sur cet exemple, l'action a démarrée dès l'activation de e (Fig I.9)

Situation	Etat de l'action	
	a_1	a_2
e_1	1	1
e_2	0	1
e_3	0	0

(Fig I. 9)

Le Grafcet précédent (Fig I.8) est alors équivalent à celui donné par la figure I.10.

Le concepteur peut ainsi éviter de répéter une action continue contrôlée par une suite d'étapes conséquentes, ainsi les Grafcets des figures 10 et 11 sont équivalents.

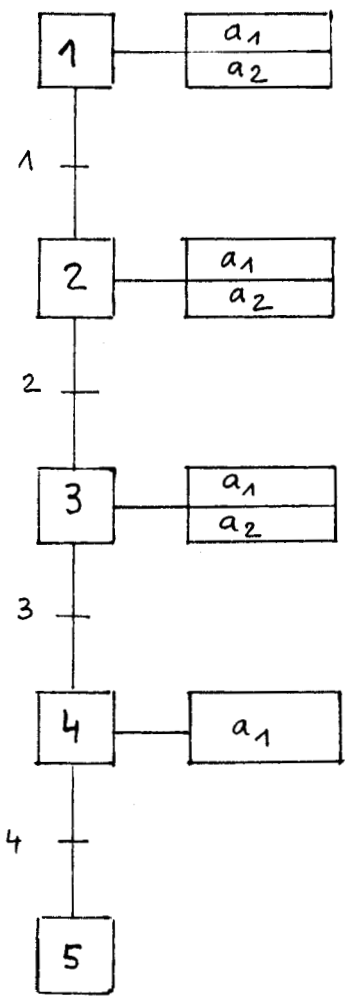


Fig. I.10

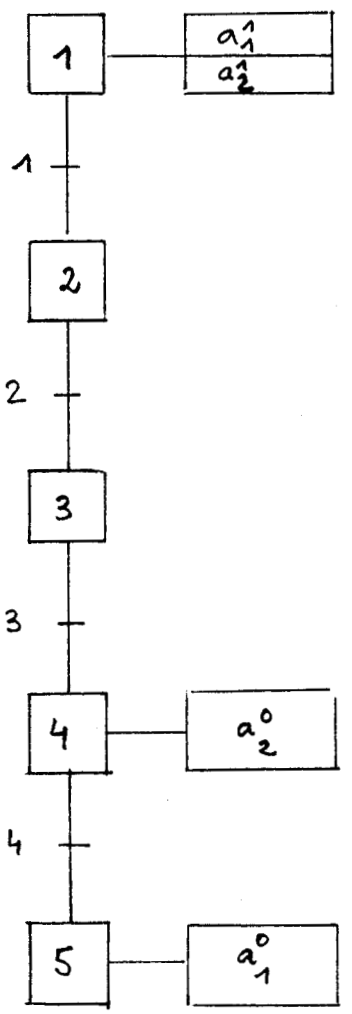
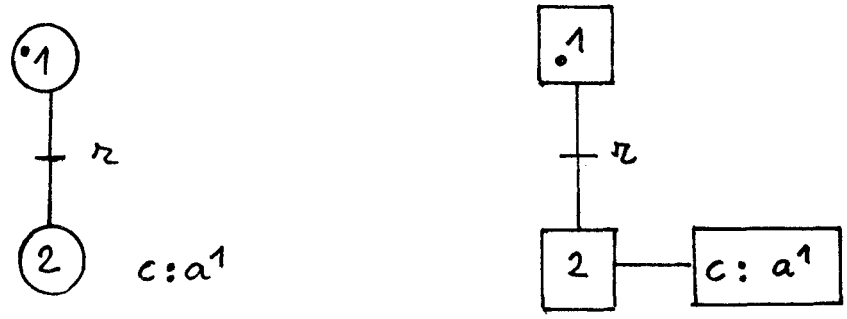


Fig. I.11

I. 3 - 2 - 4 ACTIONS CONDITIONNELLES LANCEES OU ARRETEES DES L'ACTIVATION D'UNE ETAPE :

Le lancement (resp l'arrêt) d'une action continue a^1 (resp a^0) par l'activation d'une étape peut être conditionné par une expression notée $c(a^1, e)$ (resp $c(a^0, e)$) ce dernier n'est alors effectif que si la condition est vraie à l'activation de l'étape.

Ainsi sur l'exemple de la Figure I.12 dès que la réceptivité r est vérifiée, la sortie a est lancée si la condition c est vraie, sinon elle n'est pas modifiée



(GO) Fig I.12 (GN)

I. 3 - 2 - 5 ACTIONS IMPULSIONNELLES (OU PONCTUELLES) INCONDITIONNELLES :

Certains éléments extérieurs à commander sont de nature bistable : Vérin à double effet, relais à accrochage....

Il peut-être inutile, dans ce cas, de maintenir à 1 la commande de tels dispositifs.

Nous associons à chaque étape un ensemble A^*_e d'actions impulsionnelles ; celles-ci sont notées à l'aide d'un astérisque et suivies éventuellement d'une durée, elles sont exécutées c'est à dire passent dans l'état 1, dès l'activation de l'étape e ; elles ne restent

..../...

dans l'état 1 que pendant la durée précisée, ou à défaut pendant un temps élémentaire τ égal, en général au cycle machine.

I. 3 - 2 - 6 ACTIONS IMPULSIONNELLES (OU PONCTUELLES) CONDITIONNELLES :

Une action impulsionnelle a* de durée d associée à l'étape e peut être en plus soumise à une condition c(a*,e) (Fig.13), dans ce cas elle passe à 1 pendant un temps d à l'activation de l'étape et chaque fois que la condition logique associée devient vraie. La figure 14 précise l'état de la sortie impulsionnelle a* soumise à la condition c.

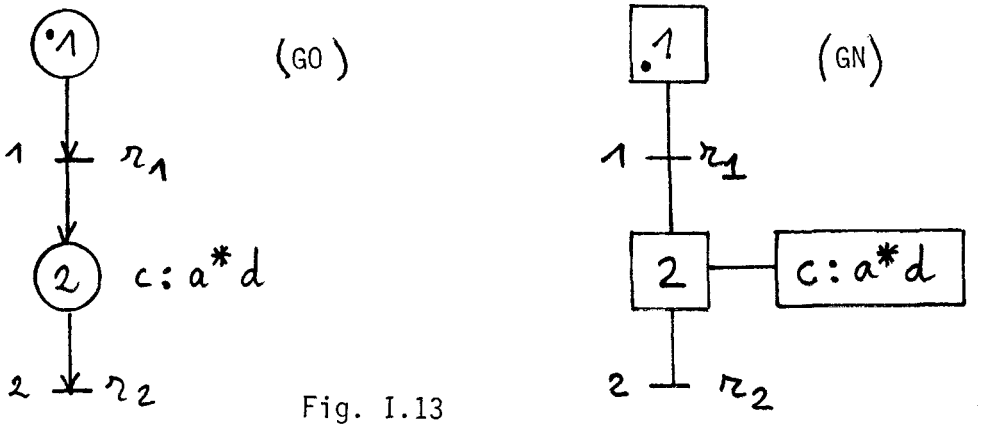


Fig. I.13

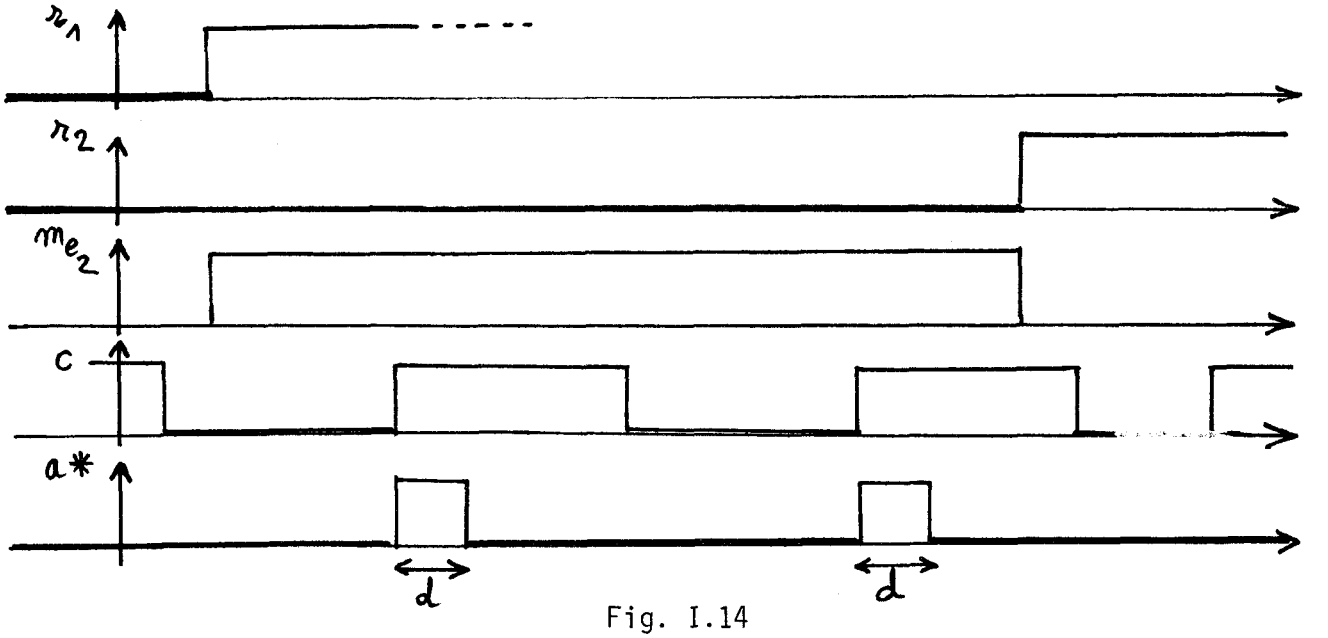


Fig. I.14

I. 3- 2 -7 ACTIONS DE TRAITEMENT SUR MOTS :

Les actions de traitement sur mots sont des deux types suivants : transfert et calcul. Une instruction de transfert permet de charger un mot (interne ou externe) par le contenu d'un autre mot ou une constante, ses formes d'écriture sont respectivement :

$$[id_1 \leftarrow id_2]$$

$$[id_1 \leftarrow v]$$

Une action de calcul est d'un des deux types :

$$[id_1 \leftarrow id_1 \quad op \quad id_2]$$

$$[id_1 \leftarrow id_1 \quad op \quad v]$$

Donnons à titre d'exemples :

. le transfert (interne ou externe) du mot A dans le mot B

$$[B \leftarrow A]$$

. la remise à 0 du mot A

$$[A \leftarrow 0]$$

. l'addition du mot B au mot A

$$[A \leftarrow A + B]$$

. la soustraction de 5 au mot B

$$[B \leftarrow B - 5]$$

.../...

I. 3. 2. 8 NOTION DE SOUS - PROGRAMME

La notion de sous-programme permet d'éviter la répétition de plusieurs sous Grafcets identiques dans la représentation. Ainsi le Grafcet (Fig I.15) qui présente deux sous Grafcets identiques constitués chacun des étapes $\{1,2,3\}$ et $\{4,5,6\}$ peut se réduire à celui de la figure I.16.

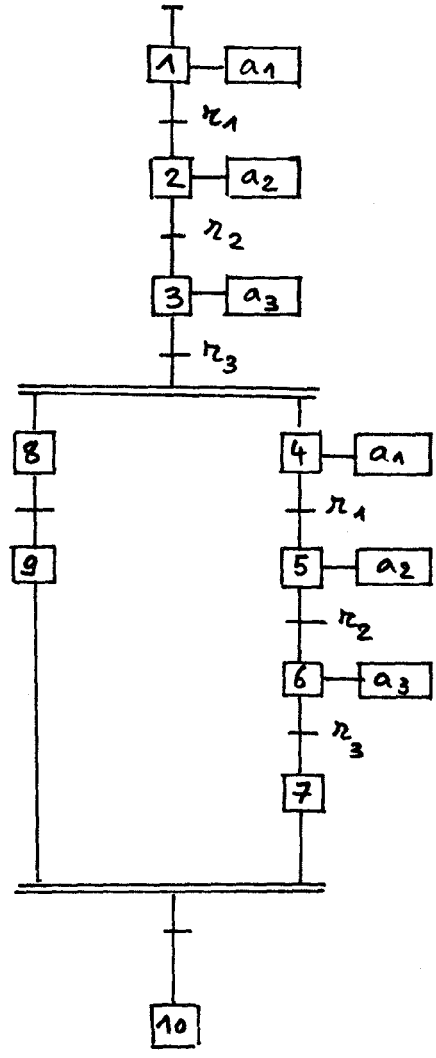
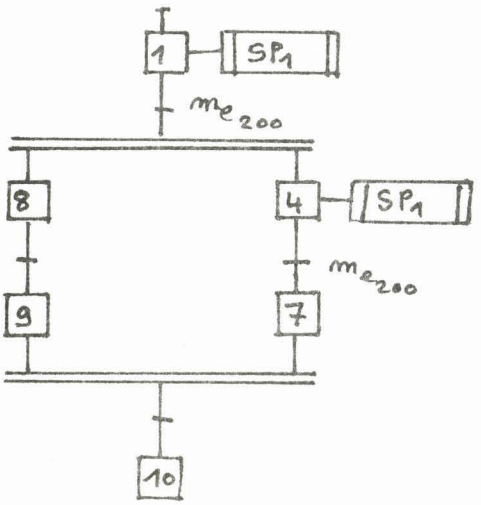


Fig. I.15



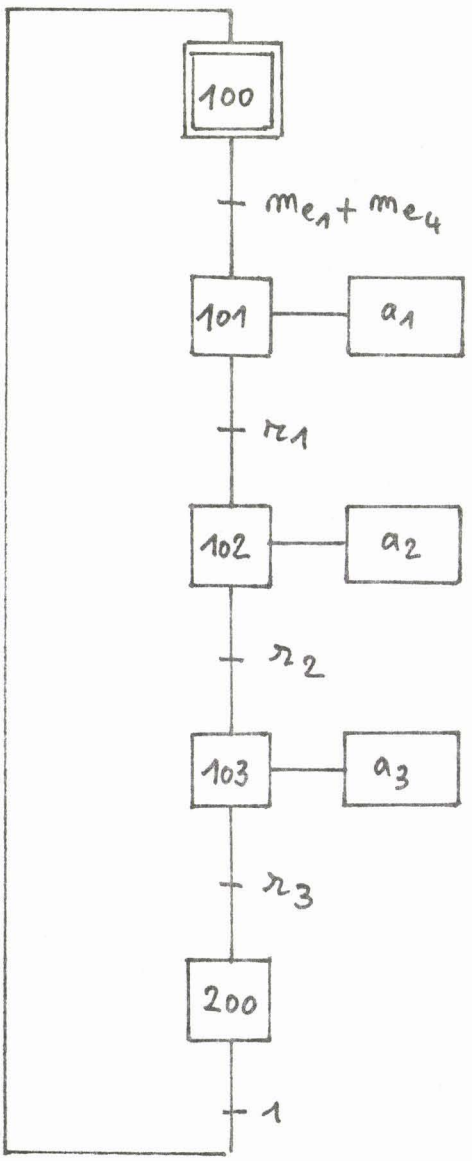


Fig. I.16

Dans un sous programme paramètré le concepteur peut préciser certaines réceptivités et actions sous forme d'une liste de paramètres, ainsi le Grafcet (Fig I.17) conduit à la représentation donnée par la figure I.18.

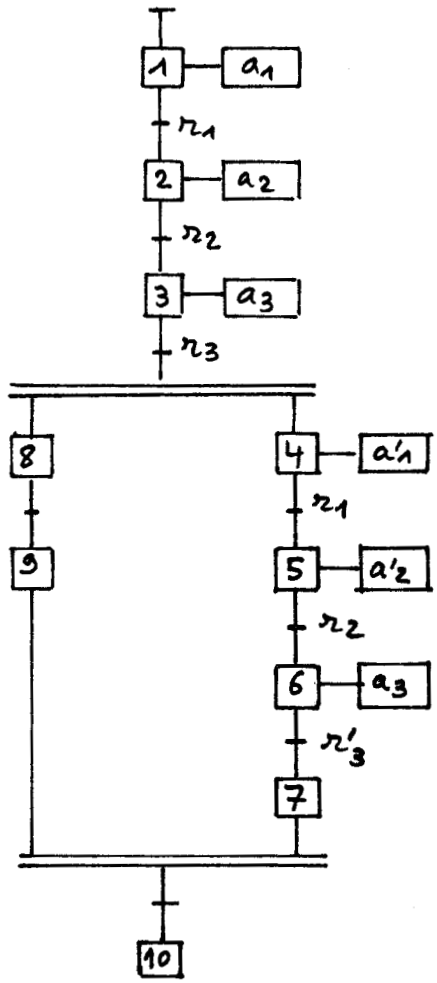


Fig. I.17

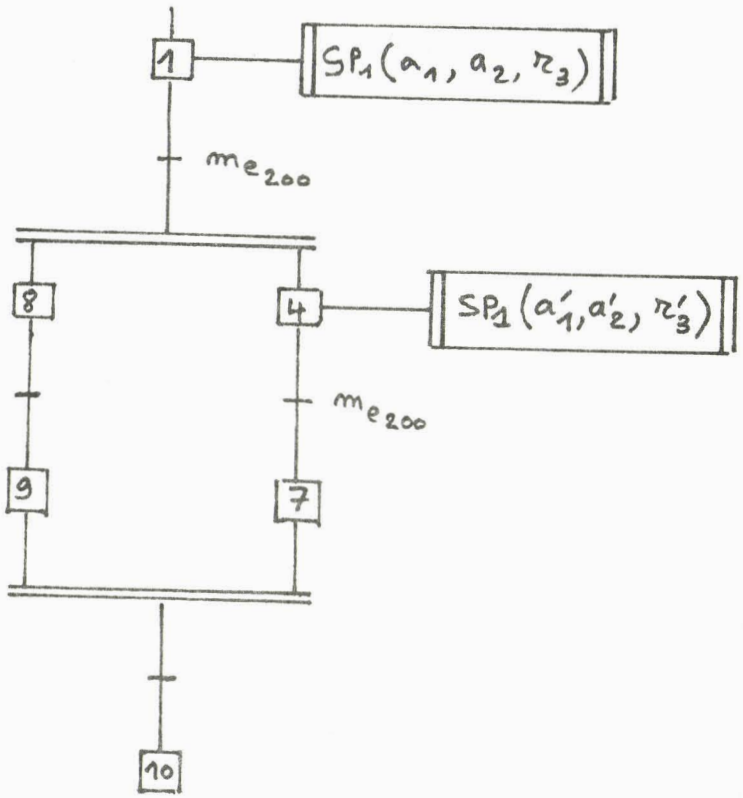


Fig. I.18

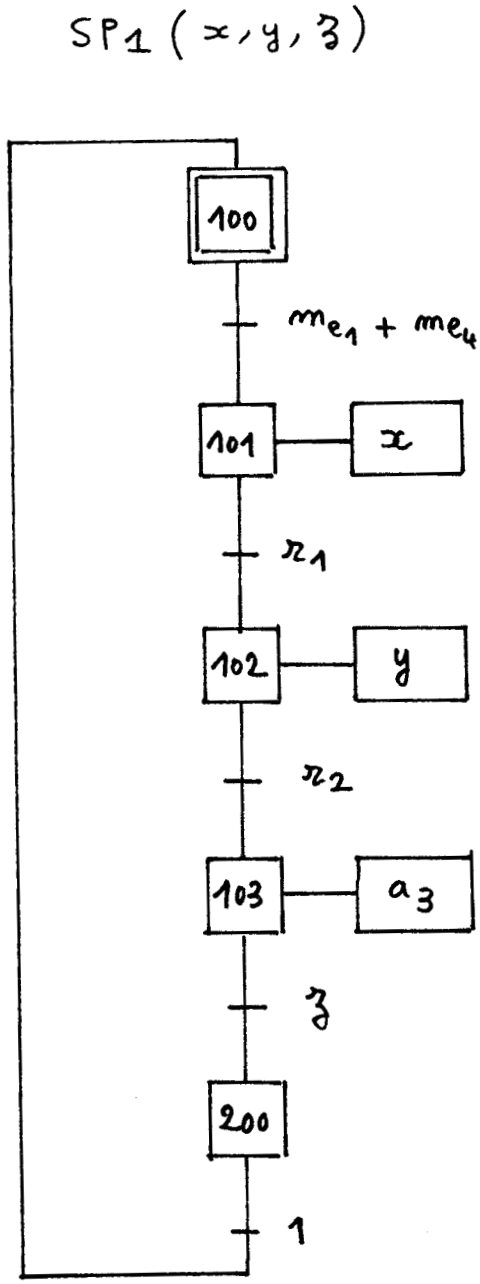


Fig. I.18 suite

Le sous programme paramètré SP_1 (x,y,z) peut s'écrire sous forme multiplexée, l'action x est remplacée par les deux actions conditionnelles:

$$m_{e_1} : a_1$$

$$m_{e_4} : a'_1$$

l'action y par:

$$m_{e_1} : a_2$$

$$m_{e_4} : a'_2$$

et la réceptivité z par :

$$m_{e_2} \cdot r_3 + m_{e_4} \cdot r'_3$$

(Fig I.19)

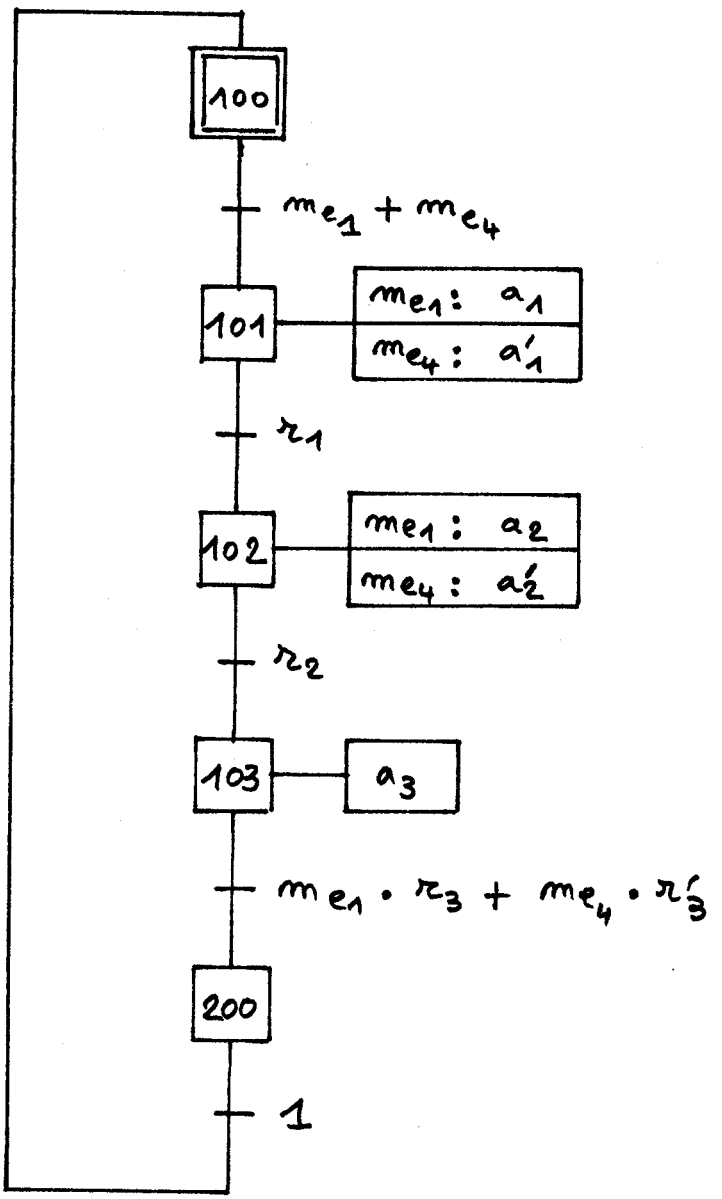


Fig. I.19

En utilisant la notion de sous programme paramètré, le concepteur peut définir des instructions de traitement sur mot sur des machines qui ne possèdent pas ces possibilités. L'appel du sous programme conduit à l'exécution d'une suite d'instructions élémentaires réalisant le traitement souhaité.

I.4 EVOLUTION D'UN GRAFCET

L'évolution d'un Grafcet obéit aux règles suivantes

R 1 Règle de validation et possibilité de franchissement d'une transition.

Une transition t est dite VALIDEE lorsque toutes ses étapes d'entrèe sont actives.

Une transition sans étape d'entrée est toujours validée.

Une transition peut être franchie (ou franchissable) lorsqu'elle est validée et que sa réceptivité r_t est vraie.

La condition de validation CV_t de la transition t est égale au produit logique des variables de ses étapes d'entrée

$$CV_t = \prod_{e \in {}^\bullet t} m_e$$

La condition de franchissement CF_t de la transition t est égale au produit logique de sa condition de validation et de sa réceptivité r_t

$$CF_t = r_t \cdot CV_t$$

$$= r_t \prod_{e \in {}^\bullet t} m_e$$

.../...

R2 Règle de franchissement d'une transition

Le franchissement d'une transition conduit simultanément à la désactivation de toutes ses étapes d'entrée et à l'activation de toutes ses étapes de sortie.

Le Grafcet étant dans la situation :

$$s = (m_{e_1} , \ldots , m_{e_n})$$

le franchissement de la transition t conduit à la situation :

$$s' = (m'_{e_1} , \ldots , m'_{e_n})$$

avec

$$m'_{e_i} = m_{e_i} - \alpha(e_i, t) + \beta(e_i, t) \quad \text{pour } i=1,2, \ldots ,n$$

Une évolution de situation peut se produire quand la condition d'évolution

$$CE = \sum_{t \in T} CF_t$$

est vérifiée.

le Grafcet possède deux autres règles d'évolution dites règles complémentaires :

R3 Plusieurs transitions simultanément franchissables sont simultanément franchies.

R4 Si une étape doit être désactivée puis activée simultanément , elle reste active.

Ces deux règles sont très importantes , mais d'un usage dangereux pour un utilisateur non averti et exigent comme nous le verrons des méthodes d'implantation plus complexes.

Nous distinguerons les Grafcet élémentaires appelés Grafcet de type I pour lesquelles ces règles complémentaires ne sont pas appliquées et les Grafcet généraux dits de type II qui utilisent ces règles.

Après avoir défini le modèle de description des systèmes logiques , nous décrivons dans le chapitre suivant l'outil de conception assistée proposé.

CHAPITRE II

DESCRIPTION DU SYSTEME D'AIDE A LA REALISATION D'ENSEMBLES LOGIQUES

Après avoir défini l'outil de description des systèmes logiques, nous présentons l'outil d'aide à la réalisation de ces derniers.

II. 1 GENERALITES : (6) (14) (15)

L'implantation du système d'aide à la réalisation d'ensembles logiques a été réalisée sur un microordinateur de taille mémoire 32 K octets et muni des périphériques double unité de disquettes et imprimante. Pour des raisons essentiellement de coût, notre choix s'est porté sur le micro-ordinateur PET CBM 3032 de Commodore (Fig II.1)

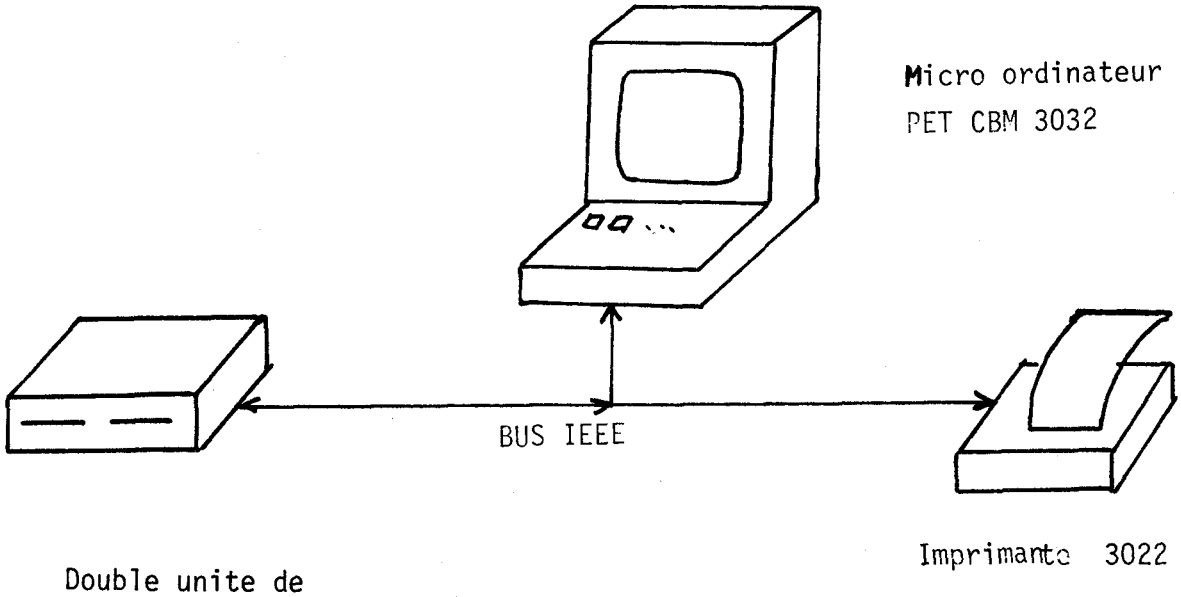


Fig II.1

Structure matérielle du système

.../...

Le langage retenu a été le langage Basic, qui était la seule possibilité à cette date sur cette machine.

Le système doit permettre au concepteur d'entrer le Grafcet de son application, de le stocker sur disquette et éventuellement de le modifier par la suite. Il doit être capable d'effectuer une simulation d'un Grafcet ; le concepteur a ainsi la possibilité de vérifier la concordance entre le cahier des charges et sa description. Il doit donner la possibilité d'assurer la correspondance entre les identificateurs choisis par le concepteur et les emplacements physiques internes et du bornier. Enfin, il doit assurer l'implantation automatique du Grafcet sur le matériel retenu, suivant éventuellement plusieurs méthodes.

Examinons les différents modules du système et leurs liaisons.

II.2. STRUCTURE GENERALE DU SYSTEME

L'entrée et la modification éventuelle d'un Grafcet est réalisé par le module EDITEUR GRAFCET (Fig II.2) Rappelons, qu'à ce niveau, toutes les variables (d'entrée, de sortie ou internes) sont désignées par un identificateur indépendant du matériel retenu.

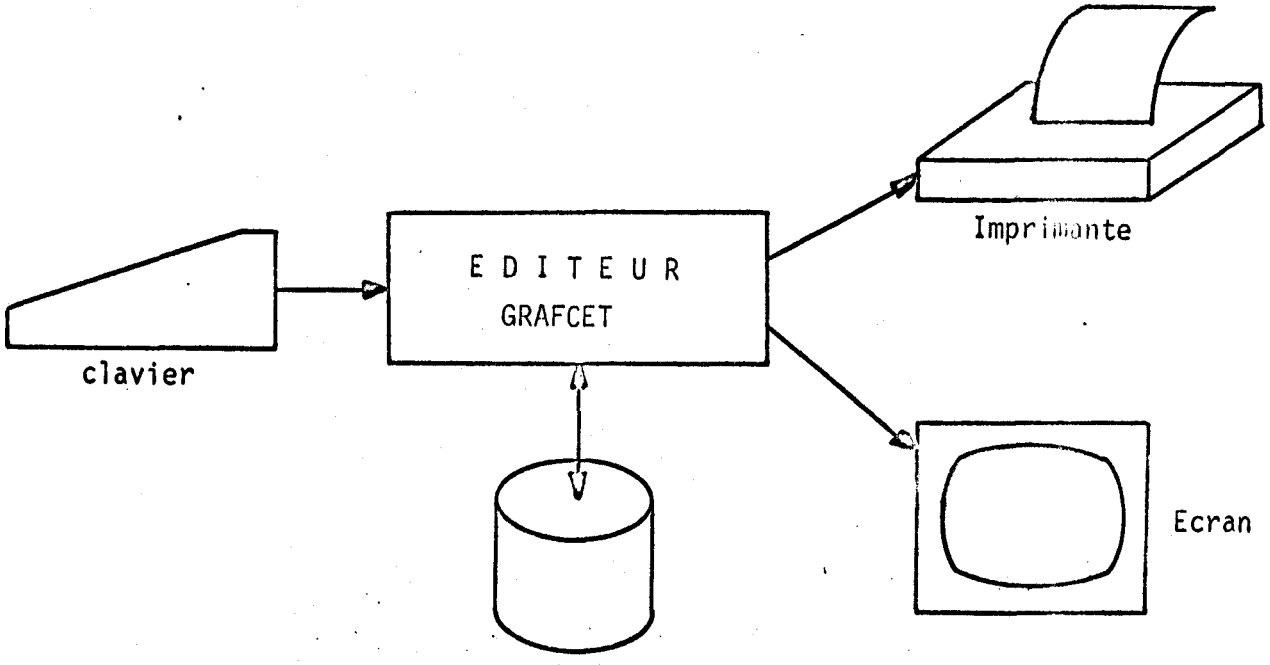


Fig II.2. Edition du Grafcet

Le concepteur peut simuler le système décrit par un Grafcet (Fig II.3).Pour cela il doit définir la situation de départ, puis imposer l'état des entrées et des fins de temporisations intervenant dans les réceptivités des transitions validées dans cette situation.

Le système détermine alors automatiquement la nouvelle situation, l'état des sorties et les transitions en conflit. L'utilisateur peut alors demander une nouvelle évolution en fixant le nouvel état des entrées, ou bien définir une nouvelle situation de départ.

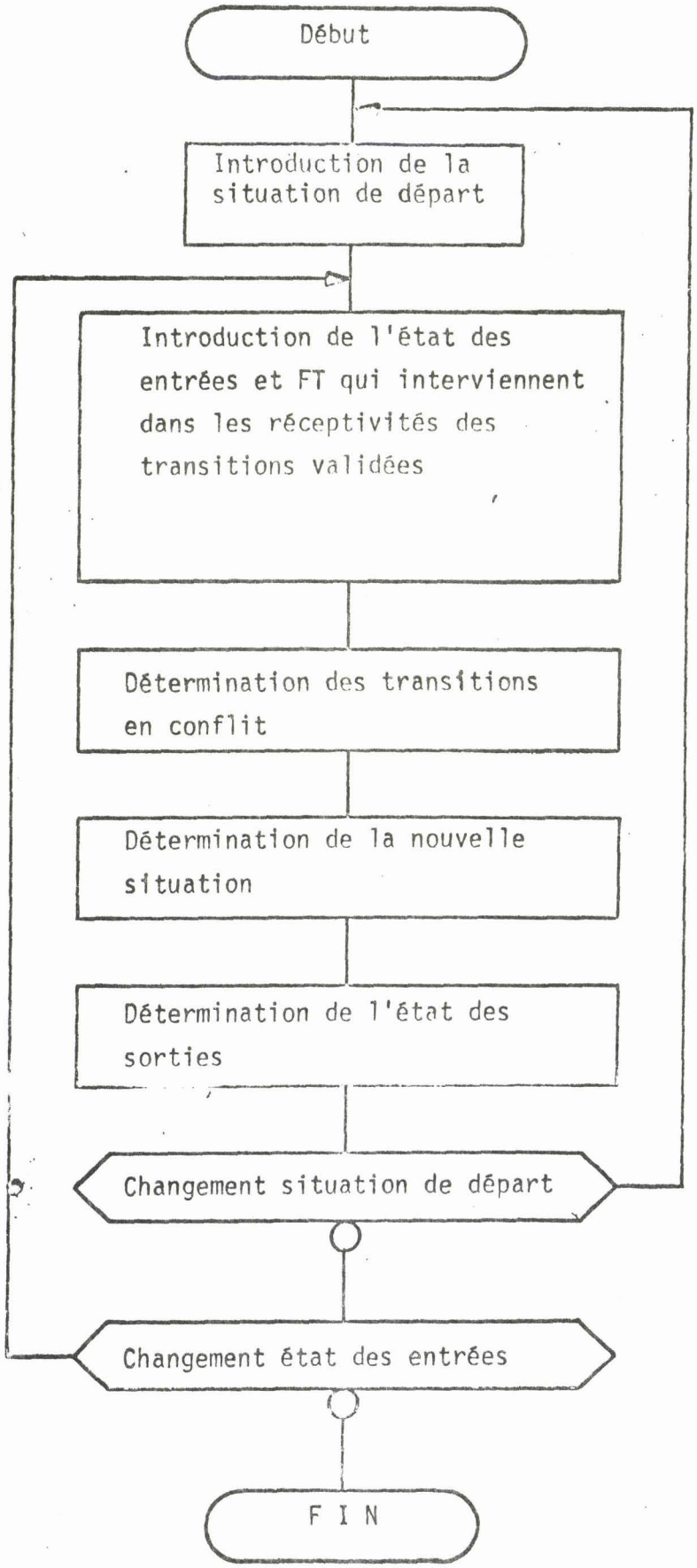


Fig II.3 Principe de la simulation

Si l'évolution obtenue ne correspond pas à celle souhaitée, le concepteur doit modifier son Grafcet. Dans un souci de vérification, il est souhaitable qu'il effectue à nouveau la même simulation ; il est ainsi obligé à chaque modification de redéfinir pour un gros problème, l'état d'un nombre important d'entrées. Un module EDITEUR PHASE DE SIMULATION (Fig II.4) permet de stocker sur disquette pour une situation donnée tous les états successifs des entrées ; Ces phases de simulation peuvent alors être utilisées à tout moment par le module SIMULATEUR (Fig II.5)

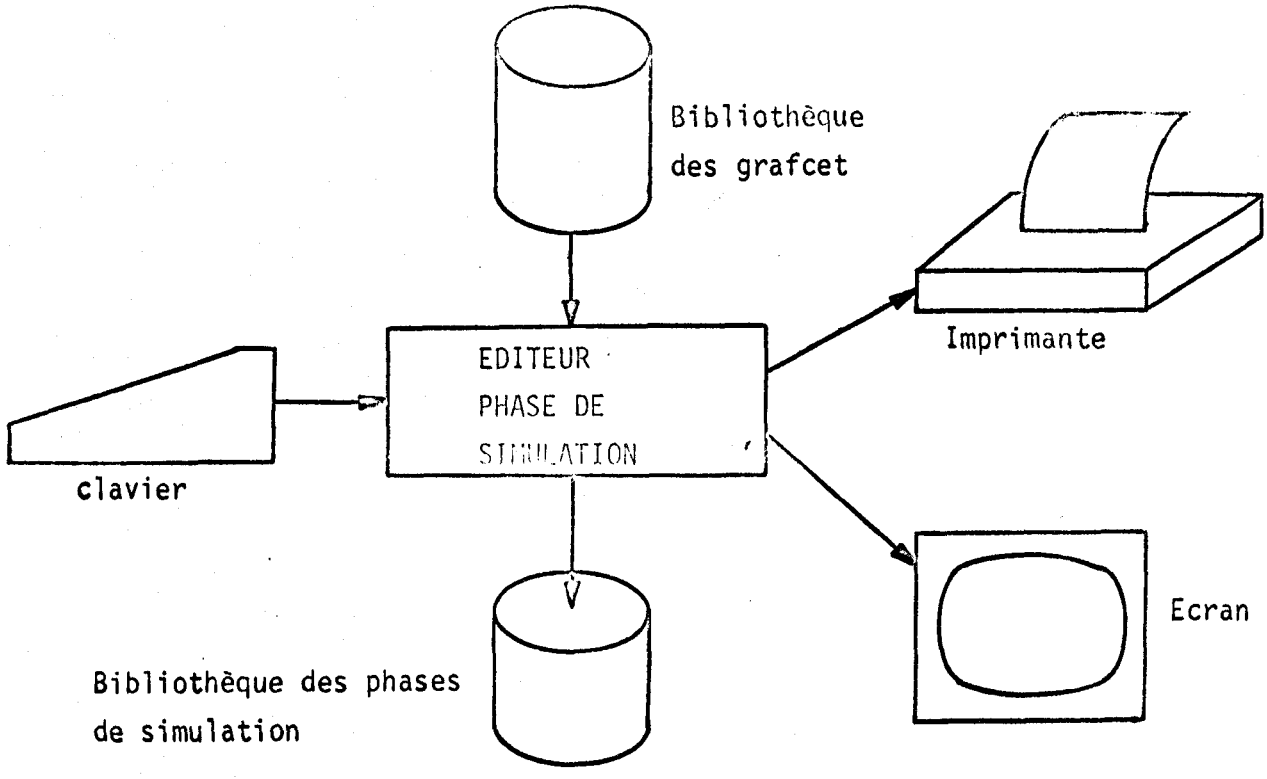


Fig II.4 Edition d'une phase de simulation

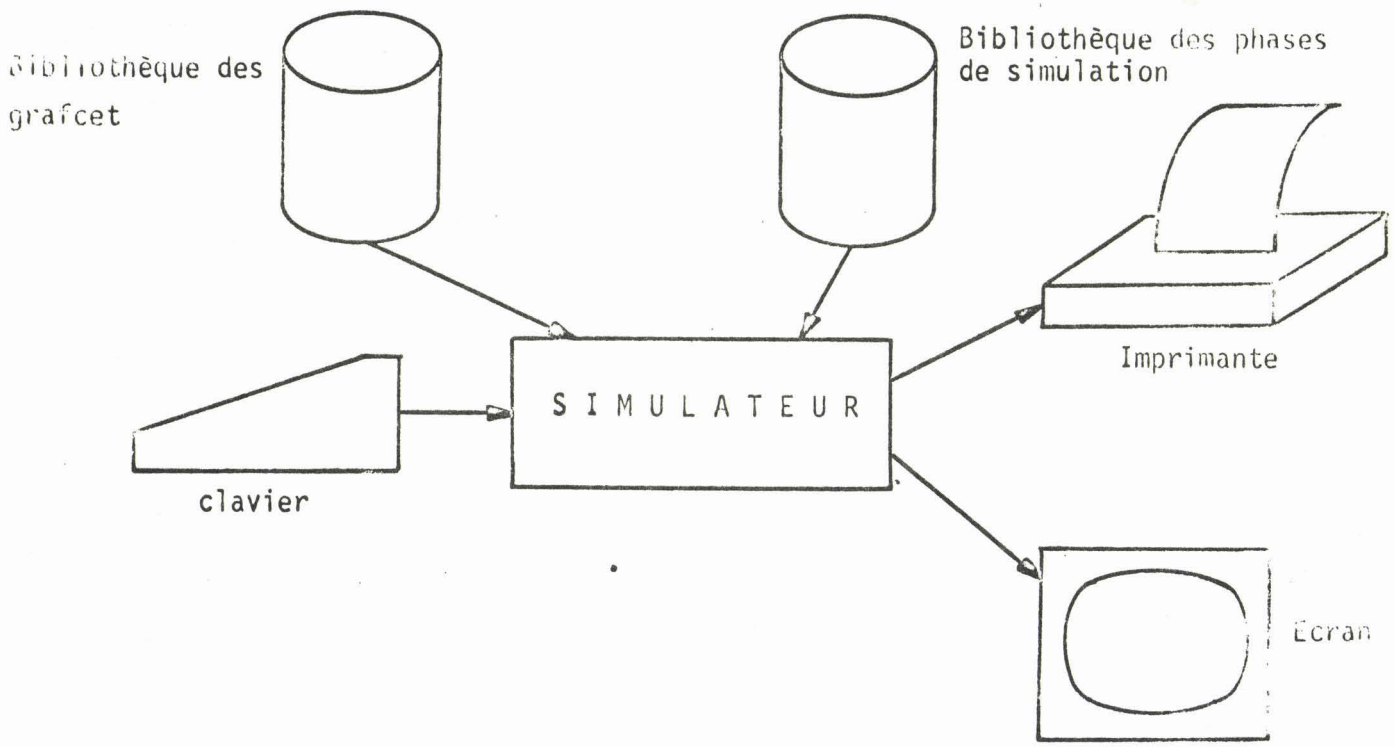


Fig II.5 Simulation

Le module EDITEUR TABLES permet au concepteur d'entrer et éventuellement de modifier les tables de correspondance entre ses identificateurs et les repères physiques. Celles-ci peuvent être entrées indépendamment du Grafcet, ou au contraire à partir de ce dernier (Fig II.6). Dans le premier cas, (dit mode manuel) le concepteur doit introduire l'identificateur et le repère correspondant ; dans le second (dit mode automatique) le système affiche chaque identificateur et il n'y a plus qu'à introduire le repère associé.

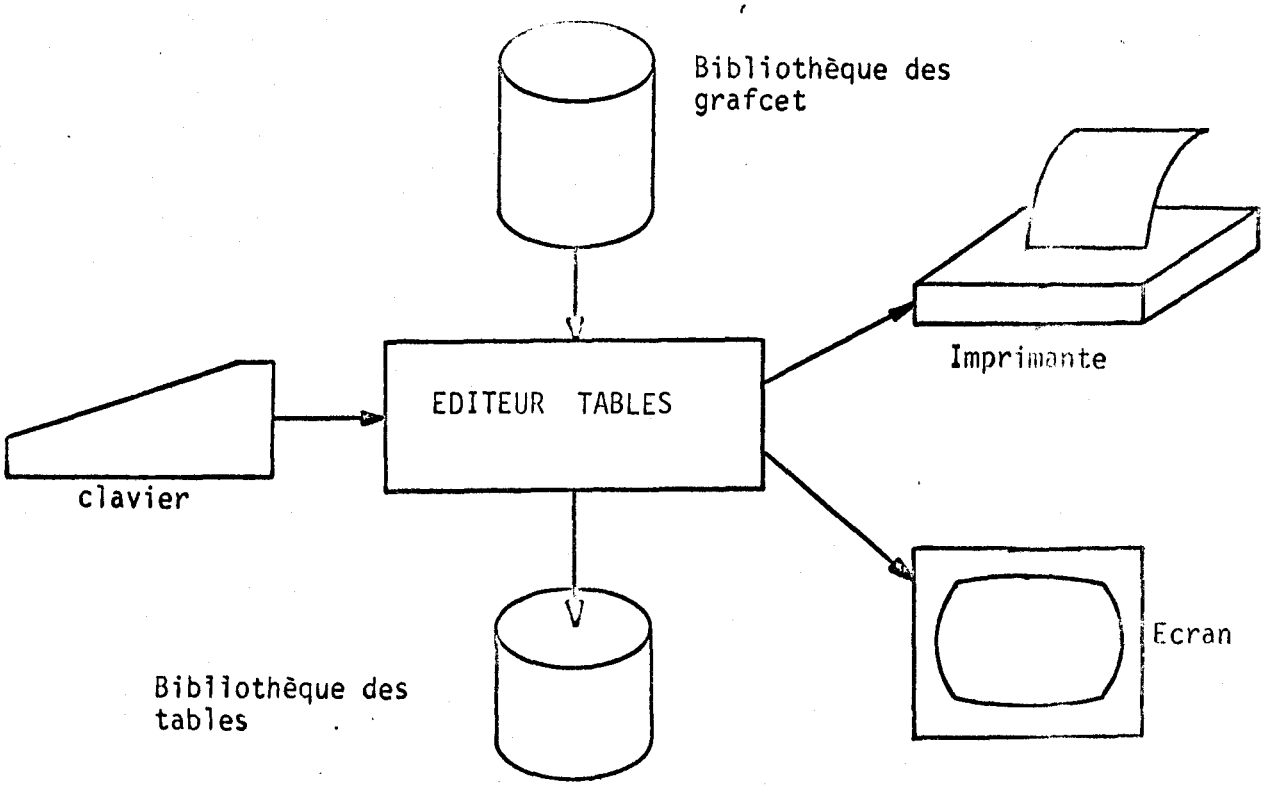


Fig II.6. Edition des tables en modes manuel ou automatique

Le système peut alors assurer l'implantation automatique du Grafcet sur le matériel choisi, suivant la méthode retenue. Pour éviter d'avoir un traducteur complet par machine , un module TRADUCTEUR GENERAL effectue une prétraduction dans un langage intermédiaire indépendant du matériel. L'un des traducteurs spécialisés par famille de machines assure ensuite la traduction finale compte-tenu des spécificités du matériel retenu. Remarquons que le programme est fourni à la fois sous forme symbolique et binaire ; sous cette dernière forme, il peut donner lieu à la programmation d'une mémoire morte, ou à un transfert dans la mémoire vive de la machine. Le système peut également effectuer une décomposition, nécessaire pour certaines méthodes d'implantation. (Fig. II.7)

Un module GRAPHE DES SITUATIONS ACCESSIBLES fournit la table des situations accessibles compte-tenu de l'interprétation des transitions solidaires et concurrentes.

Un module VERIFICATION s'assure de la conformité de structure du Grafcet.

L'enchainement entre ces différents modules est assuré par un moniteur appelé CAO GENERAL. A la mise en route du système, le menu offert, c'est à dire les différentes commandes possibles, est alors:

[E]DITION
[S]IMULATION
[V]ERIFICATION
[T]RADUCTION
[G]RAPHE DES ACTIVITES

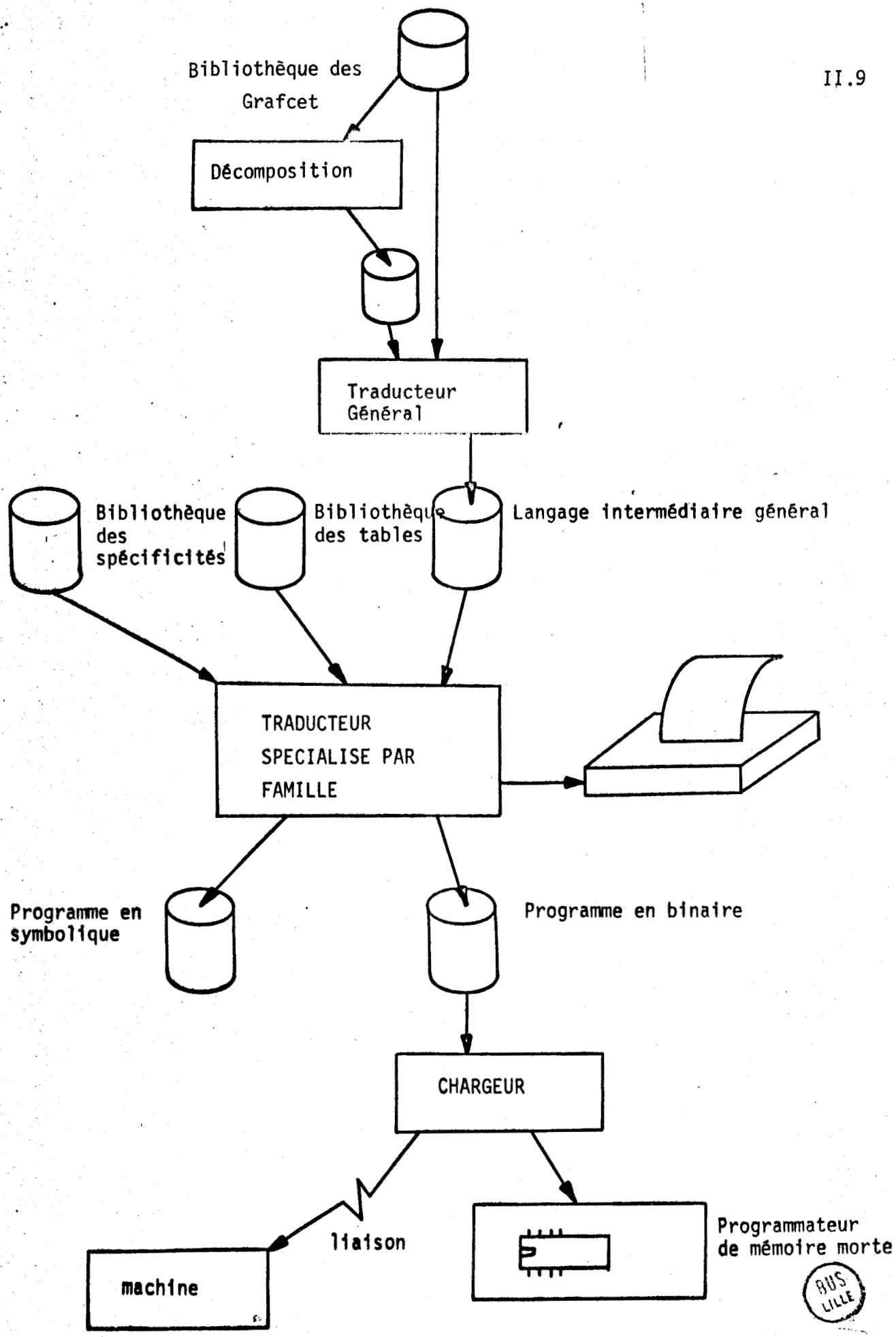


Fig II.7 Implantation automatique du Grafcet

Pour obtenir une commande ,il suffit de taper au clavier sa première lettre ; tout appui sur une lettre autre que **E** , S , V , T et G est sans effet (Fig II.8)

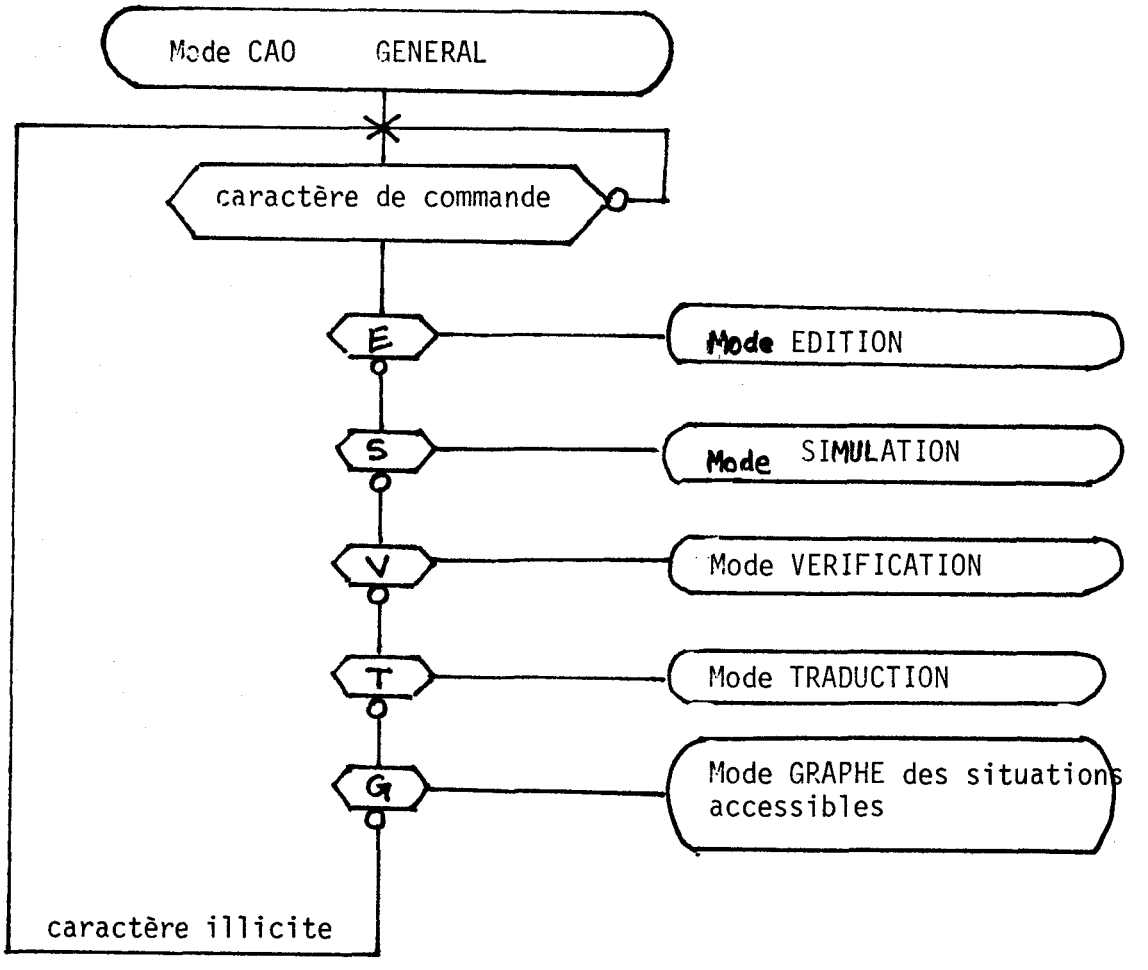


Fig II.8

Menu du Mode CAO GENERAL

Dès l'appui sur la touche E, le système passe en mode EDITION (Fig II.9), le menu offert est alors :

[G] RAFCET
[P] HASE DE SIMULATION
[T] ABLES
[C] ONFIGURATION MATERIEL
[Q] UITTER

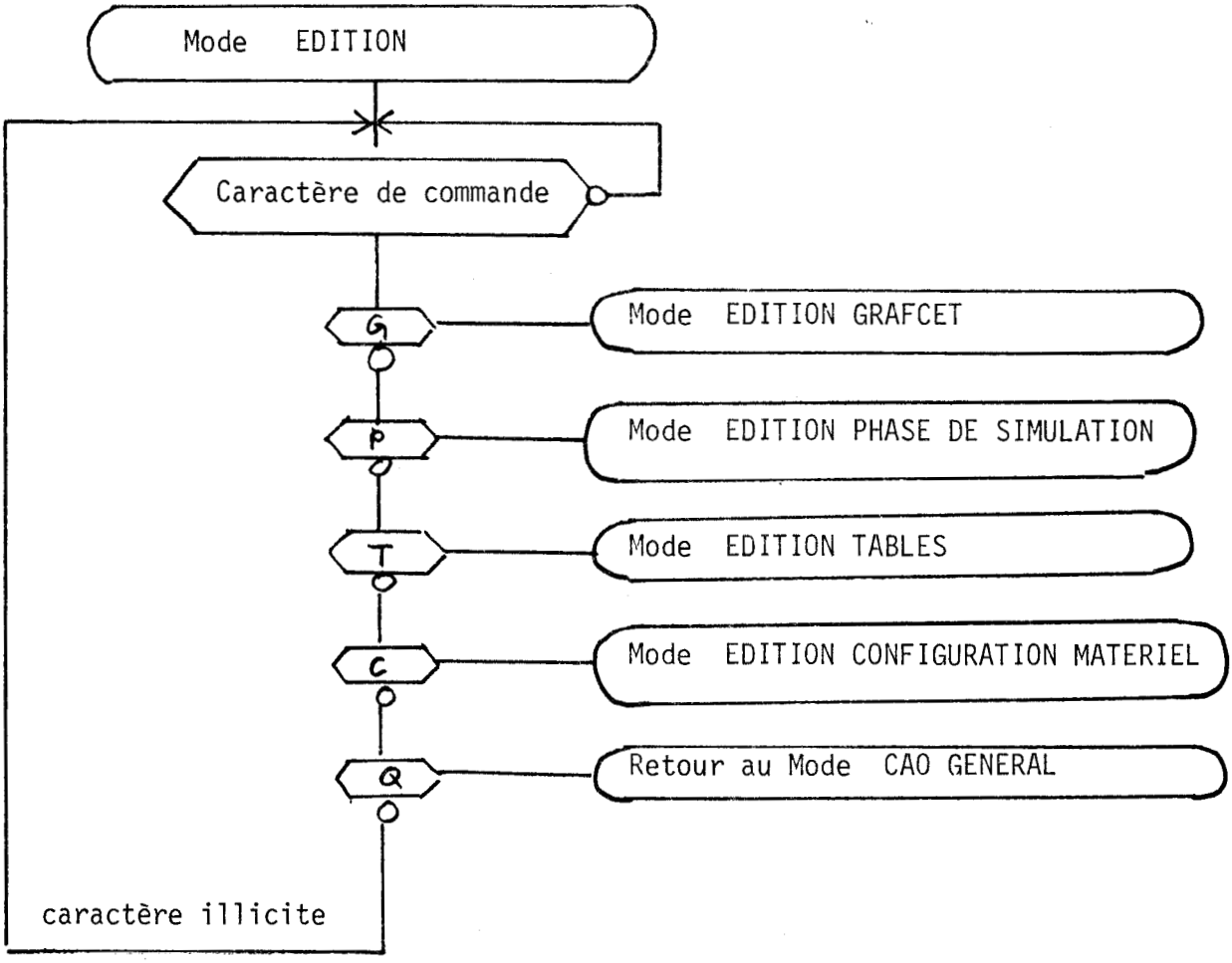


Fig II.9 Menu du mode EDITION

.../...

La commande [Q]UITTER permet de quitter le mode EDITION et assure le retour au mode CAO GENERAL. Les commandes [G]RAFCET, [P]HASE DE SIMULATION, [T]ABLES, et [C]ONFIGURATION MATERIEL permettent d'utiliser les modules EDITEUR du même nom. Le module EDITEUR GRAFCET est étudié en détail dans le paragraphe qui suit.

II.3 MODULE EDITION GRAFCET

Ce module permet de définir le Grafcet de l'application indépendamment du matériel et de la méthode de réalisation retenus. Sous cet éditeur l'utilisateur peut :

- créer un nouveau Grafcet
- charger un Grafcet précédemment entré pour le compléter ou le modifier et enfin le sauver sur disquette dans la bibliothèque des Grafcet.

La descrition du Grafcet comprend 4 parties :

(1) - la description du graphe proprement dite et son interprétation c'est à dire :

. les liaisons étapes - transitions et transitions étapes.

. les réceptivités associées aux transitions

. et les actions associées aux étapes

(2) - la situation initiale précisant les étapes initialement actives.

et éventuellement

(3) - la possibilité de désigner des fonctions combinatoires utilisés plusieurs fois dans l'interprétation du Grafcet par une variable interne.

(4) - la définition de sous Grafcet ou bloc par l'utilisateur, sans utiliser le module DECOMPOSITION.

Pour permettre une vérification croisée, la description du Grafcet est faite à la fois autour des transitions et des étapes.

La description autour des transitions donne pour chacune d'elles, les éléments suivants :

- n° transition
- Liste de ses étapes d'entrée
- liste de ses étapes de sortie
- réceptivité associée à la transition

La description autour des étapes donne pour chacune d'elles, les éléments suivants :

- n° d'étape
- liste de ses transitions d'entrée
- liste de ses transitions de sortie
- liste des sorties continues(éventuellement conditionnelles) contrôlées par son activité
- liste des sorties continues (éventuellement conditionnelles) démarrées dès son activation
- liste des sorties continues (éventuellement conditionnelles) arrêtées dès son activation
- listes des sorties impulsionnelles (éventuellement conditionnelles)
- identificateur de la variable interne qui matérialise l'activité de l'étape.

Remarquons que le choix des numéros de transitions et d'étapes est laissé à l'utilisateur,il a ainsi la possibilité d'effectuer des insertions, sans procéder à une renumérotation.

Les réceptivités, les conditions des sorties et les fonctions combinatoires d'ordre général sont des expressions logiques complexes

L'implantation d'expressions logiques comportant un bon nombre de parenthèses, de variables de front et de prédicats numériques risque de conduire à une impasse sur un matériel de bas de gamme.

Suivant la complexité du problème et du type de matériel retenu le système offre 4 possibilités qui sont :

classe 0 expressions logiques sans parenthèse et sans prédicat

classe 1 expressions logiques à 1 niveau de parenthèses et sans prédicat

classe 2 expressions logiques à plusieurs niveaux de parenthèses mais sans prédicats

classe 3 expressions logiques générales à plusieurs niveaux de parenthèses et avec possibilités de prédicats.

Le menu du mode EDITION GRAFCET est alors : (Fig. II.10)

[C]HARGER
[M]ODIFIER
[A]JOUTER
[E]CRIRE SUR ECRAN
[D]ETRUIRE
[S]AUVER
[I]MPRIMER
[B]IBLIOTHEQUE
[Q]UITTER

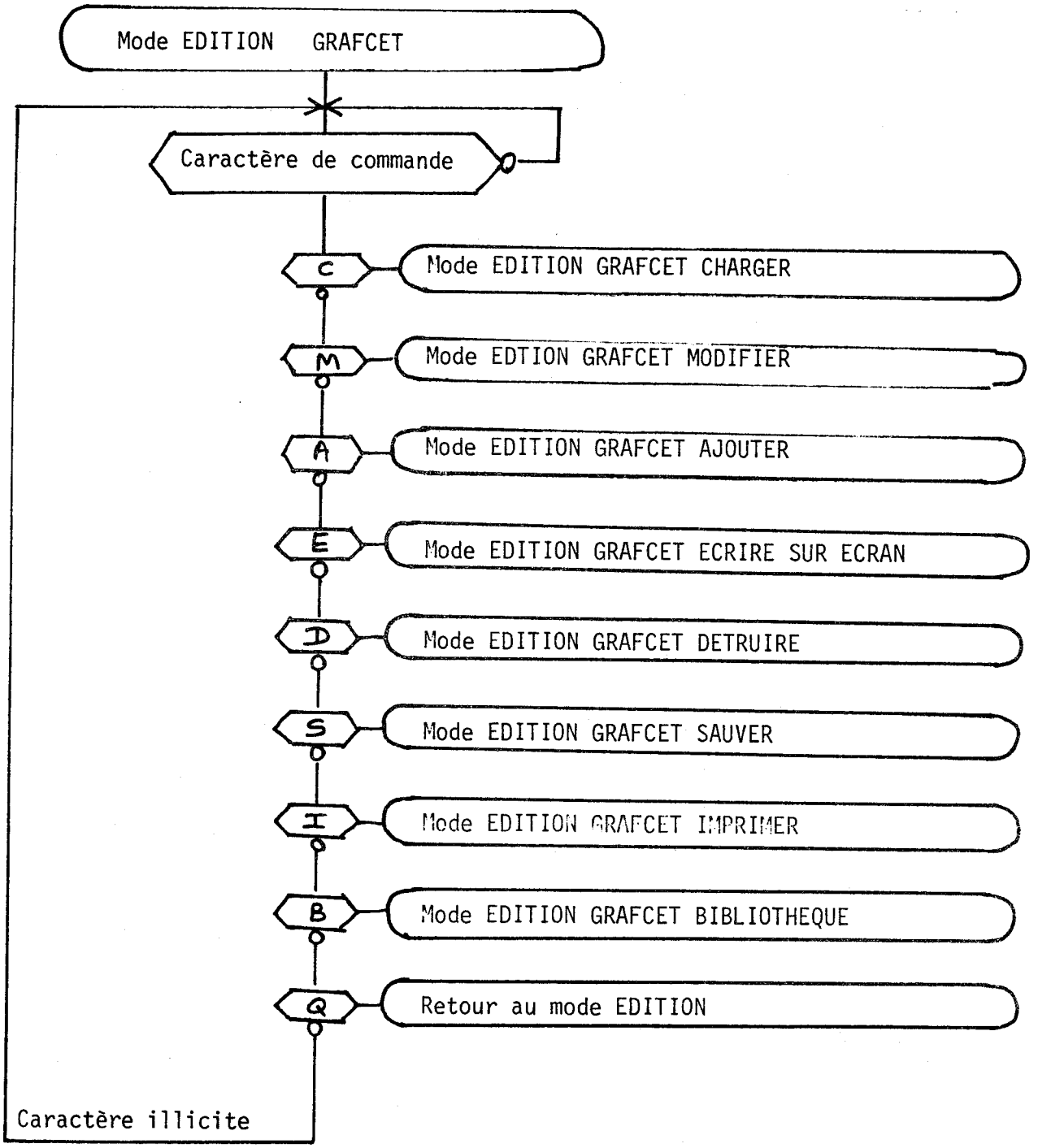


Fig II.10 Menu du mode EDITION GRAFCET

La commande [C]HARGER (Fig II.11) permet d'entrer en mémoire un grafcet qui figure dans la bibliothèque de l'une des unités de disquette, pour la compléter ou la modifier éventuellement. Dans ce cas l'utilisateur doit fournir :

- le nom du Grafcet
- son numéro éventuel
- et le numéro de la disquette où est rangé ce dernier

Pour les problèmes de taille importante, la description du grafcet peut ne pas résider complètement en mémoire l'édition en une seule étape est alors impossible. Le concepteur doit dans ce cas effectuer un découpage arbitraire, et procéder à l'édition séparée de ces différentes parties ; ces dernières doivent toutes recevoir le même nom, elles sont distinguées par leur numéro.

Si le système ne trouve pas le Grafcet dans la bibliothèque de la disquette, il laisse à l'utilisateur deux possibilités :

[C]ONTINUER
[Q]UITTER

La première lui permettra de corriger les qualificatifs du Grafcet, la seconde le renverra au mode EDITION GRAFCET où il pourra par exemple obtenir la liste des Grafcet de la disquette grâce à la commande [B]IBLIOTHEQUE.

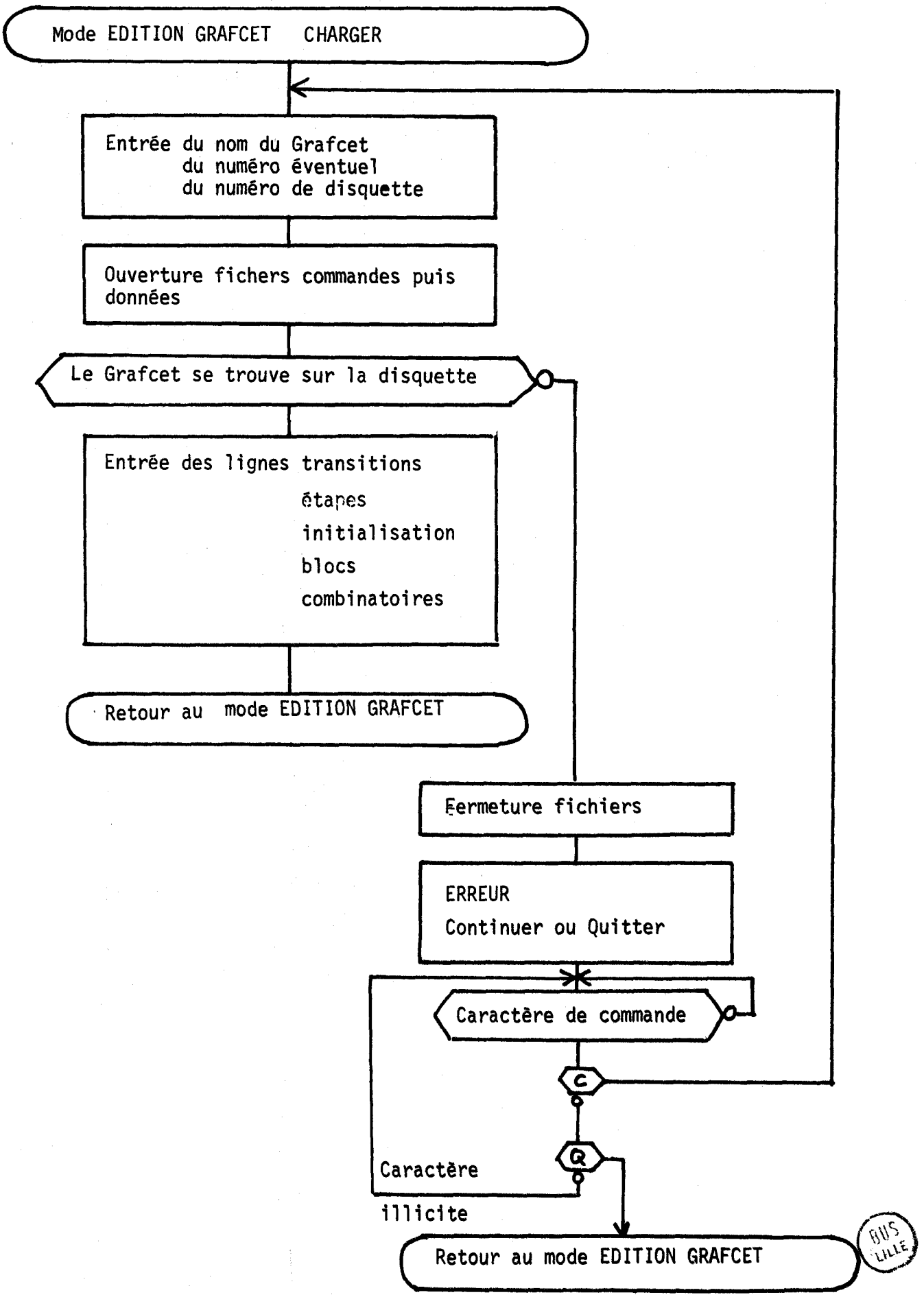


Fig II.11 mode EDITION GRAFCET CHARGER

Les commandes [S]AUVER et [I]MPRIMER permettent respectivement une sauvegarde sur disquette et une impression du Grafcet.

Les commandes [M]ODIFER, [A]JOUTER, [E]CRIRE SUR ECRAN et [D]ETRUIRE offrent le menu suivant (Fig II.12) :

[T]RANSITION
[E]TAPE
[I]NITIALISATION
[B]LOC
[C]OMBINATOIRE
[Q]UITTER

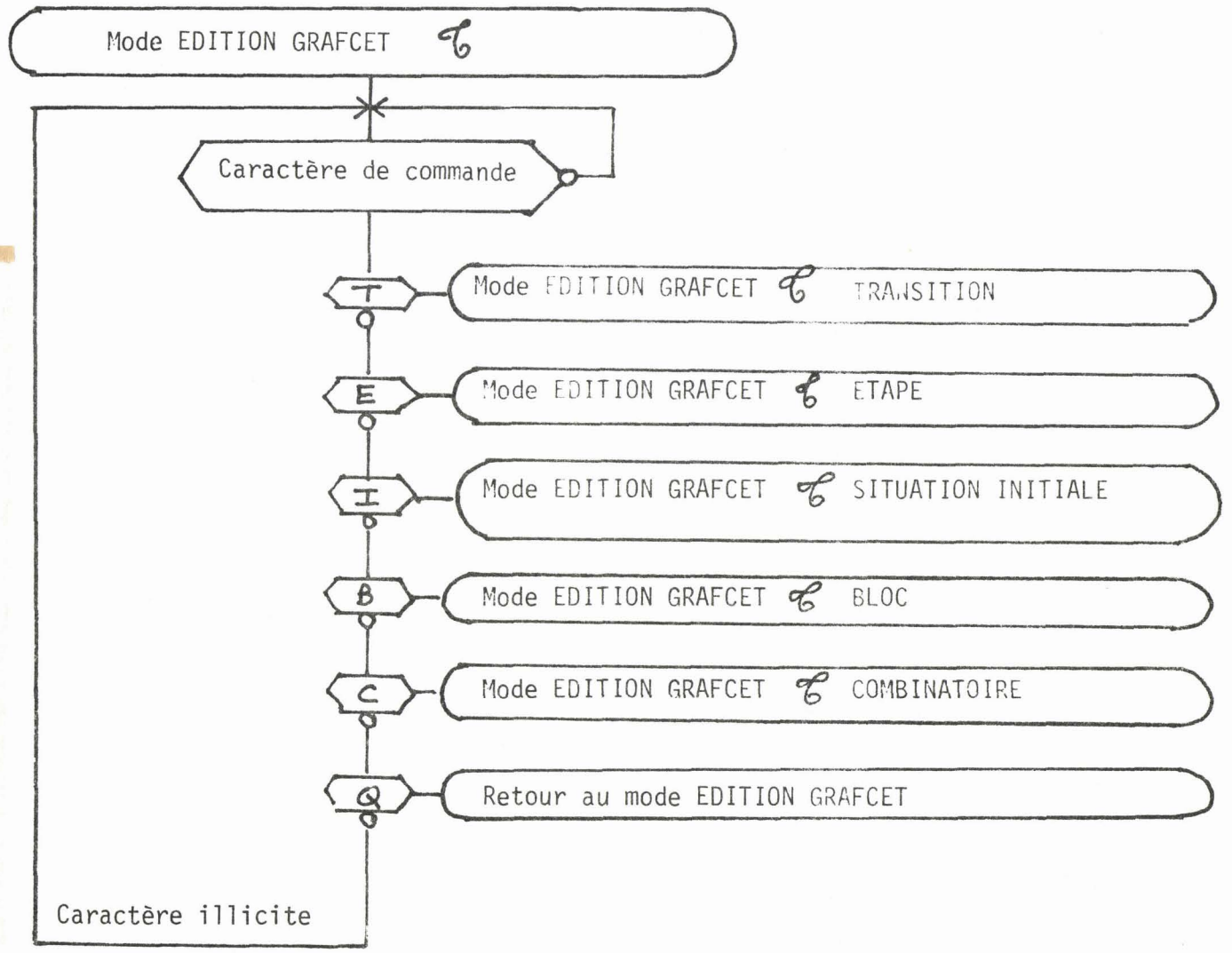


Fig II.12

Par exemple dans le mode EDITION GRAFCET TRANSITION, l'utilisateur doit entrer le numéro de la transition, la décision prise par le système est alors précisée par le tableau suivant :

	MODIFIER	AJOUTER	ECRIRE SUR ECRAN	DETRUIRE
Il existe une transition qui porte le numéro	correct	erreur (1)	correct	correct
AUCUNE transition ne porte ce numéro	erreur (2)	correct	erreur (2)	erreur (2)

En cas d'erreur le système imprime transition déjà entrée pour le cas (1) ou transition non trouvée pour le cas (2) et offre à l'utilisateur deux possibilités [C]ONTINUER permettant une modification du numéro de transition, [Q]UITTER renvoyant au mode EDITION GRAFCET.

Dans le cas contraire l'utilisateur peut réaliser la fonction retenue (MODIFER, AJOUTER, ECRIRE SUR ECRAN, DETRUIRE) sur l'élément choisi (TRANSITION, ETAPE, SITUATION INITIALE, BLOC, COMBINATOIRE).

Remarquons qu'après entrée ou modification d'un élément le système effectue une analyse syntaxique, en cas d'erreur il repositionne le curseur sur le premier caractère erroné.

Les autres modules sont construits sur le même principe. On détaillera dans les chapitres suivants, les modules DECOMPOSITION et TRADUCTION. Un module EDITEUR GRAPHIQUE est en cours d'étude. Toute adjonction de module est possible

CHAPITRE III

METHODE HEURISTIQUE DE DECOMPOSITION D'UN GRAFCET EN GRAPHES D'ETAT

Certaines méthodes de réalisation programmée ou câblée du Grafcet nécessitent une décomposition du Grafcet en graphes d'état , c'est à dire en sous Grafcet qui ne possèdent qu'au plus une étape active. Nous présentons dans ce chapitre une méthode heuristique de décomposition d'un Grafcet en graphe d'état .

III 1 METHODE DE REALISATION D'UN GRAPHE D'ETAT

En associant à chaque étape e une variable m_e qui représente son activité, nous pouvons déterminer les conditions de franchissement de chaque transition et en déduire la condition d'évolution d'un Grafcet. Si cette dernière est vérifiée, le calcul de la nouvelle situation se réduit dans le cas d'un graphe d'état à recopier dans chacune de ces variables, la condition d'activation de l'étape associée

$$\forall e \in E \qquad m_e = M_2^1 = \sum_{t \in {}^\bullet e} CF_t$$

La réalisation (Fig III.1) se fait alors suivant la procédure maître esclave :

1) Calcul et gel des conditions de franchissement

2) En cas d'évolution calcul de la nouvelle situation

...../...

La figure III 2 b) donne l'organigramme de traitement d'un Grafcet (Fig. III. 2 a).

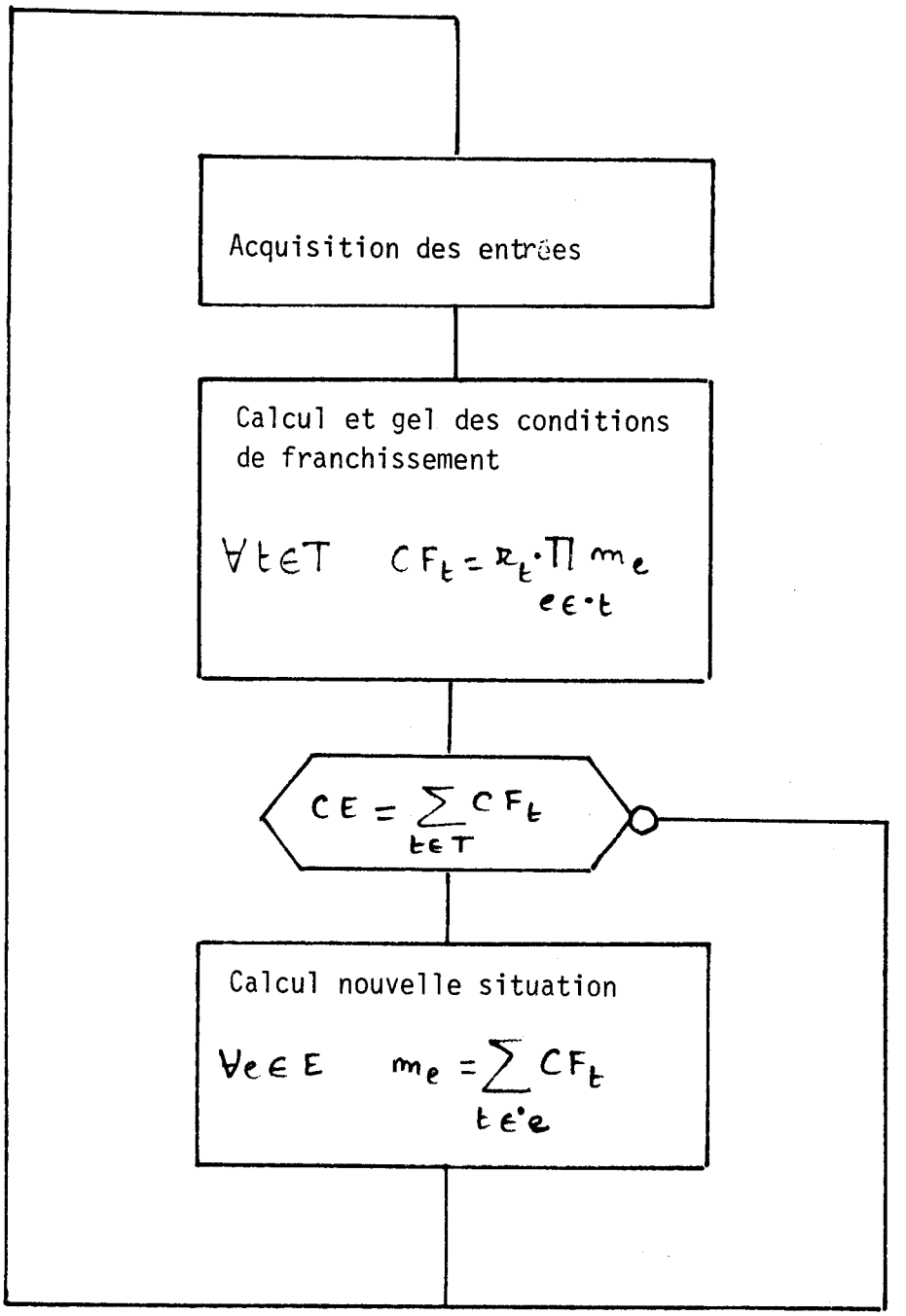
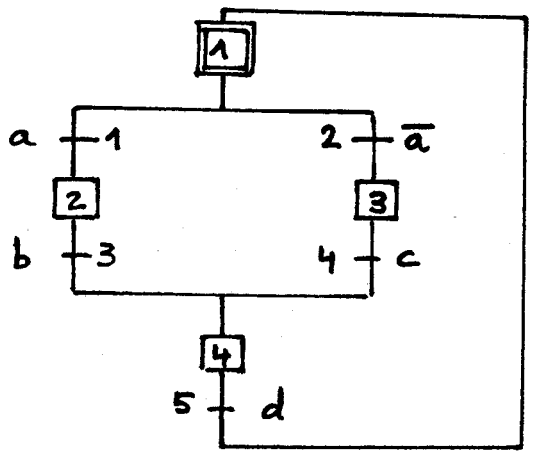
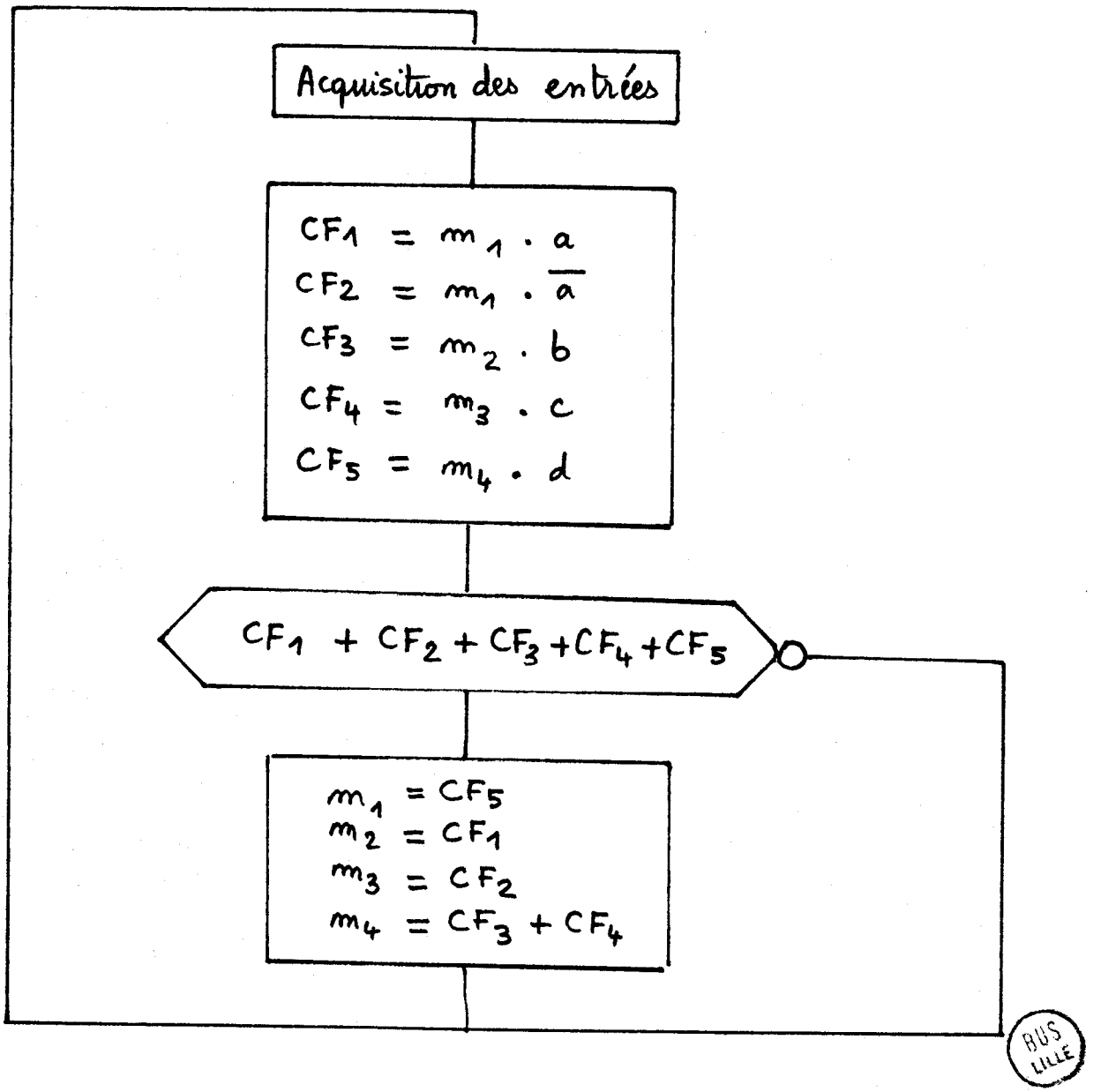


Fig . III 1



(a)



(b)

Fig III.2

Cette méthode ne convient pas pour un Grafcet qui peut comporter plusieurs étapes actives.En effet, dans la situation $\{2,3\}$ (Fig. III 3) lors du franchissement de la transition 3, l'étape 2 n'est pas maintenue active.

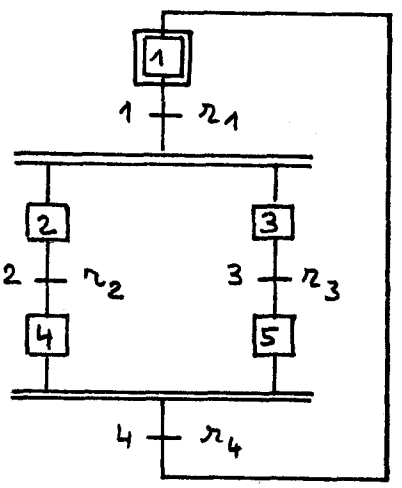


Fig. III 3

Pour appliquer cette méthode nous devons décomposer le Grafcet en graphes d'état

Nous précisons dans le paragraphe suivant la notion de partition d'un Grafcet en graphes d'état.

III 2 PARTITION D'UN GRAFCET EN GRAPHE D'ETAT (13) (19)

Pour un Grafcet défini par

. un graphe G
. une situation initiale S_0
. une interprétation J

.../...

une partition $\Pi = \{E_1, E_2, \ldots, E_n\}$ de l'ensemble de ses étapes définit n sous Grafcet

$$(G_k, S_{o_k}, \mathcal{T}_{o_k}) \qquad k = 1,2, \ldots, n$$

Le sous Grafcet défini par E_k admet pour graphe $G_k = (E_k, T_k, \alpha_k, \beta_k)$ où T_k est l'ensemble des transitions connectées à une étape de E_k, c'est à dire admettant au moins une étape de E_k comme entrée ou sortie:

$$\exists e \in E_k \qquad \alpha(e,t) = 1 \ \underline{\text{ou}} \ \beta(e,t) = 1$$

α_k et β_k sont les restrictions de α et β à $E_k \times T_k$.

La situation initiale S_{ok} est l'ensemble des étapes de E_k qui appartiennent à S_o.

$$S_{ok} = S_o \cap E_k$$

Si toutes les étapes d'entrée de la transition t appartiennent à E_k sa réceptivité r_t^k est égale à r_t, sinon elle est égale au produit logique de r_t par les variables associées à ses étapes d'entrée qui n'appartiennent pas à E_k.

$$r_t^k = r_t \cdot \prod_{e \notin E_k \text{ et } \alpha(e,t) = 1} m_e$$

Les actions associées aux étapes ne sont pas modifiées.

Remarquons que l'ensemble des n Grafcet ainsi définis a un comportement identique au précédent. En effet chaque occurence d'une même transition, dans les différents sous Grafcet ,a la même condition de franchissement ; comme en plus chaque étape conserve ses transitions d'entrée et de sorties, ses conditions d'activation et de désactivation sont identiques.

Pour le Grafcet de la Fig III. 3 la partition $\{1,2,3\}$, $\{3,5\}$ conduit aux deux sous Grafcet Fig III. 4.

Remarquons que l'ensemble des graphes d'état constitue un Grafcet de type II ; en effet les deux transitions 1 des deux sous Grafcet sont en conflit, alors que le Grafcet initial était de type I.

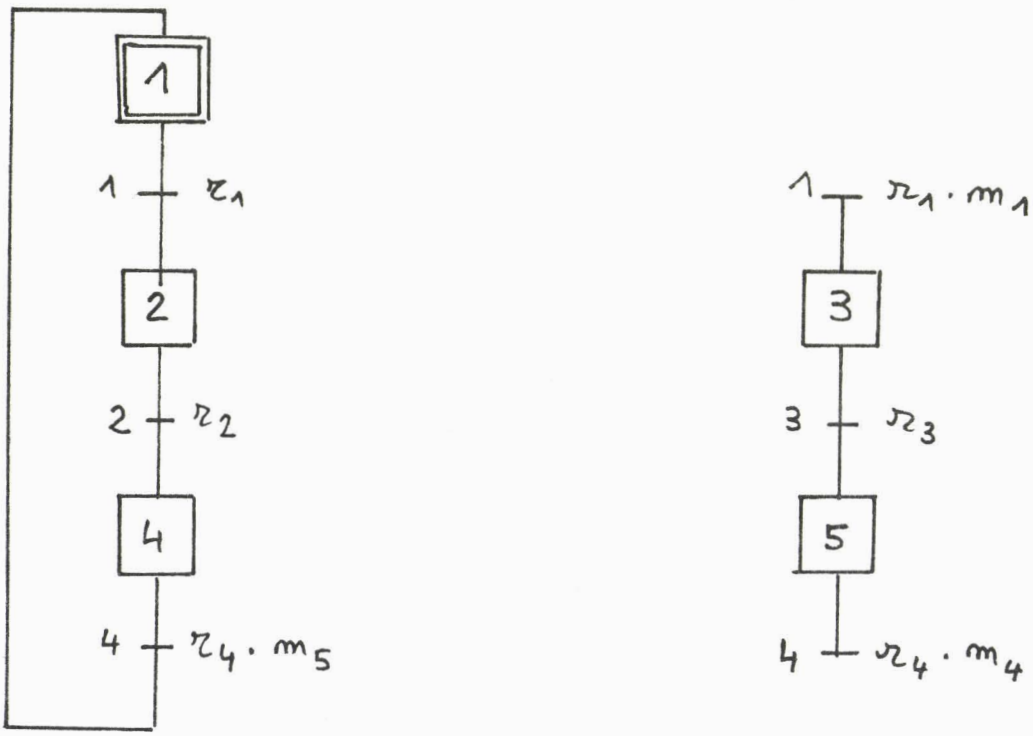


Fig III. 4

Nous cherchons à définir une partition de l'ensemble des étapes pour définir n Grafcet se réduisant chacun à un graphe d'état. Avant de présenter cette méthode montrons la procédure de réalisation.

.../...

III. 3 METHODES DE REALISATION D'UN ENSEMBLE DE GRAPHES D'ETAT (2) (3)

III - 3.1 Principe de la méthode

La condition d'évolution d'un sous Grafcet défini par E_k est la somme logique des conditions de franchissement des transitions de T_k

$$C\,E_k = \sum_{t \in T_k} C\,F_t$$

pour le graphe d'état défini par E_R la méthode de calcul de la nouvelle situation a été exposé dans le paragraphe I.

La méthode de réalisation d'un ensemble de graphes d'état est précisée par l'organigramme de la FigIII.5 Pour l'ensemble de Grafcet de la figure III. 4, l'organigramme est donné par la figure III. 6

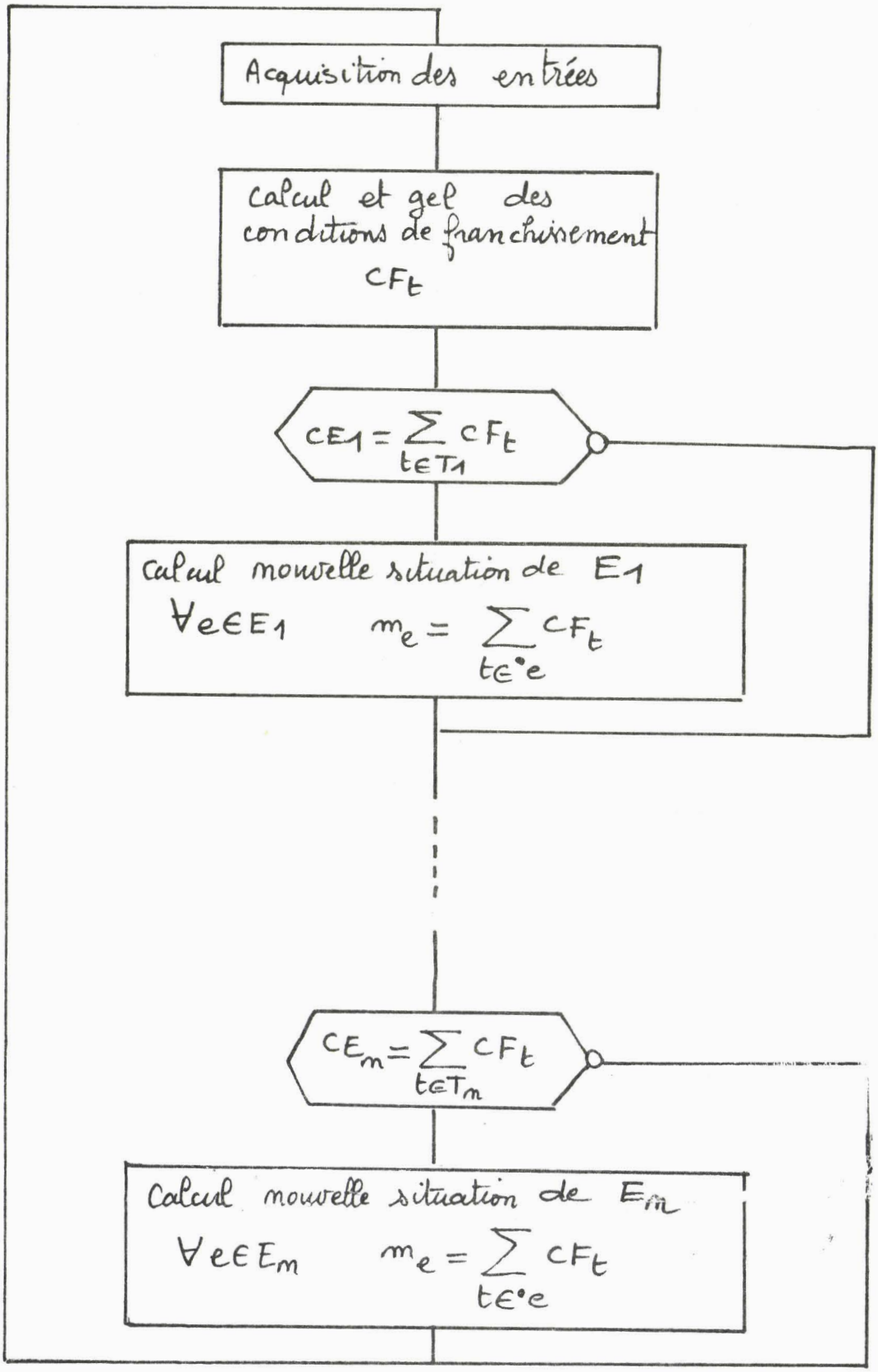


Fig. III 5

.../...

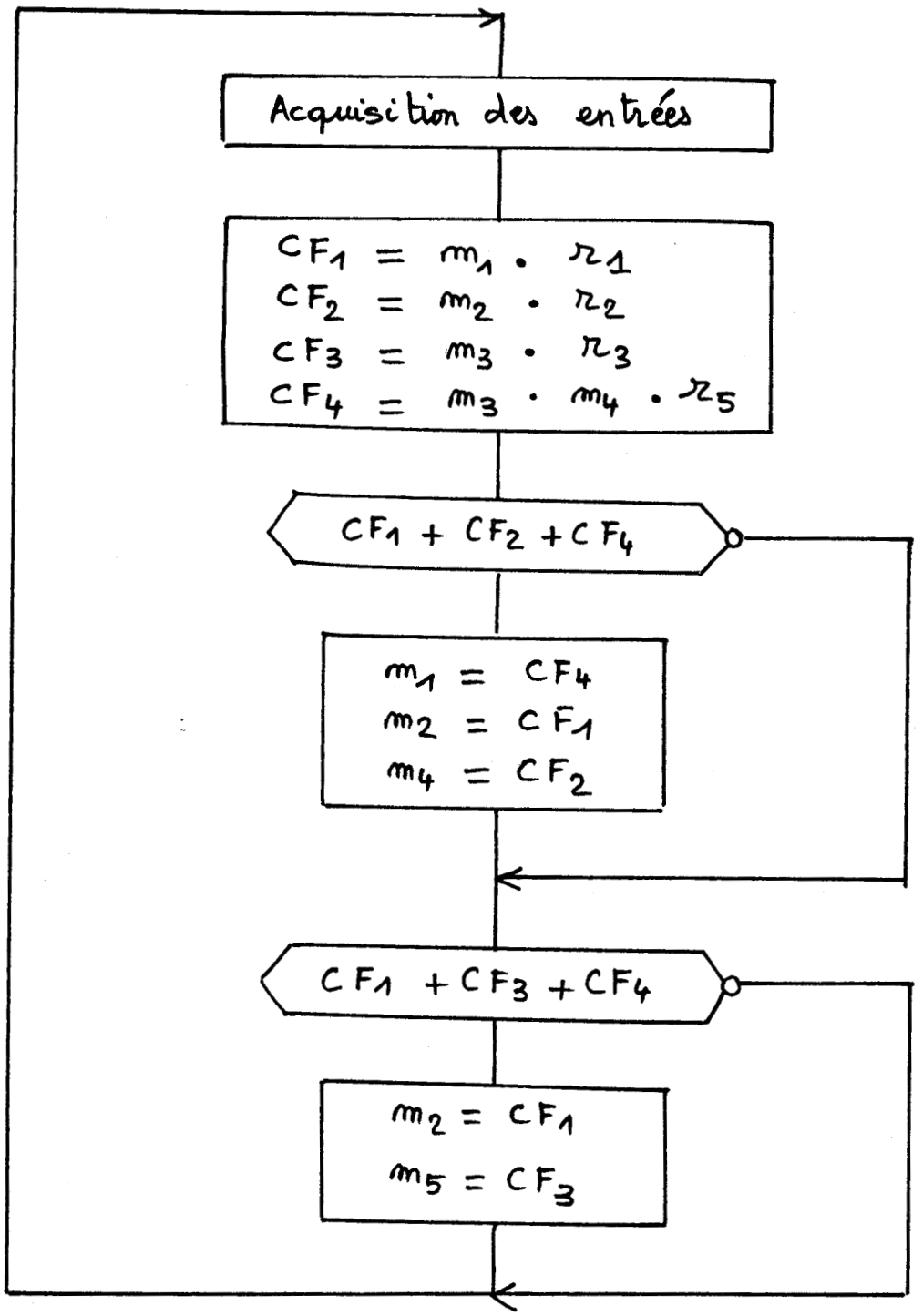


Fig. III 6

.../...

III. 3.2 Une réalisation câblée programmée Prograf (11)

Cette méthode peut conduire à une réalisation câblée programmée dans laquelle l'activité d'une étape est matérialisée par la sortie Q d'une mémoire synchrone de type D maître esclave. L'entrée d'une bascule est la condition d'activation de l'étape associée. La condition d'évolution de chaque graphe d'état commande un générateur d'horloge pilotant toutes les bascules associées à ses étapes. Les réalisations des exemples des figures III. 2 a et 3 sont données par les figures III. 7 et III. 8.

La détermination des conditions de franchissement à partir des réceptivités n'exige qu'un réseau E T. De même pour les conditions d'évolution et d'activation un réseau O U suffit. Le concepteur peut utiliser un réseau logique programmable de type FPLA, ou un panneau de programmation.

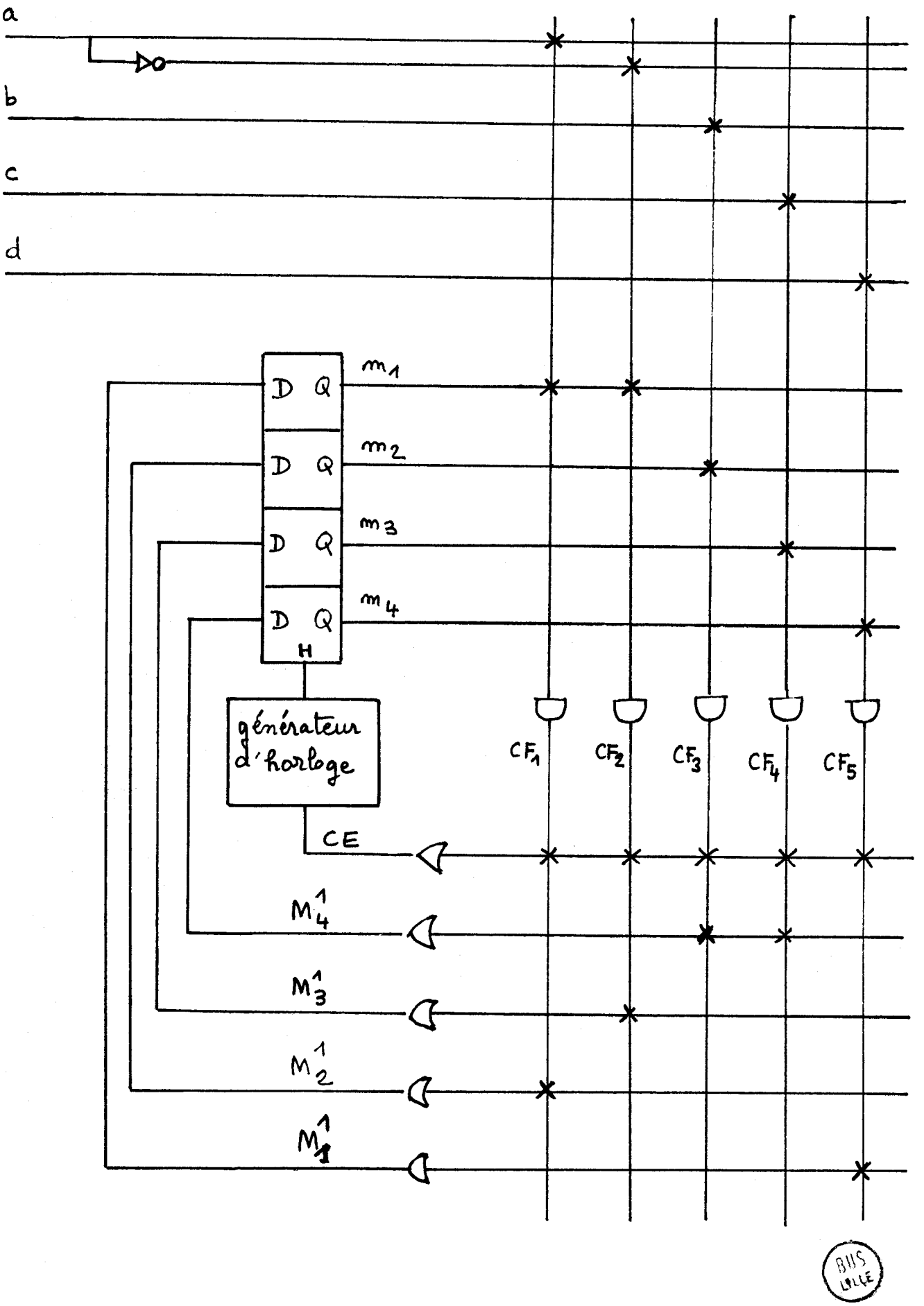


Fig. III.7

.../...

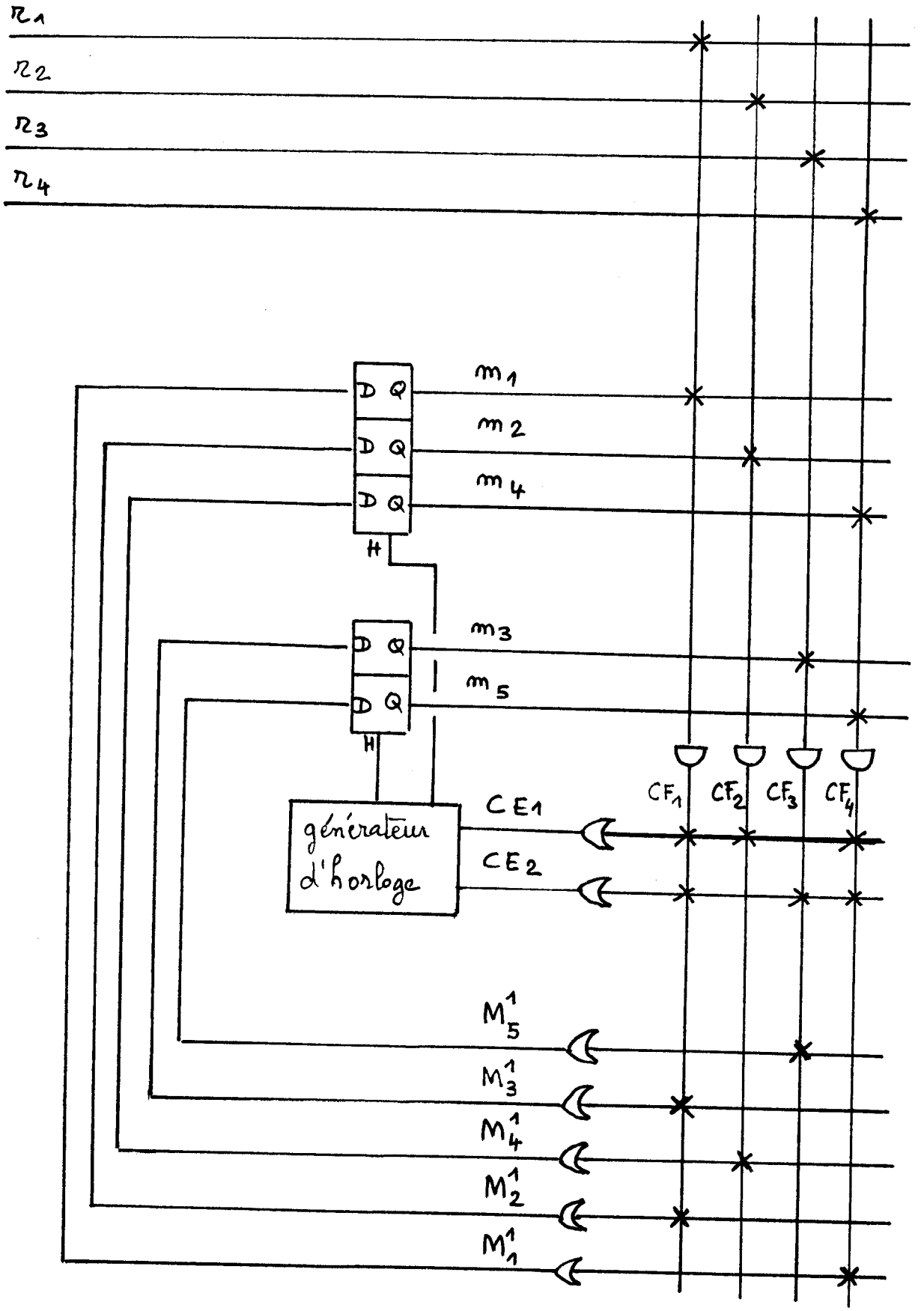

n_1
n_2
n_3
n_4
D Q
H
m_1
m_2
m_4
m_3
m_5
générateur d'horloge
CE1
CE2
CF_1
CF_2
CF_3
CF_4
M_5^1
M_3^1
M_4^1
M_2^1
M_1^1

Fig. III.8

.../...

La seconde solution a conduit au système PROGRAF (panneau de PROgrammation pour GRAphes fonctionnels) Ce dernier est un outil pédagogique qui permet la réalisation d'au plus trois graphes d'état ou branches. Les nombres de transitions et d'étapes doivent être inférieurs respectivement à 11 et 12. Le système Prograf (Fig. III. 9) est un panneau de programmation à trois plans, donc à deux niveaux de relation. Le premier permet la détermination de la condition de franchissement de chaque transition, le second, la condition d'activation de chaque étape. La partie droite du panneau permet l'élaboration des conditions d'évolution de branches, la partie inférieure précise l'appartenance des étapes aux branches. La condition de franchissement d'une transition est obtenue

Fig III. 9 a)

.../...

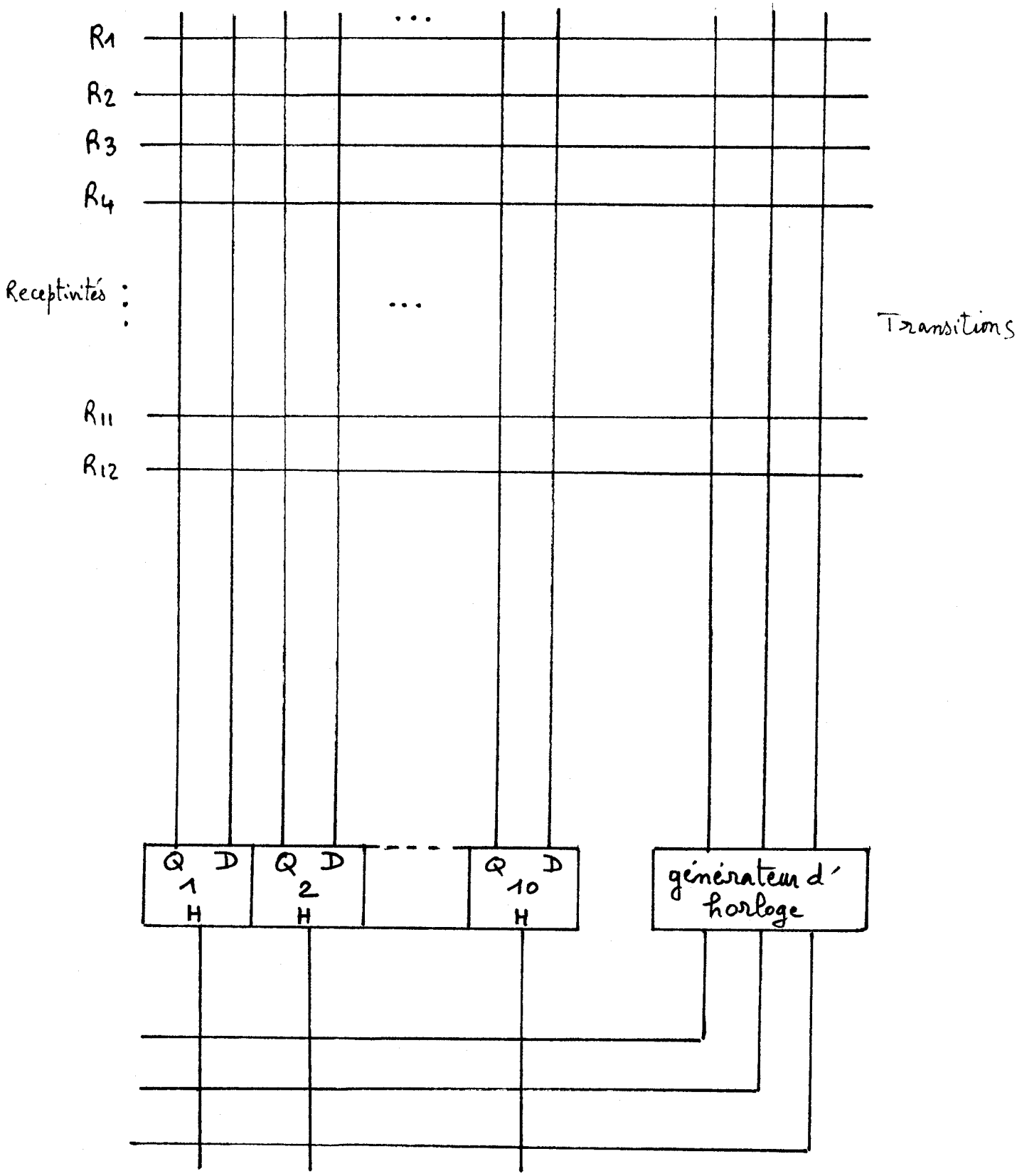
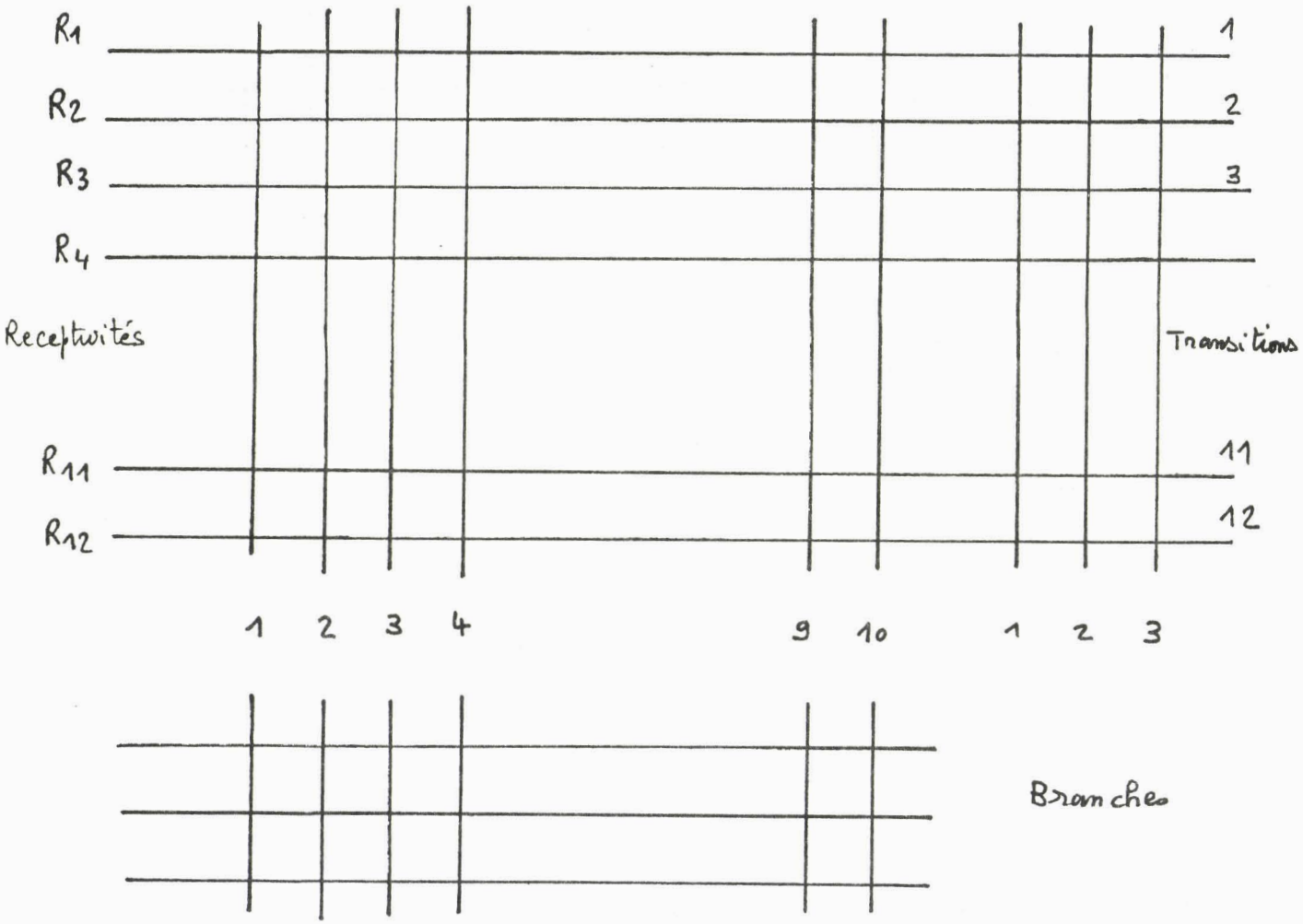


Fig. III.9 a



Fiche ET

Fiche OU

Fiche Court-circuit

Fig. III. 9 b)

.../...

en plaçant sur sa ligne une fiche ET à chaque colonne de ses étapes d'entrée. La condition d'activation d'une étape est obtenue en plaçant sur sa colonne une fiche OU à chaque ligne de ses transitions d'entrée. Pour une étape ne comportant qu'une transition d'entrée, cette fiche OU se réduit à un court circuit Ces deux niveaux ne dépendent que des liaisons étape-transition et transition-étape du Grafcet et matérialisent la matrice d'incidence du Grafcet. Dans cette solution il est impossible qu'une étape soit entrée et sortie d'une même transition ce qui ne constitue pas une restriction, car il est toujours possible d'effectuer la transformation indiquée sur la figure III. 10.

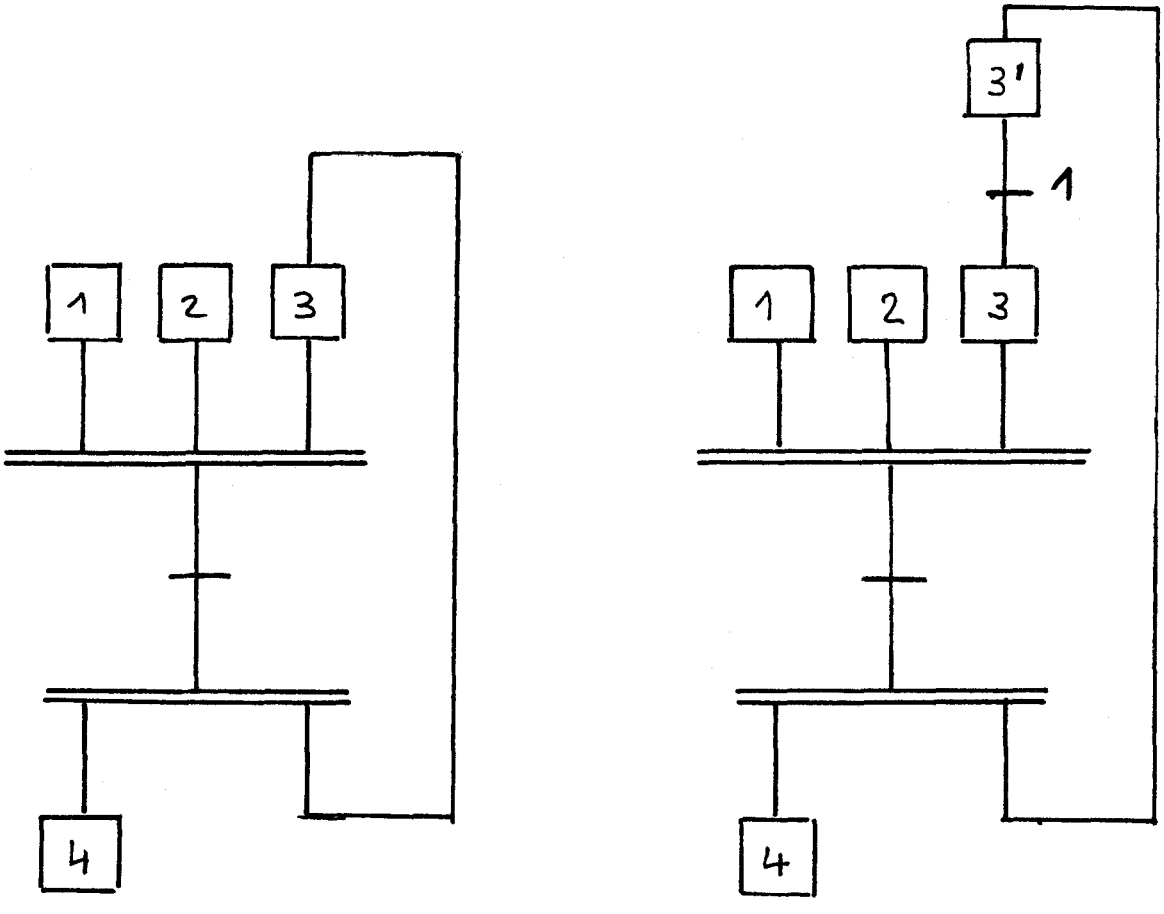


Fig III. 10

La mise sous tension active l'étape 1 .../...

Si le Grafcet comporte plusieurs étapes actives, le concepteur doit introduire une transition de réceptivité 1 admettant l'étape 1 comme entré et les étapes initialement actives comme sortie. (Fig. III. 11.)

La figure III. 13 donne les réalisations des exemples des figures III. 2 et III. 3

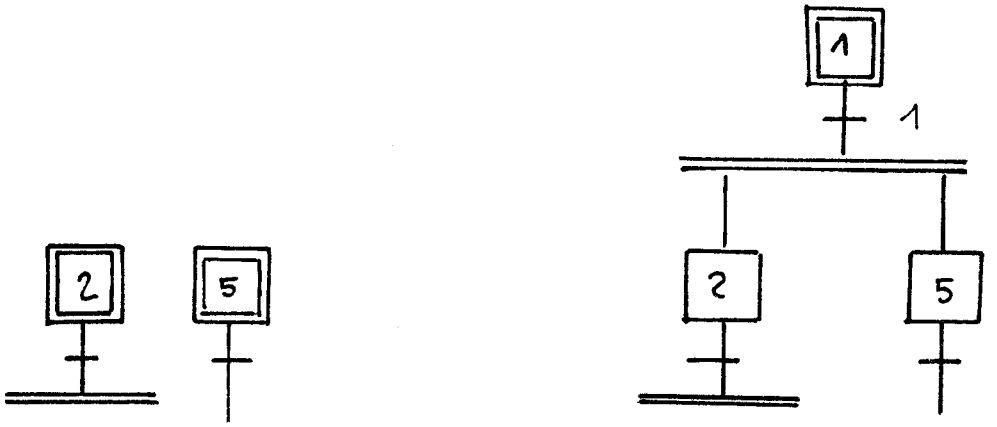


Fig III. 12

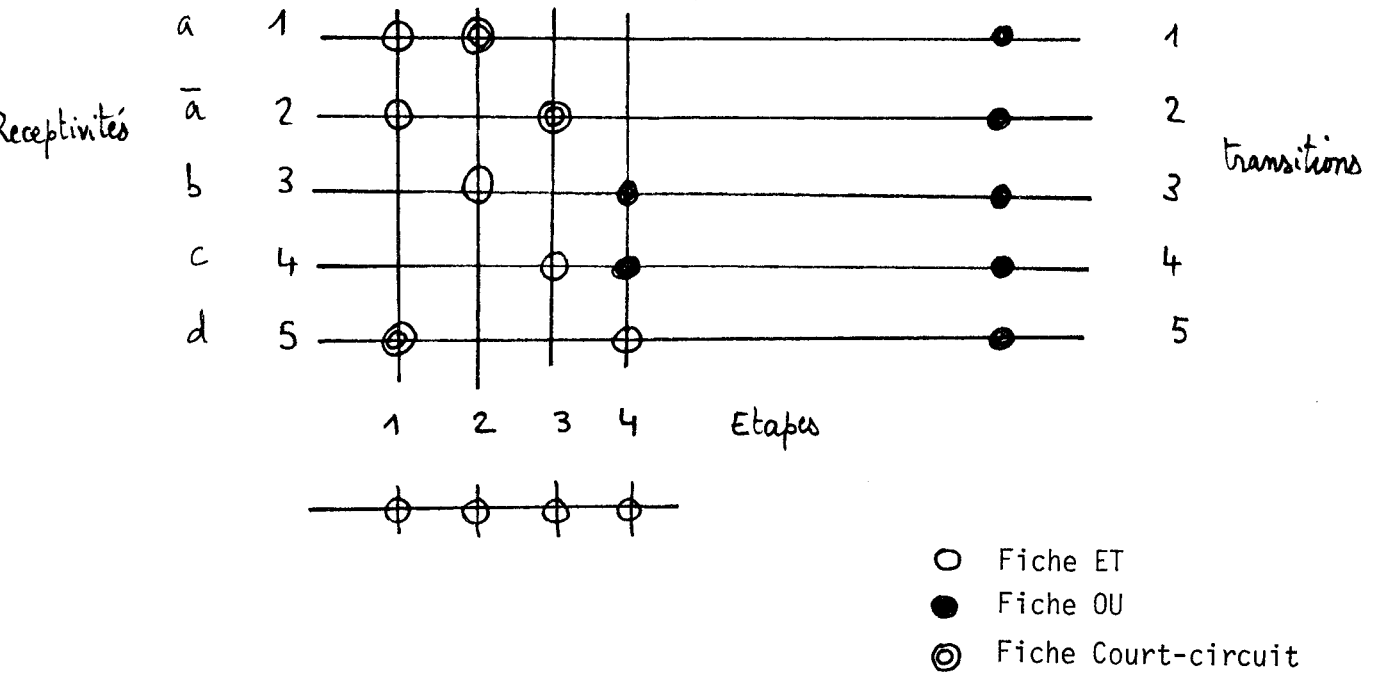


Fig.III. 13 a)

.../...

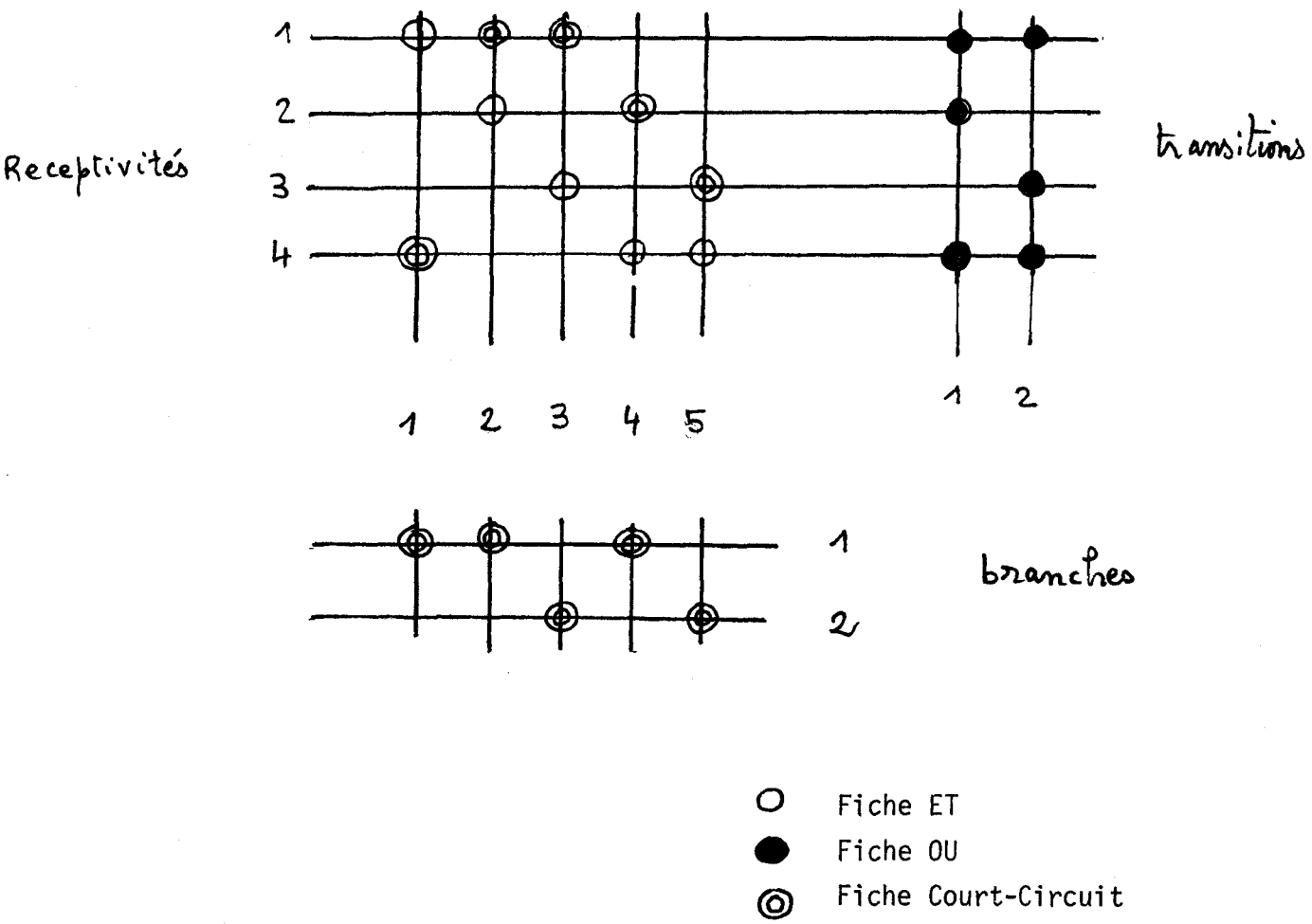


Fig III. 13 b)

III.3.3 CRITERES DE CHOIX DE LA PARTITIION

La partition triviale définie par $E_i = \{ e_i \}$ $i = 1,2, \ldots, n$ où chaque graphe d'état se réduit à une étape réalise bien une partition d'un Grafcet en graphes d'état. Mais une telle solution ne conduit pas une solution intéressante. Car dans ce cas le nombre de test à effectuer pour la solution programmée est maximum.

La solution câblée doit alors comporter autant de générateurs d'horloge que d'étapes.

Nous sommes ainsi amenés à rechercher une partition qui comporte le nombre minimal de sous ensembles.

Remarquons également que chaque test s'effectue sur la condition d'évolution du sous Grafcet :

$$CE_i = \sum_{t \in T_i} CF_t \qquad i = 1,2,\cdots,n$$

.../...

Pour réduire le calcul des conditions d'évolutions nous recherchons une partition qui minimise le nombre total de transitions connectées à chaque sous Grafcet $\sum_{i=1}^{m} |T_i|$ ($|T_i|$ représente le nombre d'éléménts de l'ensemble T_i)

Il est évident que la détermination de la partition en graphes d'état optimale pour ces deux artères risque de conduire à un algorithme peu performant. Car cette dernière nécessite en effet l'énumération de toutes les partitions en graphes d'état à partir de l'ensembe des situations accessibles. Nous proposons une méthode heuristique, basée sur l'examen des transitions qui ne nécessite pas l'énumération de toutes les situations accessibles. Le temps de traitement est proportionnel aux nombres de transitions.

Dans un premier temps, nous décrivons cette méthode pour les Grafcet de type I qui correspondent à la plupart des problèmes industriels.

III 4 METHODE DE PARTITION D'UN GRAFCET DE TYPE I EN GRAPHES D'ETAT

III.4.1 Classement des transitions

Nous proposons un classement des transitions en considérant le nombre d'étapes d'entrée noté $|{}^\bullet t|$ et le nombre d'étapes de sortie noté $|t^\bullet|$

Une transition source est une transition sans étape d'entrée ($|{}^\bullet t| = 0$) et une transition puits est une transition sans étape de sortie ($|t^\bullet| = 0$)

Une transition transfert possède une étape d'entrée et une étape de sortie. ($|{}^\bullet t| = 1$ et $|t^\bullet| = 1$)

Pour une transition ET appelée aussi noeud ET le nombre d'étapes d'entrée ou de sortie est strictement supérieur à 1.

($|{}^\bullet t| > 1$ ou $|t^\bullet| > 1$)

Nous distinguons deux types de noeud ET particuliers, le noeud ET de distribution où $|{}^\bullet t| \leq 1$ et $|t^\bullet| > 1$

.../...

et le noeud ET de synchronisation où $|{}^{\bullet}t| \geq 1$ et $|t^{\bullet}| = 1$

Remarquons qu'il existe des noeuds ET qui ne sont pas de distribution ni de synchronisation.

III 4.2 Remarques préliminaires

La méthode de décomposition proposée est indépendante de l'interprétation du Grafcet , elle ne dépend que de la structure du graphe.

Nous considérons un Grafcet ne comportant qu'une seule étape initialement active.

La partition recherchée doit être telle que chaque sous Grafcet obtenu ne comporte pas de noeud ET.Cette condition nécessaire n'est pas suffisante comme le montre l'exemple de la figure III.14.

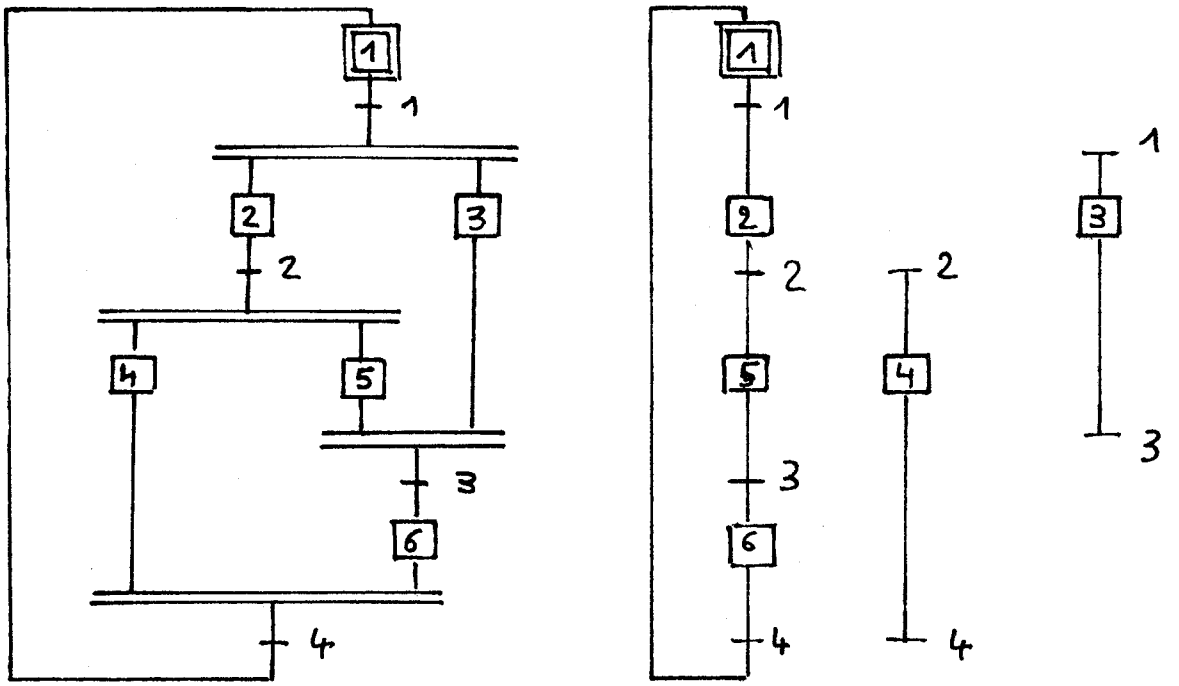


Fig III. 14

Les sous Grafcet définis par $E_1 = \{1,2,5,6\}$ et $\{3,4\}$ ne comportent pas de noeuds ET, mais dans le sous Grafcet défini par $\{3,4\}$ les étapes 3 et 4 peuvent être simultanément actives. Ce sous Grafcet n'est pas en un seul morceau.

Un Grafcet sans noeud ET peut être considéré comme un graphe orienté (E,U) au sens habituel. L'ensemble des sommets est l'ensemble des étapes E. Il existe un arc (e_i, e_j) entre sommets e_i et e_j s'il existe une transition t_k admettant e_i comme unique entrée et e_j comme unique sortie :

$$(\alpha(e_i, t_k) = 1 \quad \underline{\text{et}} \quad \beta(e_j, t_k) = 1)$$

Une chaîne de longueur k est une suite d'arcs $u_1, u_2, \ldots, u_k$ telle que chaque arc de la séquence posséde une extrémité en commun avec l'arc précédent et l'autre extrémité avec l'arc suivant. (Fig. III. 15)

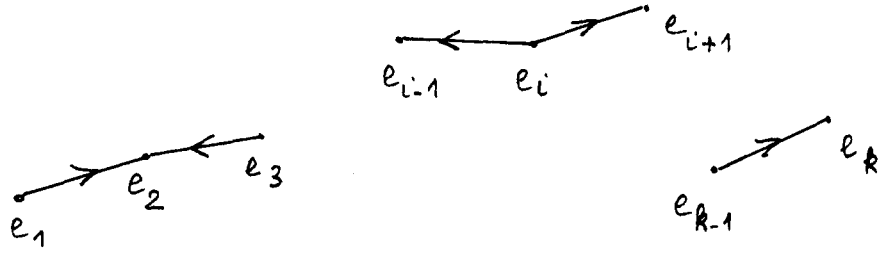


Fig III. 15

Un graphe est connexe si pour toute paire de deux sommets distincts, il existe une chaîne reliant ces deux sommets.

La relation de connexité permet de définir une relation d'équivalence.

$x \equiv y \iff [x=y$ ou il existe une chaîne reliant x à $y]$

Les classes de cette relation d'équivalence constituent une partition de sous graphes connexes de G appelées composants connexes.

Pour l'exemple de la Fig. III. 14 la partition en composantes connexes est E_1, E_2, E_3

.../...

avec $E_1 = \{1,2,5,6\}$ $E_2 = \{3\}$ $E_3 = \{4\}$ réalise une décomposition en trois graphes d'état.

III 4.3 Ensemble conséquent d'une étape

Soit e une étape, nous définissons l'ensemble des étapes pouvant être activées à partir de e sans franchir de transition ET. Cet ensemble noté $\vec{e}$ est appelé ensemble conséquent de e. Le sous Grafcet défini par e ne possède pas de noeud ET ; pour un Grafcet de type I sans transition source le sous Grafcet ainsi défini ne possède qu'au plus une étape active.

La détermination de l'ensemble conséquent d'une étape est déterminée par l'algorithme suivant :

1 initialiser la liste courante à vide
2 l'étape e est l'étape courante x
3 placer l'étape courante dans la liste L
4 classer toutes les transitions de sortie de l'étape courante x en 3 sous ensembles

Sa désigne les transferts
Sb les noeuds ET de distribution
Sc les noeuds ET de synchronisation
Sd les noeuds ET quelconques

5 si Sa est vide alors fin (la liste L contient $\vec{e}$)
6 ôter une transition t de Sa
7 l'étape de sortie de t devient étape courante
8 aller à 3

III 4.5 Obtention de la partition par l'examen des noeuds ET

A partir de l'étape initiale nous examinons les transitions dans l'ordre de leur franchissement possible.

L'algorithme démarre avec la partition triviale $\Pi^{(1)} = \{ E^{(1)} \}$ avec $E^{(1)} = E$

Pas à pas nous construisons une partition :

$$\Pi^{(k)} = \left\{ E^{(k)} ; B_1^{(k)}, B_2^{(k)}, \ldots, B_{q(k)}^{(k)} \right\}$$

telle que les sous Grafcet définis par les ensembles d'étapes ou blocs $B_i^{(k)}$ $(i = 1,2, \ldots, q(k))$ ne possèdent pas de noeud ET, seul le sous Grafcet défini par le bloc $E^{(k)}$ peut éventuellement posséder des noeuds ET.

Si e_1 est l'étape initialement active, la détermination de $\overrightarrow{e_1}$ donne également les ensembles de transitions $S_b^{(1)}$, $S_c^{(1)}$ et $S_d^{(1)}$ ce qui constitue, le point de départ de la procédure.

Le franchissement d'un noeud ET de distribution active plusieurs étapes simultanément et nécessite une modification de la partition $\Pi^{(k)}$. Soit un noeud ET de $S_b^{(k)}$ et $e_1, e_2, \ldots, e_\alpha$ ses étapes de sortie, nous déterminons les conséquents de ces dernières ainsi que $S_b^{(k+1)}$, $S_c^{(k+1)}$, $S_d^{(k+1)}$

Nous effectuons la partition de $E^{(k)}$ suivante :

$$A_j = \overrightarrow{e_j} \quad \text{pour} \quad j \in [1,\alpha] \text{ et } j \neq i$$

$$A_i = E^{(k)} - \bigcup_{\substack{j=1 \\ j \neq i}}^{\alpha} \overrightarrow{e_j}$$

Le choix de i est fait en vue de rendre minimal le nombre total de transitions aux sous Grafcet définis par les A_i. La partition $\Pi^{(k)}$ devient $\Pi^{(k+1)} = \left\{ E^{k+1}, B^{(k+1)}, B^{(k+1)}, \ldots, B^{(k+1)}_{q(k+1)} \right\}$ avec $E^{(k+1)} = E^{(k)} - \bigcup_{\substack{j=1 \\ j \neq i}}^{\alpha} \overrightarrow{ej}$

pour $j=1,\ldots, q(k)$ $\quad B_j^{(k+1)} = B_j^{(k)}$

.../...

pour $j=q(k) + 1,\ldots q(k)+i-1 \quad B_j^{(k+1)} = \overrightarrow{e_j} - q(k)$

pour $j=q(k) + i,\ldots, q(k)+\alpha -1 \quad B_j^{(k+1)} = \overrightarrow{e_j} - q(k)+1$

$$q(k+1) = q(k) + \alpha - 1$$

Nous pouvons remarquer que si le Grafcet ne présente pas de réactivation d'étape les $\overrightarrow{e_i}$ sont disjoints deux à deux. Si il existe une étape e appartenant à deux ensembles conséquents d'étapes de sortie d'un noeud ET e peut être active à partir de e_i sans désactiver e_j. Dans ce cas nous devons réduire les conséquents e_i à $\overrightarrow{e_i} - (\overrightarrow{e_1} \cup \overrightarrow{e_2} \cup \ldots \cup \overrightarrow{e_{i-1}})$ pour $i = 1,2\ldots \alpha$

Ces derniers sont bien ainsi disjoints deux à deux.

Les sous Grafcet définis par les blocs $B_j^{(k+1)}$ peuvent posséder des noeuds ET. Ainsi sur l'exemple de la Fig. III.16

$$\overrightarrow{e_1} = \{1\} \qquad S_b^{(1)} = 1 \qquad S_c^{(1)} = \emptyset \qquad S_d^{(1)} = \emptyset$$

L'examen du noeud ET conduit à la recherche des conséquents des étapes e_2 et e_3

$$\overrightarrow{e_2} = \{2,4\} \qquad \overrightarrow{e_3} = \{3,5,6\}$$

et $S_b^{(2)} = \emptyset \qquad S_c^{(2)} = \{6,7,8\} \qquad S_d^{(2)} = \emptyset$

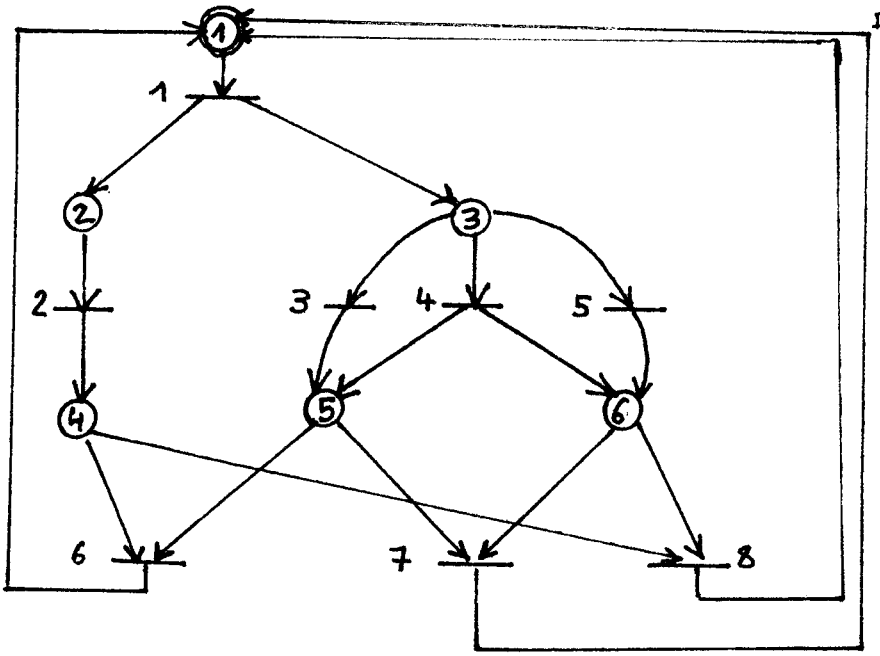
Nous avons le choix entre deux partitions

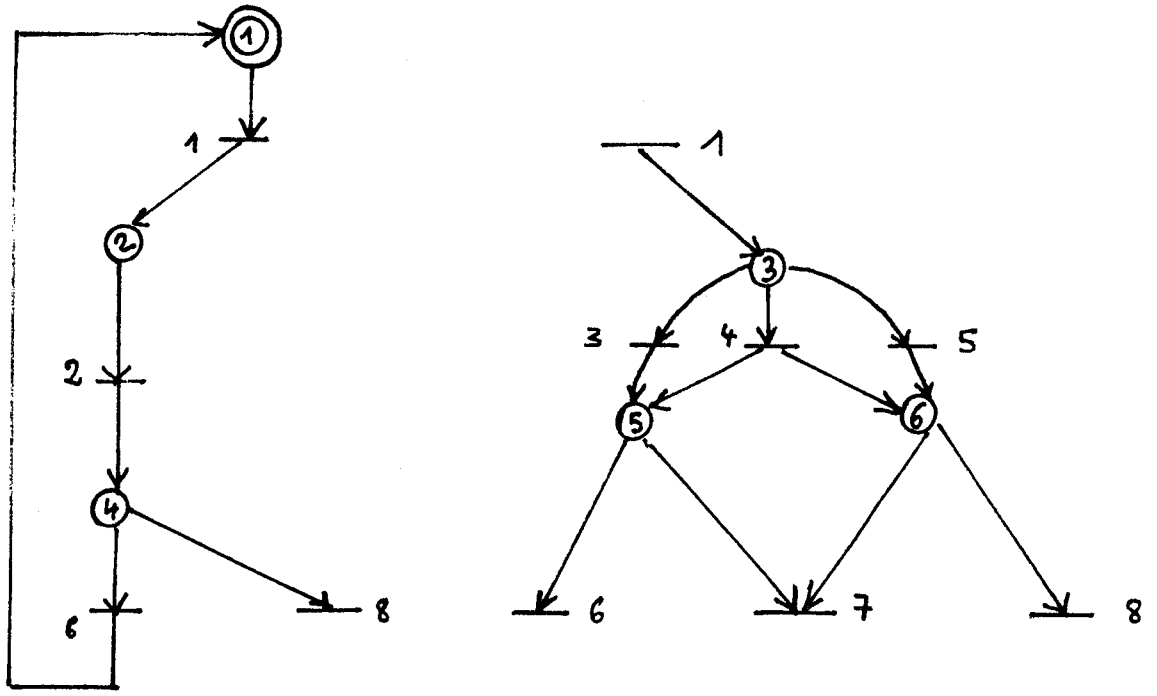
$\{\{1,2,4\} ; \{3,5,6\}\}$

$\{\{1,3,5,6\}; \{2,4\}\}$ de coût identique 11

Dans la première le Grafcet défini par le bloc $\{3,5,6\}$ comporte encore un noeud ET

.../...





GO

Fig III. 16

Dans ce cas, nous devons à nouveau effectuer une partition sur ce bloc

A la suite de la partition, nous reclassons éventuellement les transitions des ensembles $S_c^{(k+1)}$, $S_c^{(K+1)}$, $S_d^{(k+1)}$ en fonction de la structure du sous réseau défini par

.../...

$E^{(k+1)}$ quand $S_b^{(k)}$ est vide, nous devons considérer les noeuds ET de synchronisation soit $t \in S_c^{(k)}$ une transition de synchronisation du sous réseau défini par $E^{(k)}$ et $e_1, e_2, \ldots e_\beta$ ses étapes d'entrées. Une étape e_j de t peut être simultanément active avec les autres étapes d'entrée, mais aussi avec une autre étape de $E^{(k)}$ si e_j n'est pas connectée au sous Grafcet $E^{(k)}$.

Nous sommes ainsi amenés à distinguer deux cas :

1) si il existe une étape d'entrée e_i de t connectée à $E^{(k)}$, nous effectuons la partition de en sous ensembles suivante:

$$A_j = \{e_j\} \qquad j \in [1, \beta] \quad \text{et} \quad j \neq i$$

$$A_i = E^{(k)} - \bigcup_{j=1}^{\beta} e_j$$

$$j=1$$

La partition $\Pi^{(k)}$ devient $\Pi^{(k+1)}$

avec $E^{(k+1)} = E^{(k)} - \bigcup_{j=1}^{\beta} \{e_j\}$

$$j=1$$

pour $j = 1, \ldots q^{(k)}$ $\quad B_j^{(k+1)} = B_j^{(k)}$

pour $j = q^{(k)} + 1, \ldots, q^{(k)} + i - 1$ $\quad B_j^{(k+1)} = \{e_j - q^{(k)}\}$

pour $j = q^{(k)} + i, \ldots, q^{(k)} + \beta - 1$; $B_j^{(k+1)} = \{e_j - q^{(k)} + 1\}$

$$q^{(k+1)} = q^{(k)} + \beta - 1$$

2) Si il n'existe aucune étape d'entrée de e_i connectée à $E^{(k)}$ nous effectuons la partition de $E^{(k)}$ nous effectuons la partition de $E^{(k)}$ en $(\beta + 1)$ sous ensembles

.../...

$$A_j = \{ e_j \} \qquad j \in [1, \beta]$$

$$A_{\beta+1} = E^{(k)} - \bigcup_{j=1}^{\beta} \{ e_j \}$$

La partition $\Pi^{(k)}$ devient $\Pi^{(k+1)}$ avec :

$$E^{(k+1)} = E^{(k)} - \bigcup_{j=1}^{\beta} e_j$$

pour $j = 1, \ldots, q^{(k)}$ $\qquad B_j^{(k+1)} = B_j^{(k)}$

pour $j = q^{(k)+1}, \ldots, q^{(k)}+\beta$ $\qquad B_j^{(k+1)} = \{ e_j - q(k) \}$

$$q^{(k+1)} = q^{(k)} + \beta$$

Si e est l'étape de sortie de t, nous plaçons dans les ensembles $S_a^{(k+1)}$, $S_b^{(k+1)}$, $S_c^{(k+1)}$, $S_d^{(k+1)}$ les transitions de sortie de cette étape.

Remarquons que l'algorithme procède éventuellement à un reclassement en fonction de la nature du sous Grafcet $E^{(k+1)}$. Si après ce nouveau classement les ensembles $S_a^{(k+1)}$, $S_b^{(k+1)}$, $S_c^{(k+1)}$ sont vides, nous considérons l'examen des noeuds ET généraux. Pour ces derniers, nous devons appliquer le traitement des noeuds de synchronisation, puis celui des distributions.

A la fin de l'algorithme, nous devons vérifier la connectivité du Grafcet $E^{(f)}$; s'il n'est pas connexe, en appelant $A_1, A_2, \ldots, A_p$ ses composantes connexes la partition finale est

$$\Pi = \{ A_1 ; B_1^{(\ell)}, \dots B_{q(\ell)}^{(\ell)}, A_2, \dots A_p \}$$

Elle sera notée dans la suite de l'exposé

$$\Pi = \{ C_1, C_2 \dots, C_q \}$$

III. 4-5 Fusion des blocs

Les sous Grafcet C_i sont sans noeuds ET et connexes. Deux sous Grafcets définis par C_i et C_j sont dits fusionnables, si le Grafcet défini par $C_i \cup C_j$ ne comporte pas de transitions ET et est connexe. Pour cela il est nécessaire qu'une transition puits de l'un soit source de l'autre pour faciliter les recherches, nous pouvons classer les transitions d'un bloc C_i en :

- transition source ou départ
- transition transfert ou interne
- transition puits ou stop.

III. 5 PROBLEMES POSES PAR LES GRAFCETS DE TYPE II

Le premier problème est constitué par la présence des noeuds OU (c'est à dire des étapes qui admettent plusieurs transitions de sortie). En effet pour le Grafcet de la figure III.17 si dans la situation où l'étape 1 est active et les réceptivités r_1 et r_2 sont vérifiées, les transitions 1 et 2 doivent être simultanément franchies, ce qui conduit à la situation (2, 3).

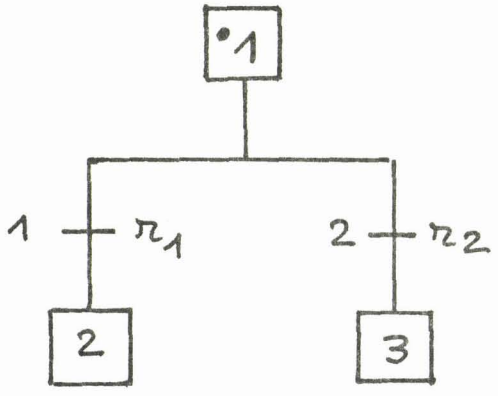


Fig III. 17

Pour un tel noeud OU nous devons distinger quatre cas:

(1) les réceptivités r_1 et r_2 sont logiquement exclusives
($r_1 . r_2 = 0$)

(2) les réceptivités r_1 et r_2 sont technologiquement exclusives

(3) les réceptivités r_1 et r_2 sont identiques

(4) les réceptivités r_1 et r_2 sont quelconques

Le premier cas ne pose pas de problème. Le second exige un complément sur la partie opérative. Pour les troisième et quatrième cas le Grafcet de la figure III.17 est respectivement identique à celui des figures III.18 (a) et (b).

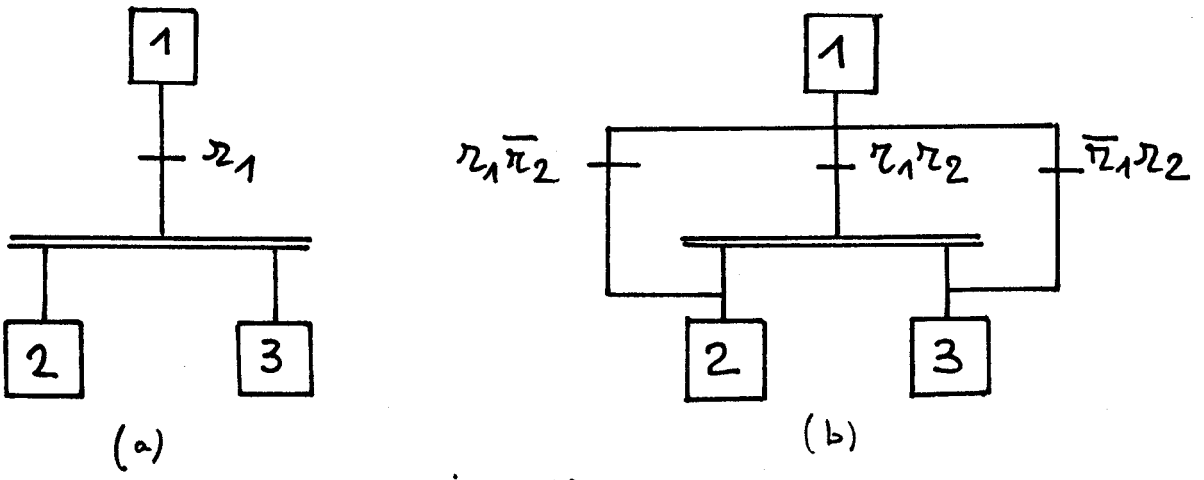


Fig III. 18

Dans ces deux derniers cas, pour adapter la méthode de décomposition le noeud OU doit être remplacé par un noeud ET (Fig III.19)

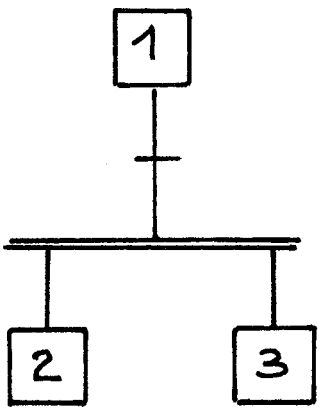


Fig III. 19

La synchronisation pose encore un autre problème, car l'utilisateur a plusieurs solutions pour la réaliser. Ainsi pour l'exemple de la figure III.20 celle-ci peut se faire au niveau des transitions 5 et 6 ou bien de la transition 7. Dans le premier cas les réceptivités des transitions 5 et 6 sont respectivement de la forme

$$r_5 = m_5 \cdot f$$

$$r_6 = m_6 \cdot g$$

et la décomposition en graphes d'état est définie par la partition:

$$\{\{1,2,4,6,7\}\{3\ 5\}\}$$

Dans le second cas la réceptivité de la transition 7 est de la forme:

$$r_7 = \overline{m_2} \cdot \overline{m_3} \cdot \overline{m_4} \cdot \overline{m_5} \cdot h$$

et la décomposition est définie par la partition:

$$\{\{1,2,4,7\},\{3,5\},\{6\}\}$$

Remarquons qu'en l'absence de synchronisation la décomposition doit être donnée par la partition

$$\{\{1\},\{2,4\},\{3,5\},\{6\},\{7\}\}$$

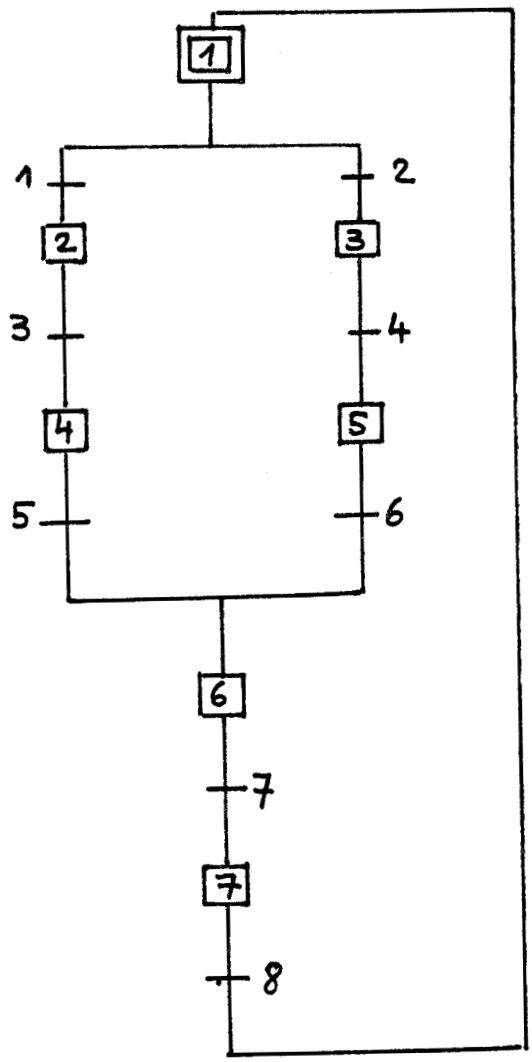


Fig III. 20

Un autre problème est illustré par le grafcet de la figure III-21 qui représente un cycle d'opération qui évolue à chaque apparition d'un front montant d'une commande a. Le cycle comporte successivement les opérations OP_1, puis OP_1 et OP_2, puis OP_2 et OP_3 et enfin OP_3.

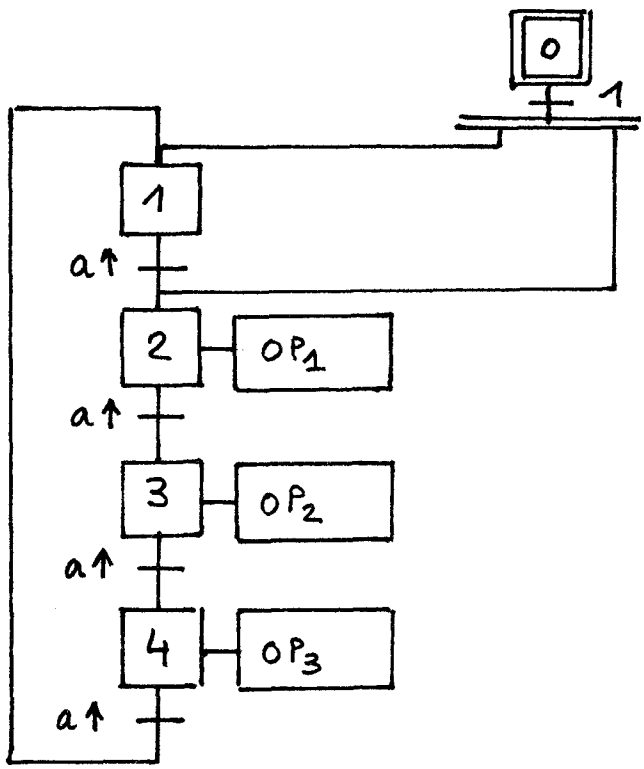


Fig. III-21

Dans ce cas la décomposition est définie par la partition

$$\{ \{1\}, \{2\}, \{3\}, \{4\} \}$$

Dans ce chapitre, nous avons présenté une méthode de décomposition en graphes d'état. Cette dernière est réalisée automatiquement pour les Grafcet de type I . Par contre pour les Grafcet de type II elle doit tenir compte de l'interprétation et de la description de la partie opérative. Ces compléments doivent être fournis préalablement ou sous forme interactive au cours de la description.

Le chapitre suivant présente les diverses méthodes de réalisation d'un Grafcet compte tenu des possibilités offertes par les sytèmes programmés.

CHAPITRE IV

METHODOLOGIE D'IMPLANTATION D'UN GRAFCET SUR SYSTEMES PROGRAMMES

Dans ce chapitre nous présentons les diverses méthodes de réalisation d'un Grafcet et exposons les problèmes liés à à l'usage des différents types de matériel.

IV . 1 REPRESENTATION DES SITUATIONS D'UN GRAFCET

La mise en oeuvre d'un Grafcet sur un système programmé nécessite la possibilité de représenter l'activité de ses étapes. Une première solution consiste à associer à chaque étape une mémoire qui matérialise son activité (cf. I-2 1) ; le nombre de mémoires nécessaires est alors égal au nombre d'étapes. Pour un graphe d'état l'association de p cellules mémoire permet de représenter 2^p étapes ; le nombre de mémoires nécessaires est le plus petit p tel que 2^p soit supérieur ou égal au nombre d'étapes. Dans cette solution le nombre de mémoires utile est inférieur mais pour connaître l'étape active on est obligé de procéder à un décodage. Pour un Grafcet quelconque, on peut comme l'indique le chapitre précédent, procèder à une décomposition en graphes d'états (E_i) i = 1,2,..., n ; la représentation des étapes du sous Grafcet E_i nécessite p_i mémoires où :

$$2^{p_i - 1} < |E_i| \leq 2^{p_i}$$

$|E_i|$ représentant le nombre d'étapes du sous Grafcet.

Le nombre total de mémoires nécessaire est donc:

$$\sum_{i=1}^{m} p_i$$

Sur les machines possédant des instructions de traitement sur mots, on a intérêt à associer à chaque étape, un code généralement sur 8 bits. La liste des codes correspondant aux étapes actives est rangée dans une table ; c'est à dire dans une suite d'emplacements mémoires contigus.

.../...

Précisons les différentes parties du traitement nécessaires pour la mise en oeuvre d'un Grafcet.

IV.2 Différentes parties du Traitement permettant la réalisation d'un Grafcet

Les différentes parties du Traitement d'un Grafcet sont les suivantes :

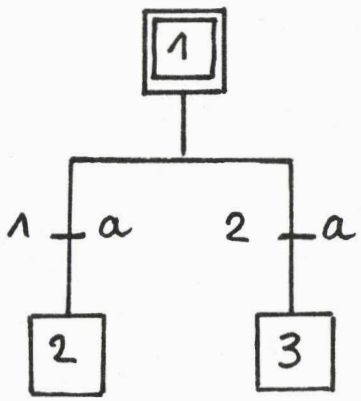
- initialisation
- prise en compte des entrées
- combinatoire général
- détermination de la condition d'évolution à partir des conditions de franchissement
- calcul de la nouvelle situation en cas d'évolution, maintien dans le cas contraire
- combinatoire local
- affectation des sorties

La commande d'initialisation I implicite ou explicite (cf I 2) entraîne l'activation des étapes initialement actives et la désactivation des autres. Suivant la représentation des étapes, le système doit mettre à 1 les mémoires associées aux étapes initialement actives et à 0 les autres ou bien placer les codes des étapes actives dans la table représentative de l'activité du Grafcet.

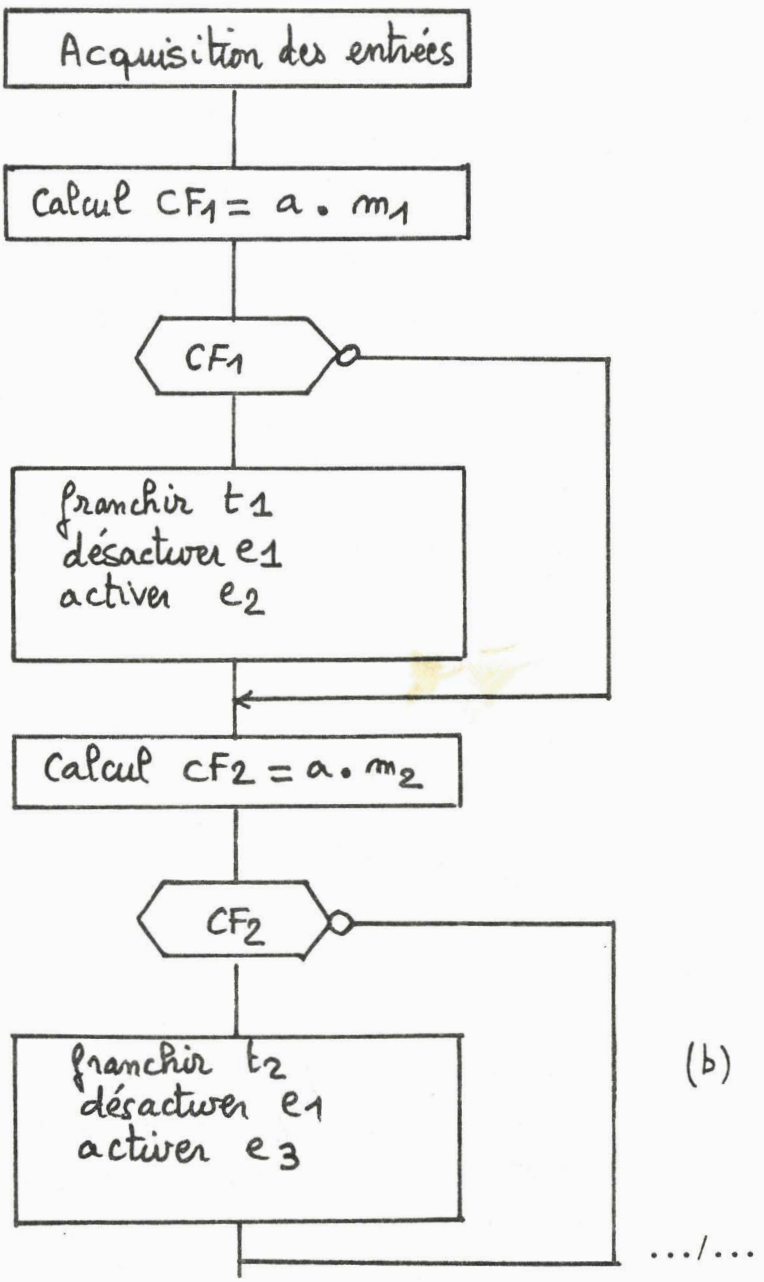
Après l'acquisition des entrées le système doit élaborer le combinatoire général destiné à calculer des fonctions logiques particulières, (du type conditions de sécurité). Il passe ensuite au calcul des conditions de franchissement en fonction des réceptivités et de la situation présente. Ensuite en cas d'évolution il doit procéder à l'établissement de la nouvelle situation, dans le cas contraire il doit maintenir la situation en cours. Puis enfin il doit affecter les valeurs aux sorties compte tenu des étapes actives de sa nouvelle situation. Nous donnons dans le paragraphe la définition d'une réalisation d'un Grafcet à évolution synchrone.

IV. 3 Réalisation d'un Grafcet à évolution synchrone ou à évolution asynchrone.

Une réalisation d'un Grafcet est dite à évolution synchrone si toutes les transitions simultanément franchissables à un instant donné sont simultanément franchies. Une telle réalisation interdit une intéraction entre les conditions de franchissement des différentes transitions, au cours d'un même cycle de traitement. Pour l'exemple de la figure IV 1 (a) l'organigramme de la figure IV 1 (b) correspond à un traitement asynchrone, en effet après le franchissement de la transition 1, le calcul de $CF_2 = m_2$ a donne 0 et la transition 2



(a)



(b)

Fig IV.1

.../...

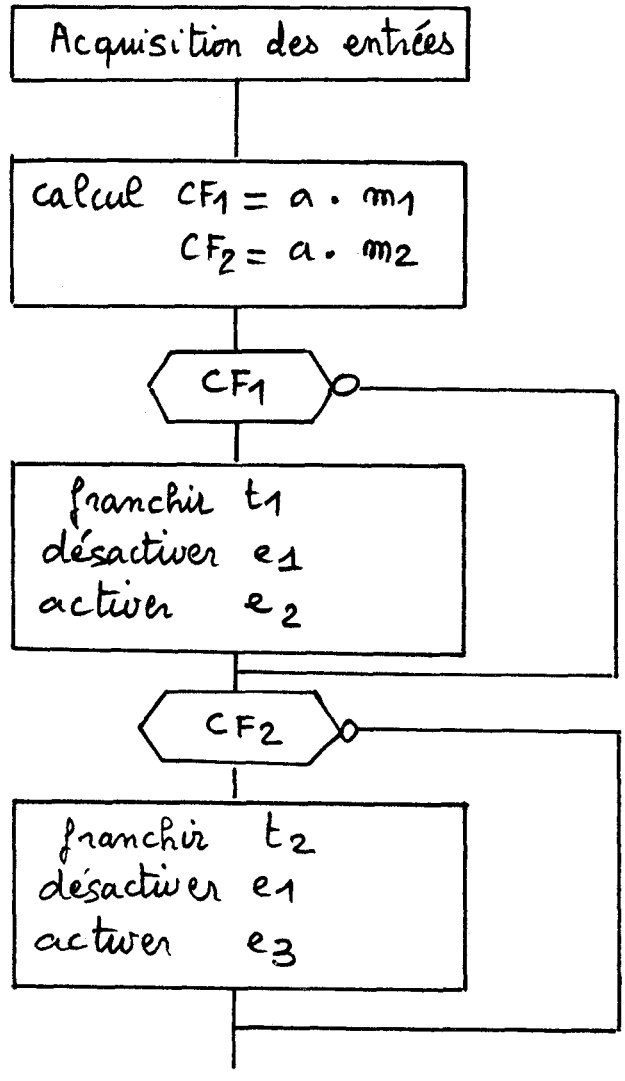


Fig. IV. 1 (C)

ne peut pas être franchie comme elle devrait. L'organigramme de la Fig IV. 1.(C) correspond lui à une réalisation à évolution synchrone.

Une telle réalisation doit se faire en deux étapes :

1) calcul et gel de toutes les conditions de franchissement

2) en cas d'évolution détermination de la nouvelle situation à partir des conditions de franchissement calculées précédemment.

La réalisation d'un Grafcet de type II doit être obligatoirement à évolution synchrone. Remarquons que le gel des conditions de franchissements nécessite une variable interne par transitions, nous verrons dans le paragraphe suivant que le programme correspondant à un traitement synchrone

.../...

est plus long, mais les possibilités offertes par le Grafcet de ce type permettent de présenter la partie commande de façon plus concise, c'est à dire par un graphe qui comporte moins d'étapes et de transitions.

Examinons maintenant les méthodes de réalisation possibles d'un Grafcet en fonction du matériel choisi.

IV.4 METHODES DE REALISATIONS POSSIBLES D'UN GRAFCET SUIVANT LE MATERIEL RETENU (16) (17) (18)

Dans la présentation de ses méthodes nous supposons que le combinatoire général et les conditions de franchissement sont calculées en fonction des entrées.

Rappelons que la condition de franchissement CF_t de la transition est égale au produit logique de sa réceptivité r_t et des variables associées à ses étapes d'entrée :

$$CF_t = r_t \cdot \prod_{e \in \cdot t} m_e$$

Nous nous intéressons, dans un premier temps qu'à la détermination de la nouvelle situation en cas d'évolution et à l'initialisation.

Suivant les possibilités de la machine retenue, nous proposons un classement en trois types de méthodes

1° méthodes d'implantation sur machines qui ne disposent que d'instructions de type combinatoire

2° méthodes d'implantation sur machines qui disposent de mise à 1 et de mise à 0 de mémoires

3° méthodes d'implantation sur machines qui disposent d'instruction de saut général et de traitement sur mots.

IV 4.1 Mise en oeuvre sur machines qui ne disposent que d'instructions combinatoires .

Si la machine ne dispose pas d'instruction de mise à 1 et mise à 0 de mémoires, le concepteur doit alors écrire explicitement les équations des mémoires associées aux étapes.

Pour un Grafcet de type I, ou plus particulièrement qui ne présente pas de réactivation d'étapes, les mémoires associées aux étapes peuvent être indifféremment à la marche (1) ou à arrêt prioritaire (2)

$$m_e = M_e^1 + \overline{M_e^0} \cdot m_e \qquad (1)$$

$$m_e = \left(M_e^1 + m_e\right) \overline{M_e^0} \qquad (2)$$

M_e^1 (resp M_e^0) désignant les conditions d'activation((resp de désactivation) de l'étape e

$$M_e^1 = \sum_{t \in \cdot e} CF_t \qquad M_e^0 = \sum_{t \in e^\cdot} CF_t$$

Mais pour un Grafcet de type II pouvant présenter des réactivations d'étapes, les mémoires doivent être obligatoirement à marche prioritaire (1)

Pour tenir compte de la commande d'initialisation I les équations (1) et (2) deviennent pour les étapes initialement inactives

$$m_e = \left(M_e^1 + \overline{M_e^0} \cdot m_e\right) \overline{I} \qquad (1')$$

$$m_e = \left(M_e^1 + m_e\right) \overline{M_e^0}\ \overline{I} \qquad (2')$$

.../...

et pour les étapes initialement actives

$$m_e = M_e^1 + \overline{M_e^0}\, m_e + I \quad (1'')$$

$$m_e = \left(M_e^1 + m_e\right)\overline{M_e^0} + I \quad (2'')$$

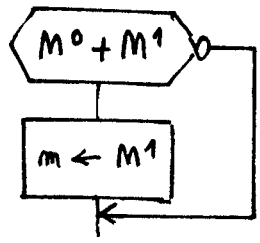
IV. 4.2 Mise en oeuvre sur machines qui disposent d'instructions de réalisation de mémoire sans écrire son équation.

Nous présentons d'abord les différents types d'instructions qui permettent la réalisation d'une mémoire sans écrire son équation. Certaines machines disposent d'instruction de mise à 1 et de mise à 0 conditionnelles comme le PB6. D'autres disposent d'une instruction de blocage des sorties Pour cela l'utilisateur doit préciser leur nombre (cas du 5 TI) ou positionner un indicateur début de zone et fin de zone. (cas du 14500 B). La suite des instructions données par la figure (IV 2) réalise une mémoire à marche prioritaire sur les trois machines citées plus haut.

SI	M^0	si M^0 mise à 0 de la mémoire m
MZ	m	
SI	M^1	si M^1 mise à 1 de la mémoire m
MU	m	

(a)

STR	M^0
OR	M^1
JMP	1
STR	M^1
OUT	m



(b)

LD	M^0
OR	M
OEN	RR
LD	M^1
STO	m

ORC	RR
OEN	RR

(c)

Fig IV. 2

Sur ce type de machines nous pouvons effectuer une mise en oeuvre autour des étapes ou autour des transitions.

IV 4.2.1 Mise en oeuvre autour des étapes

Il suffit alors de réaliser les mémoires associées aux étapes, compte-tenu de la condition d'initialisation. La figure IV. 3 donne les programmes pour les trois types de matériel cités.

Pour une étape e $\notin S_0$

SI	M^0_e
SI	I
OU	VI
SI	VI
MZ	m_e
SI	M^1_e
MU	m_e

Pour une étape e $\in S_0$

SI	M^0_e
MZ	m_e
SI	M^1_e
SI	I
OU	VI
SI	VI
MU	m_e

(a)

STR	M^0_e
OR	M^1_e
OR	I
JMP	1
STR	M^1_e
OUT	m_e

STR	M^0_e
OR	M^1_e
OR	I
JMP	1
STR	M^1_e
OR	I
OUT	m_e

(b)

.../...

LD	M^0_e		LD	M^0_e
OR	M^1_e		OR	M^1_e
OR	I		OR	I
OEN	RR		OEN	RR
LD	M^1_e		LD	M^1_e
STO	m_e		OR	I
		(c)	STO	m_e

Fig. IV-3

IV.4.4.2 Mise en oeuvre autour des transitions

Dans la mise en oeuvre autour des transitions la condition de franchissement de chaque transition conditionne la désactivation de ses étapes d'entrée et l'activation de ses étapes de sortie. L'organigramme de cette méthode est donné par le Fig. IV. 4. L'initialisation est obtenue en considérant une transition source fictive qui force à 1 les étapes initialement actives, et à 0 les autres. Le traitement de l'initialisation doit être prioritaire.

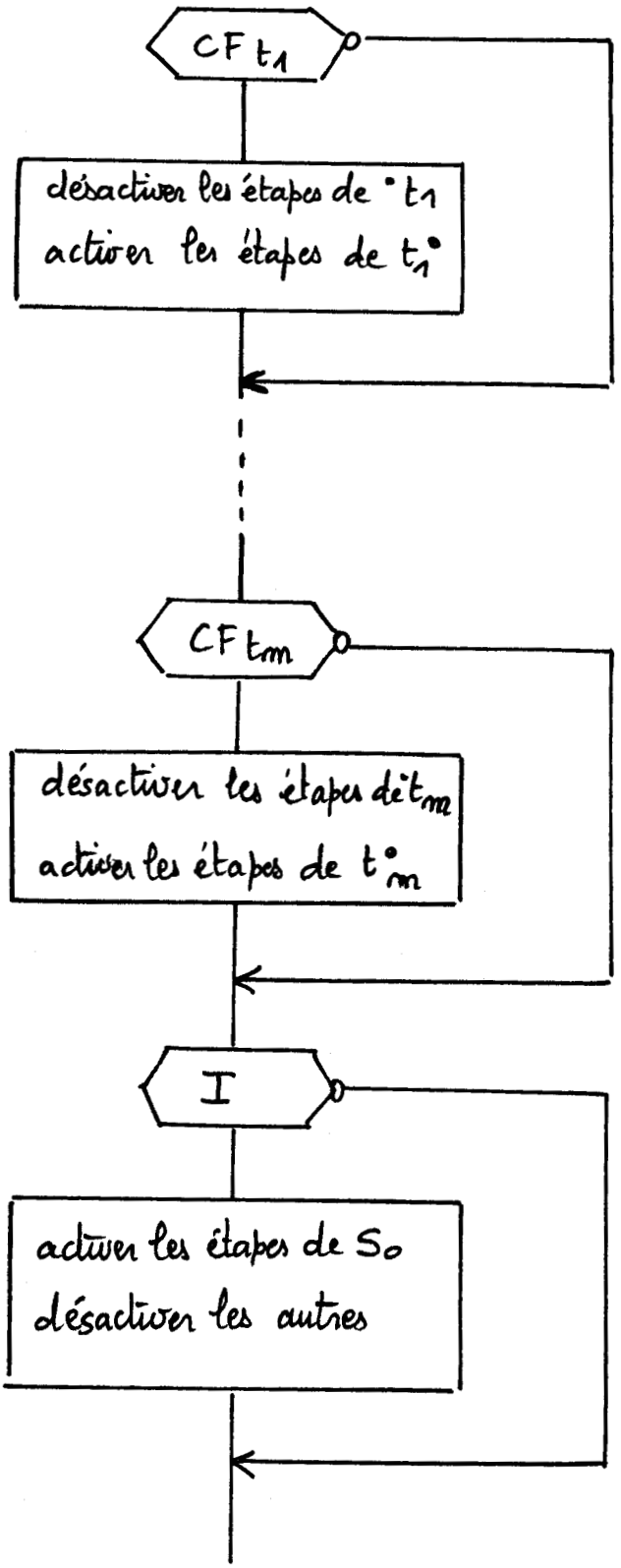

CF t_1
désactiver les étapes de $^\bullet t_1$
activer les étapes de $t_1^\bullet$
CF t_m
désactiver les étapes de $^\bullet t_m$
activer les étapes de $t_m^\bullet$
I
activer les étapes de S_0
désactiver les autres

Fig. IV . 4

.../...

La figure IV.5 donne la suite des instructions correspondant à une transition t dont les étapes d'entrée sont : $e_1, \ldots, e_\alpha$ et les étapes de sortie : $e'_1, \ldots, e'_\beta$ pour les trois types de matériel.

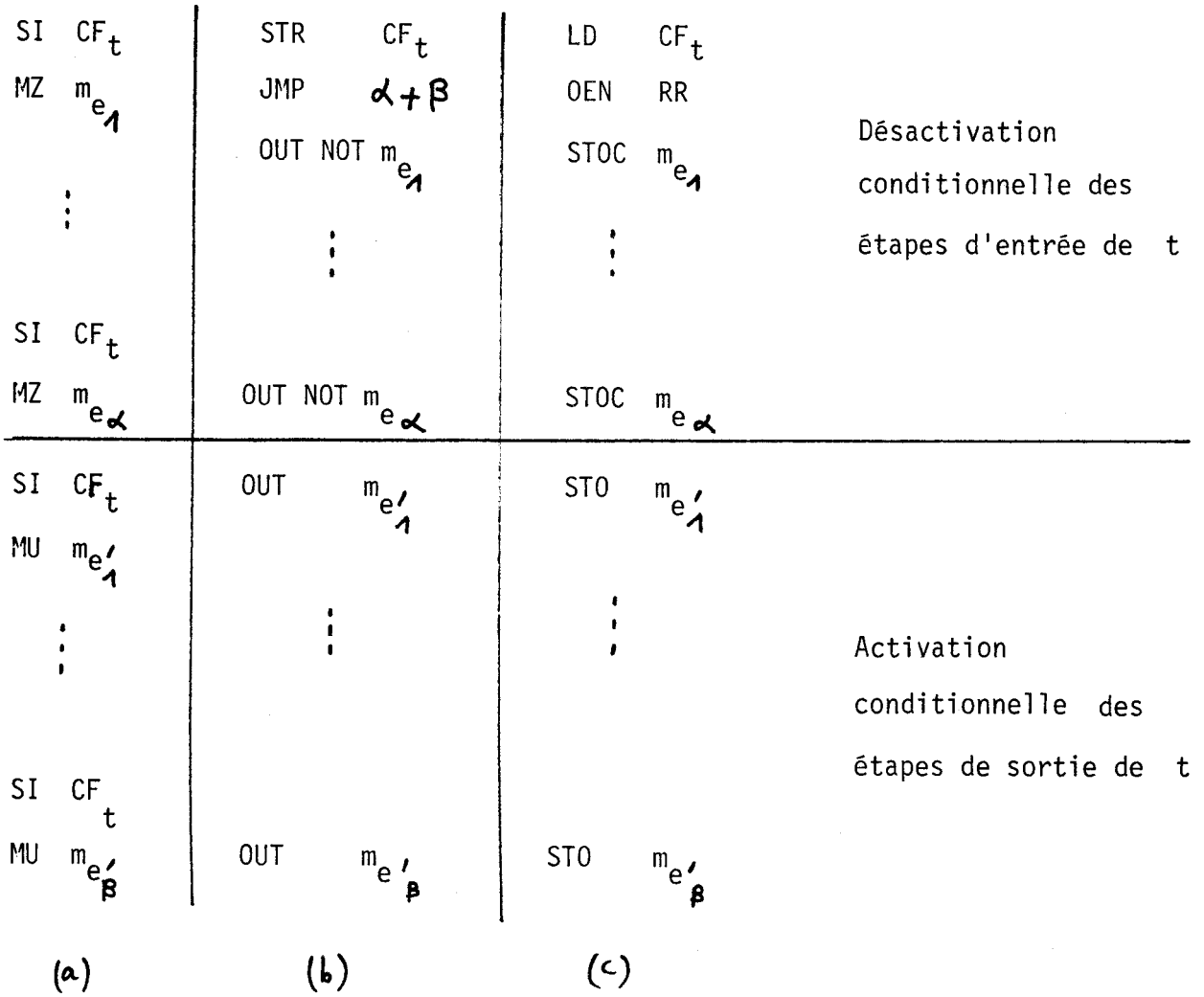


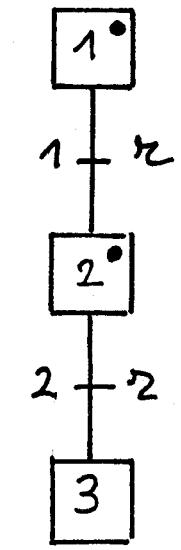
Fig IV.5

Sur la gamme PB de Merlin Gérin l'utilisation de l'instruction de saut permet d'éviter les tests successifs de la condition de franchissement (Fig IV.6)

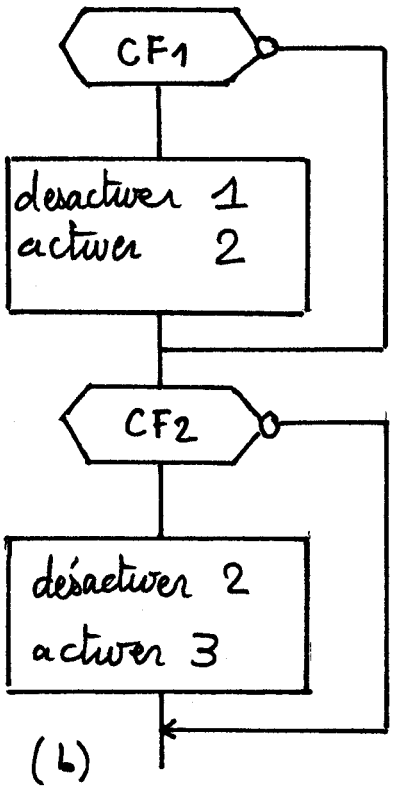
```
        SI /   CF t
        SAUT   α
        MZ     m e1
               :
        MZ     m eα
        MU     m e'1
               :
        MU     m e'β
α :
```

Fig IV.6

Remarquons que cette méthode de réalisation ne convient pas pour les Grafcets qui présentent des réactivations d'étapes. Ainsi pour le Grafcet de la figure IV.7 (a) la vérification de la réceptivité r doit conduire à la situation (2,3). Or la réalisation donnée par l'organigramme de la figure IV.6 (b) conduit à une situation où l'étape 2 est inactive, car la désactivation se fait après l'activation

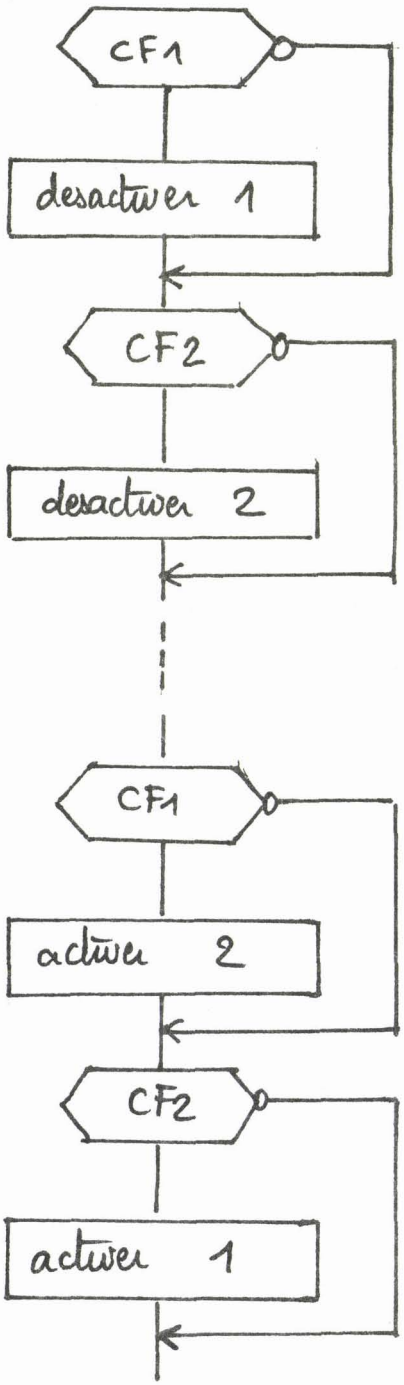


(a)



(b)

Fig IV. 7


CF1
desactiver 1
CF2
desactiver 2
CF1
activer 2
CF2
activer 1

(c)

Fig. IV-7

Pour assurer la priorité à l'activation, nous devons d'abord désactiver toutes les étapes d'entrée des transitions franchies et ensuite activer toutes les étapes de sortie. La figure IV.7 (c) donne l'organigramme correspondant au Grafcet précédent. L'organigramme général est donné par la figure IV.8 .

IV 4 .3 Mise en oeuvre sur machines qui disposent d'instructions de saut général et de traitement sur mots.

Pour ce type de machines les méthodes précédentes sont encore applicables. L'instruction de saut permet en plus d'éviter de calculer les réceptivités des transitions non validées.

Les méthodes de mise en oeuvre peuvent se répartir en deux grandes classes : celles orientées "programme" et celles orientées "données". Dans le premier cas à chaque Grafcet correspond un programme spécifique et dans ce sens toutes les méthodes précédentes appartiennent à cette classe.

Pour une structure orientée "données", à un Grafcet correspond une table de données gérée par un programme, appelé moniteur, constant et indépendant du Grafcet . La situation du Grafcet est matérialisée par la listes des étapes actives ou des transitions validées; dans le premier cas le traitement est dit réalisé autour des étapes, dans le second autour des transitions.

Nous nous limitons ici à un rappel succinct de la méthode orientée "données" autour des étapes. Pour chaque étape de la liste, la table fournit les transitions validées. Le calcul des réceptivités correspondantes peut nécessiter une scrutation de la liste des étapes actives lorsque une transition en possède plusieurs. Lorsqu' une transition est franchissable, la table fournit ses étapes d'entrée et de sortie, ces

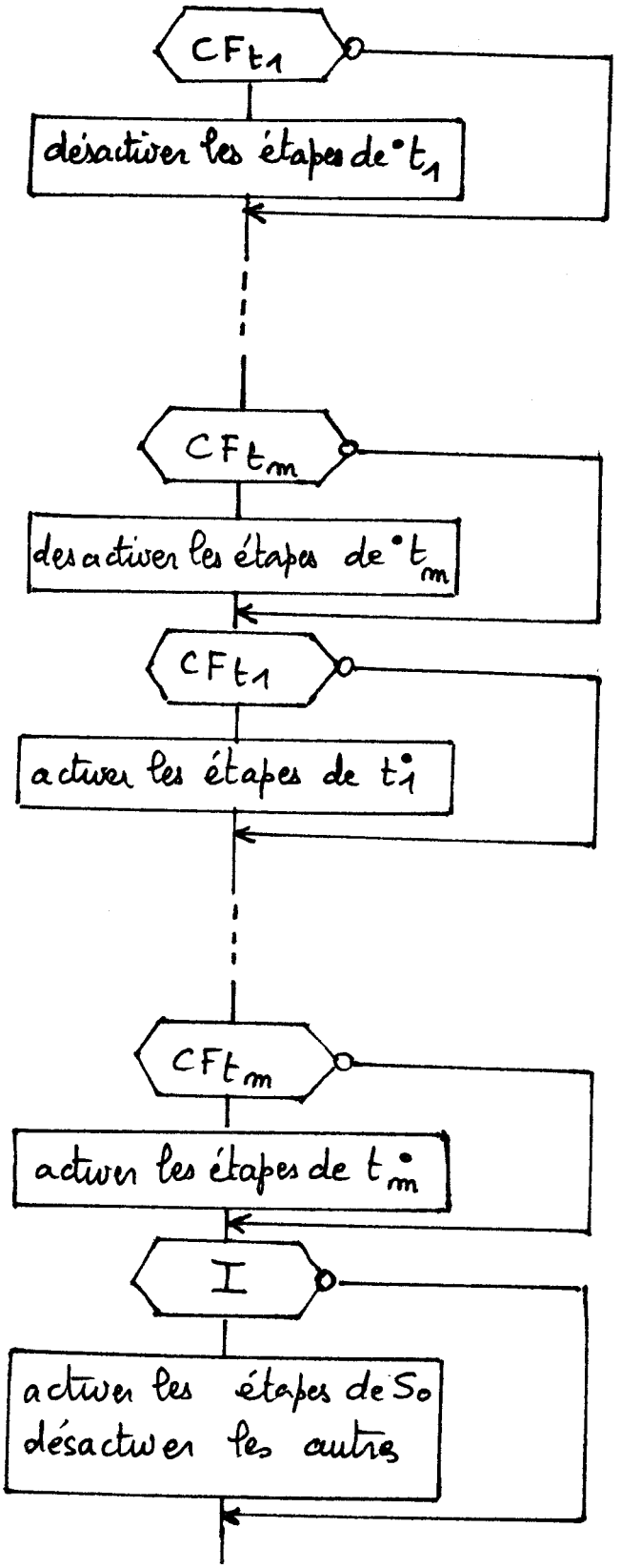

CF t1
désactiver les étapes de °t1
CF tm
desactiver les étapes de °tm
CF t1
activer les étapes de t1
CF tm
activer les étapes de tm
I
activer les étapes de So
désactiver les autres

Fig IV.8

dernières sont recopiées dans les listes des étapes à activer et à désactiver. Après avoir traité toutes les étapes de la liste des étapes actives, avec le calcul des sorties associées, le programme moniteur doit à partir des trois listes définir la liste des étapes actives en assurant la priorité à l'activation sur la désactivation.

Nous précisons dans le paragraphe suivant la partie du traitement qui concerne l'établissement des sorties ou combinatoire local.

IV 5 COMBINATOIRE LOCAL

Nous supposons que l'affectation des sorties est ou est rendue synchrone, c'est à dire que toutes les sorties sont affectées en même temps.

Rappelons que l'interprétation du Grafcet associe à chaque étape les ensembles de sortie suivants:

A_e l'ensemble des sorties continues (éventuellement conditionnelles) contrôlées par l'activité de l'étape e .

A_e^1 (resp A_e^0) l'ensemble des sorties à lancer (resp à arrêter) (éventuellement sous condition) dès l'activation de l'étape e

A_e^* l'ensemble des sorties impulsionnelles ou ponctuelles (éventuellement conditionnelles) commandées par l'étape e

nous noterons:

$$\mathcal{A} = \bigcup_{e \in E} A_e \quad , \quad \mathcal{A}^1 = \bigcup_{e \in E} A_e^1 \quad , \quad \mathcal{A}^0 = \bigcup_{e \in E} A_e^0 \quad , \quad \mathcal{A}^* = \bigcup_{e \in E} A_e^*$$

et nous présentons les traitements correspondants a chacun de ces types de sorties.

IV 5.1 Sorties de l'ensemble $\mathcal{A}$

Pour chaque sortie a de $\mathcal{A}$, c'est à dire contrôléee par l'activité d'une ou plusieurs étapes, le système doit matérialiser l'équation combinatoire suivante :

$$a = \sum_{\{e \in E \mid a \in A_e\}} c(a,e) \cdot m_e$$

où $c(a,e)$ représente la condition éventuelle de la sortie a associée à e

IV 5.2 Sorties de l'ensemble $\mathcal{A}^1$ (ou $\mathcal{A}^0$)

Par contre les sories de $\mathcal{A}^1$ (ou $\mathcal{A}^0$) sont lancées (ou arrêtées) par l'activation d'une ou plusieurs étapes; elles ne sont plus ensuite contrôlées par l'activité de ces dernières et ne peuvent donc s'exprimer en fonction des variables associées. Ces sorties doivent être gérées comme des mémoires. Pour une sortie a lancée et arrêtée toujours sans condition quel que soit l'étape (c'est à dire $c(a,e) = 1$ pour tout e),les conditions de mise à 1 et de mise à 0 de la mémoire associée sont respectivement

$$M_a^1 = \sum_{\{e \in E \mid a \in A_e^1\}} m_e \uparrow = \sum_{\{e \in E \mid a \in A_e^1\}} \overline{m_e} \cdot M_e^1$$

$$M_a^0 = \sum_{\{e \in E \mid a \in A_e^0\}} m_e \uparrow = \sum_{\{e \in E \mid a \in A_e^0\}} \overline{m_e} \cdot M_e^1$$

Pour une sortie a lancée (resp arrêtéé) conditionnellement par l'activité d'une étape e , deux interprétations sont possibles. Dans la première à l'activation de l'étape e , si la condition est vraie la sortie est lancée (resp arrêtée) , si la condition est fausse elle ne sera jamais lancée (resp arrêtée) . Dans la seconde la sortie a est lancée (resp arrêtée) sur le premier front du produit logique de la variable d'étape m_e et de la condition c(a,e). La figure IV.9 précise ces deux interprétations; pour la première les conditions de mise à 1 et de mise à 0 de la mémoire associée sont respectivement:

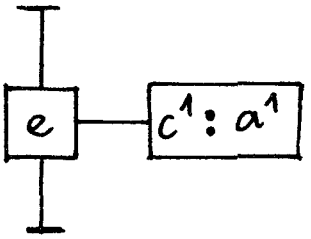
$$M_a^1 = \sum_{\{e \in E \mid a \in A_e^1\}} m_e \uparrow . \; c(a,e) = \sum_{\{e \in E \mid a \in A_e^1\}} \overline{m_e} \cdot M_e^1 \cdot c(a,e)$$

$$M_a^0 = \sum_{\{e \in E \mid a \in A_e^0\}} m_e \uparrow . \; c(a,e) = \sum_{\{e \in E \mid a \in A_e^0\}} \overline{m_e} \cdot M_e^1 \cdot c(a,e)$$

dans la seconde, elles sont respectivement:

$$M_a^1 = \sum_{\{e \in E \mid a \in A_e^1\}} [m_e \cdot c(a,e)] \uparrow$$

$$M_a^0 = \sum_{\{e \in E \mid a \in A_e^0\}} [m_e \cdot c(a,e)] \uparrow$$



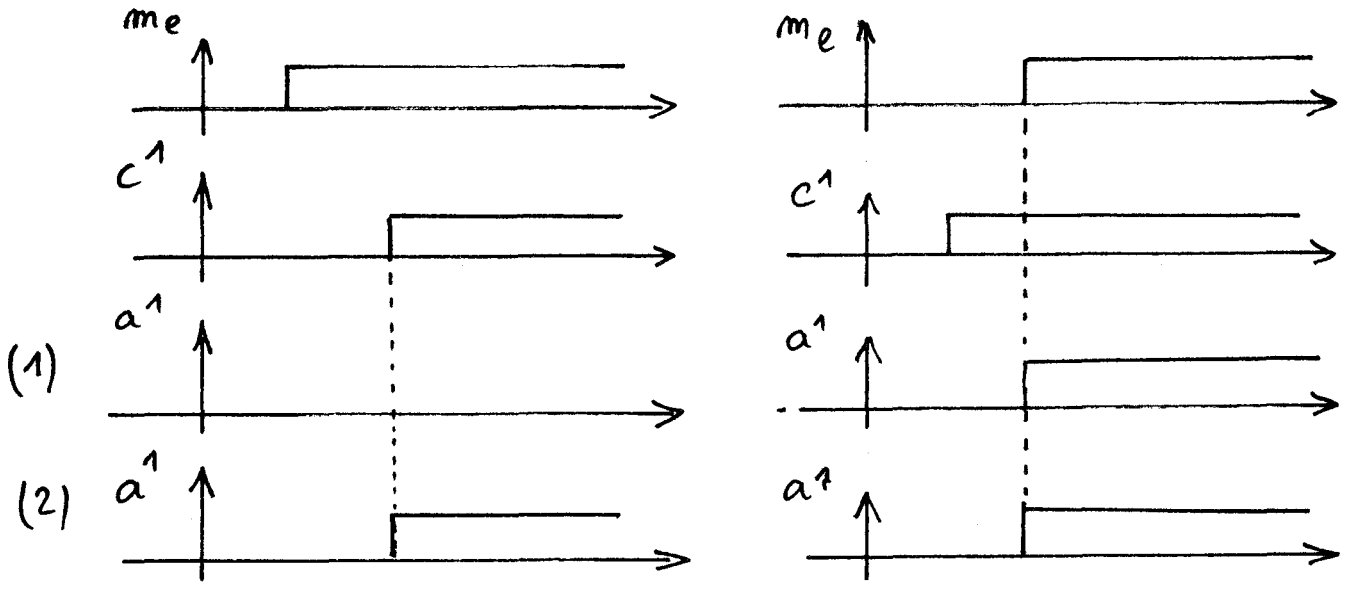


Fig IV.9 Etat de la sortie a^1 lancée par l'activation de e avec la condition c^1 suivant les 2 interprétations

Une sortie a lancée par l'activation d'une étape avec une condition éventuelle n'est ensuite plus contrôlée par l'activité de cette étape. Cette sortie ne peut être arrêtée que par l'activation d'une étape e dont l'ensemble A_e^0 contient a . Pour permettre un contrôle constant d'une telle sortie a et ainsi la soumettre à tout moment à un arrêt d'urgence A , le concepteur doit ajouter une étape e' telle que: $A_{e'} = \{a\}$ $\quad c(a, e') = A$

Cette étape e' ne possède qu'une transition d'entrée et une transition de sortie dont les réceptivtités sont respectivement M^1_a et M^0_a les conditions de mise à 1 et à 0 de la mémoire associée à la sortie a (Fig IV. 10)

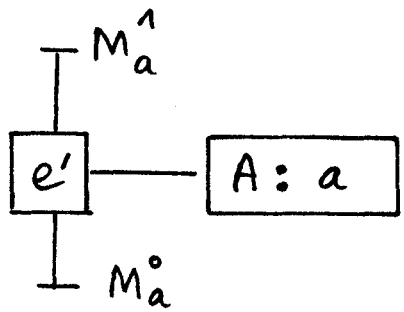


Fig IV.10

Remarquons que l'initialisation doit comprendre une commande des des sorties à lancer ou à arrêter associées aux étape de la situation initiale S_0 .

IV.5.3 Sorties de l'ensemble $\mathcal{A}^*$

Pour obtenir une sortie impulsionnelle $a \in \mathcal{A}^*$ commandée par l'étape e, de durée d explicite, nous utilisons une temporisation dont l'ordre de lancement est:

$$\left[m_e \cdot c(a,e) \right] \uparrow$$

Cette sortie impulsionnelle est alors égale à :

$$m_e \cdot \overline{FT}_{d,e}$$

où $FT_{d,e}$ représente le signal fin de temporisation (Fig. IV-11)

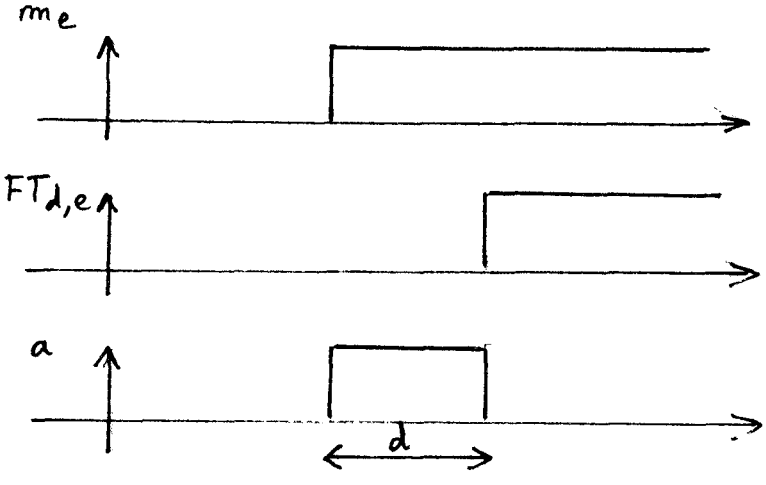


Fig. IV.11 - Commande de la sortie impulsionnelle a^* de durée d appartenant à A^*_e

Cette solution nécessite une temporisation pour chaque sortie A^*_e; le nombre total de temporisations nécessaires est alors :

$$\sum_{e \in E} |A^*_e|$$

où $|A^*_e|$ représente le nombre de sorties de A^*_e. Nous pouvons regrouper les temporisations de même durée, lancées par ces étapes non simultanément actives.

Dans le cas où la durée n'est pas précisée, la sortie a doit être gérée en variable monocoup, pour cela, il est nécessaire d'utiliser une variable intermédiaire a' et de réaliser les deux équations logiques :

$$a = \left[m_e \cdot c\ (a,e) \right] \uparrow \cdot \bar{a}' \qquad (1)$$

$$a' = a \qquad (2)$$

Pour chaque sortie impulsionnelle de durée non explicitée, nous devons réaliser les deux équations

$$a = \bar{a}' \sum_{a \in \mathcal{A}^* \text{ et d non explicite}} m_e \cdot c\ (a,e)$$

$$a' = a$$

Après avoir défini la méthodologie d'implantation d'un grafcet nous présentons dans le chapitre suivant le module traducteur.

CHAPITRE V

MODULE D'IMPLANTATION DU GRAFCET

Après avoir rappelé les différentes méthodes de réalisations d'un Grafcet sur systèmes programmés, nous donnons la structure du module TRADUCTEUR qui réalise l'implantation automatique.

V. I STRUCTURE GENERALE DU MODULE TRADUCTEUR

Suivant le type du materiel choisi, le système offre à l'utilisateur les choix suivants:

1° - si le matériel retenu ne possède pas des instructions de type combinatoire, une seule méthode appelée activation désactivation est possible.

2° - si le matériel retenu possède des instructions de mises à 1 et 0 conditionnelles, trois méthodes d'implantation sont alors possibles.Le concepteur peut, en effet, choisir une méthode de réalisation par rapport aux étapes, ou par rapport aux transitions ou encore par rapport aux blocs. Remarquons que la dernière exige une décomposition préalable du Grafcet en graphes d'état. Ces derniers appelés blocs peuvent être définis lors de l'introduction du Grafcet ou déterminés automatiquement par le module décomposition.

3° - si le matériel retenu possède des instructions de saut général ou de traitement sur mots, l'utilisateur a alors le choix entre deux grandes classes de méthode.

L'une est constituée des méthodes orientées "programme", l'autre des méthodes orientées "données". La première classe offre les méthodes citées au point (2), l'instruction de saut permet en plus de réduire le temps de traitement en évitant à la machine de

calculer les réceptivités des transitions non validées et les sorties non conditionnelles en cas de non évolution, nous verrons que ces possibilités entrainent une augmentation de la taille du programme.

Pour les méthodes orientées données, la structure des tables peut être définie à partir des étapes ou des transitions. Remarquons que la complexité de la structure de ces tables et à fortiori celle de leurs moniteurs de gestion dépendent du type du Grafcet : les plus simples sont obtenues pour un graphe d'état, les plus compliquées pour un Grafcet de type II. Une décomposition associée à un moniteur gestion de multi graphes d'état permet de pallier ces difficultés.

Toutes ces méthodes peuvent en plus conduire à une réalisation de caractère asynchrone ou synchrone ; nous rappelons que les Grafcets de type II exigent une réalisation synchrone.

Pour les méthodes orientées données, le programme moniteur indépendant du Grafcet est fixe, la détermination de la table à partir du Grafcet ne pose aucun problème. Pour les autres une traduction immédiate dans le langage de la machine choisie exige un traducteur spécialisé par méthode et par type de matériel. Or pour une même méthode, tous ces traducteurs spécialisés possèdent une partie commune qui comprend au minimum l'établissement des directives de traduction. Ces dernières ne dépendent que de la méthode d'implantation retenue. Nous montrons dans les paragraphes suivants que ces dernières peuvent être exprimées dans un langage intermédiaire tel que les traducteurs spécialisés se réduisent à des simples programmes de transcriptions.

V. II ETABLISSEMENT DE DIRECTIVES DE TRADUCTION (17) (18)

Nous présentons ses directives de traduction pour les différentes méthodes d'implantation.

V. II. 1 Directives de traduction pour les machines qui ne disposent que d'instructions de type combinatoire.

Pour la méthode activation - désactivation

..../...

les directives de traduction sont constituées d'une suite d'équations combinatoires du type

expression = variable

Dans un premier temps, les expressions logiques sont écrites sous la forme habituelle, nous verrons qu'elles peuvent s'exprimer dans un langage intermédiaire qui permettra une transcription aisée dans le langage de n'importe quelle machine.

Nous pouvons distinguer trois parties :

- combinatoire général
- équations des mémoires associées aux étapes et aux sorties à lancer ou à arrêter
- calcul des sorties.

l'établissement des équations correspondants au combinatoire général ne pose aucun problème, car elles font partie de la description du Grafcet. Les équations des mémoires associées aux étapes seront pour les étapes initialement inactives de la forme.

$$(M_e^1 + \overline{M_e^o} \cdot m_e^*)\ \overline{I} = m_e^*$$

pour les étapes initialement actives :

$$M_e^1 + \overline{M_e^o} \cdot m_e^* + I = m_e^*$$

où M_e^1 et M_e^o représentent les conditions d'activation et de désactivation de l'étape e, leurs expressions respectives sont :

$$M_e^1 = \sum_{t \in \cdot e} CF_t = \sum_{t \in \cdot e} r_t \left(\prod_{e \in \cdot t} m_e^* \right)$$

$$M_e^0 = \sum_{t \in e^\cdot} CF_t = \sum_{t \in e^\cdot} r_t \left(\prod_{e \in \cdot t} m_e^* \right)$$

I est la commande d'initialisation.

- m_e^* est le nom de la variable qui matérialise l'activité de l'étape e, ce dernier peut être défini par l'utilisateur ou à défaut le symbole % E e

Dans le premier cas, l'adresse de cette variable devra être donnée par l'utilisateur lors de l'introduction des tables, dans le second cas, elle sera déterminée par le traducteur spécialisé. Les équations des mémoires associées aux sorties à lancer ou à arrêter sont du même type.

Le calcul des sorties contrôlées par l'activité d'une étape est constituée d'une suite d'équations de la forme

$$\sum_{\{ e \mid a \in A_e \}} m_e^* \cdot c(a,e) = a$$

Pour chaque sortie impulsionnelle a^* de durée d précisée associée à une étape e, nous aurons l'ordre de lancement de la temporisation $T_{d',e}$ associée

$$\left[m_e \cdot c(a,e) \right] \uparrow$$

et l'équation

$$m_e \cdot \overline{FT}_{d,e} = a^*$$

Une sortie impulsionnelle dont la durée n'est pas précisée conduit au couple d'équations :

$$a' \quad \sum \left[m_e^* \; c\,(a,e) \right] \uparrow = a$$

$a \in A^*$ et d non précisée

$$a = a'$$

La version synchrone de cette méthode exige en plus l'établissement préalable des conditions de franchissement

$$r_t \left(\prod_{e \in {}^\bullet t} m_e^* \right) = \% T_t$$

$\% T_t$ repère la variable qui donne la valeur de la condition de franchissement de la transition t

V. II . 2 Directives de traduction pour les machines qui disposent d'instructions de mise à 1 et à 0 conditionnelles :

Nous nous limitons à la présentation de la version synchrone de ces méthodes.

V.II. 2.1 Directives de traduction pour la méthode par rapport aux étapes

La méthode de réalisation par rapport aux étapes conduit aux directives de traduction suivantes :

1° Calcul des conditions de franchissement

Pour toute transition t : $r_t \left(\prod_{e \in {}^\bullet t} m_e^* \right) = \% T t$

2° Etablissement de l'activité du Grafcet et lancement ou arrêt des sorties de $\mathcal{A}' \cup \mathcal{A}^\circ$

Pour chaque étape e initialement inactive :

S I	M_e° + I	M Z	m_e^*
S I	M_e^1	M U	m_e^*
		LANCER	$A_e^{1'}$
		ARRETER	$A_e^{\circ'}$

Pour chaque étape e initialement active

S I	M_e°	M Z	m_e^*
S I	M_e^1 + I	M $\cup$	m_e^*
		LANCER	$A_e^{1'}$
		ARRETER	$A_e^{\circ'}$

où $A_e^{1'}$ et $A_e^{\circ'}$ représentent les sorties lancées ou arrêtées inconditionnellement par l'activation de e ; pour les autres nous aurons

S I	M_a°	M Z	a
S I	M_a^1	M U	a

.../...

3° Le calcul des sorties contrôlées par les étapes et des sorties impulsionnelles est identique à celui du paragraphe V.II. 1

V.II. 2.2 Directives de traduction pour la méthode par rapport aux transitions :

La méthode de réalisation par rapport aux transitions conduit aux directives suivantes :

1° Calcul des conditions de franchissement

2° Désactivation des étapes d'entrée des transitions franchies

Pour toute transition t :

S I % T t M Z $\{ m_e^* \mid e \in t^\bullet \}$

3° Activation des étapes de sorties des transitions franchies et lancement ou arrêt des sorties de

$\bigcup_{e \in t^\bullet} A_e^{1'}$ et $\bigcup_{e \in t^\bullet} A_e^{0'}$

S I % T t M U $\{ m_e^* \mid e \in t^\bullet \}$

LANCER $\bigcup_{e \in t^\bullet} A_e^{1'}$

ARRETER $\bigcup_{e \in t^\bullet} A_e^{0'}$

Pour les autres sorties à lancer ou à arrêter conditionnellement, nous aurons :

S I M_a^0 M Z a

S I M_a^1 M U a

4° Initialisation

S I I M U S_o

M Z $\subset_E$ S_o

5° Le calcul des sorites contrôlées par les étapes et des sorties impulsionnelles est également identique à celui du paragraphe V.II. 1

V.II. 2.3 Directives de traduction pour la méthode par rapport aux blocs

Dans la méthode de réalisation par rapport aux blocs les points (2) à (4) sont remplacés par :

(2') calcul des conditions d'évolution des blocs

Pour chaque bloc B_i

$$\sum_{t \in T_i} \% T t + I = \% B_i$$

3° Calcul de l'activité des étapes de chacun des blocs

S I % B_i

pour chaque étape e du bloc B_i nous aurons :

.../...

si e n'est pas initialement active

$$\sum_{t \in {}^{\bullet}e} \% \; T \; t \quad = \quad m_e^*$$

si e est initialement active

$$\sum_{t \in {}^{\bullet}e} \% \; T \; t \; + I = \quad m_e^*$$

4° lancement ou arrêt des sorties de $A^1 \cup A^\circ$

Pour chaque sortie de $A^1 \cup A^\circ$ nous aurons :

S I	M_a°	M Z	a
S I	M_a^1	M U	a

Ces directives de traduction étant établies, il ne reste plus qu'à exprimer les équations combinatoires dans un langage intermédiaire qui devra être suffisamment puissant pour représenter ces expressions, mais aussi assez proches des langages des diverses machines pour permettre une transcription aisée. Pour définir ce langage nous commençons par étudier les différents types de primitives logiques des automates programmables.

V. III DIFFERENTS TYPES DE PRIMITIVES LOGIQUES DES AUTOMATES PROGRAMMABLES (9) (10) (20)

Nous pouvons distinguer 6 types de primitives logiques :

(1) modules logiques prédéfinis

(2) jeu d'instruction d'un calculateur portant un bit (appel, et, ou, rangement ...)

..../...

(3) écriture de l'équation booléenne élément par élément (analyseur booléen)

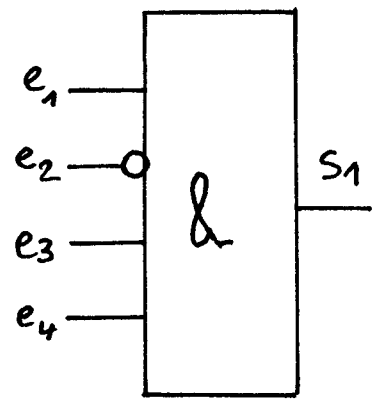
(4) notation polonaise inverse

(5) diagramme à relais

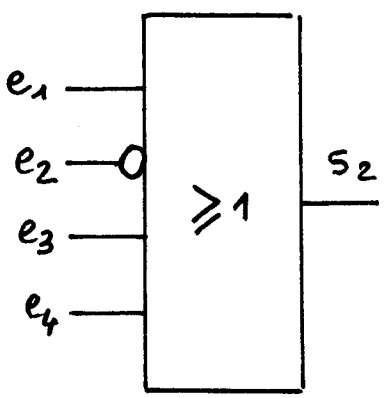
(6) organigramme avec saut conditionnel

V.III. 1 Modules logiques prédéfinis

Nous citons le module SPRINT LOGIC de Lepaute Alsthom et la gamme PB (PB - 3 PB - 6 PB - 100) de Merlin gerin. Sur ces types de matériel la réalisation d'un opérateur ET (resp OU) à 4 entrées e_1 e_2 e_3 e_4 de sorties respectives S_1 et S_2 s'écrit :



ET		SI	e_1
e_1		SI /	e_2
PAS			
e_2		SI	e_3
e_3		SI	e_4
e_4		ET	S_1
=			
S_1			



OU		SI	e_1
e_1		SI/	e_2
PAS		SI	e_3
e_2		SI	e_4
e_4		OU	S_2
=			
S_2			

module SPRINT LOGIC gamme PB

.../...

V.III.2 Jeu d'instruction d'un calculateur sur un bit

Nous citons le TEC 16 construit à partir du microprocesseur 1 bit 14500 B de Motorola, et le SD 77/64 de crouzet.

Cette classe de matériel possède le jeu d'instruction classique d'un calculateur portant sur 1 bit, la figure précise les jeux d'instructions

TEC 16		SD 77/64		Signification
LIR	v	LDA	v	$A \leftarrow v$
LIR C	v	LDAC	v	$A \leftarrow \overline{v}$
ET	v	AND	v	$A \leftarrow (A).v$
ETC	v	ANDC	v	$A \leftarrow (A).\overline{v}$
OU	v	OU	v	$A \leftarrow (A)+v$
OUC	v	OUC	v	$A \leftarrow (A)+\overline{v}$
SOR	v	STO	v	$v \leftarrow (A)$
SORC	v	STOC	v	$v \leftarrow \overline{(A)}$

V.III.3 Ecriture de l'équation booléenne élément par élément.

Nous pouvons citer le TSX 80 de Télémécanique et la gamme SMC de Renault. Ce type de matériel permet l'écriture de l'équation logique sous sa forme habituelle ; l'affectation du résultat (opérateur =) se place toute fois en fin d'équation. Le nombre de niveaux de parenthèses est limité à 1 sur la gamme SMC Le TSX 80 offre la possibilité d'ouvrir plusieurs parenthèses, mais avec une fermeture globale unique, ce qui correspond en fait à un seul niveau.

V.III.4 Notation polonaise inverse

Certaines machines possèdent une structure de pile à plusieurs niveaux, ce qui permet l'utilisation des notations polonaise inverse ou post fixée . Dans la première, l'ordre des opérandes est conversé, mais la taille de la pile nécessaire est plus importante. Ainsi l'expression logique

a. (b.c + d . (e.f + g.h) . (i + j))

s'écrit en notation polonaise inverse :

abc . def . gh . + . i j + . + .

et notation postfixée :

bc . ef . gh . + d . ij + . + a.

A titre d'exemple, nous pouvons citer le 5 TI de Texas qui possède une pile à 4 niveaux ; sur ce matériel, la réalisation de la fonction : a . b + $\overline{c}$.d peut se faire sous les deux formes données par le tableau de la fig. V-1

PILE

			Niveau 0	Niveau 1	Niveau 2	Niveau 3
STR	a	a	-	-	-	-
AND	b	a.b	-	-	-	-
STR NOT	c	$\overline{c}$	a.b	-	-	-
AND	d	$\overline{c}$ .d	a.b	-	-	-
OR	STR	a.b+$\overline{c}$.d	-	-	-	-

Fig. V.1.

FILE

			Niveau 0	Niveau 1	Niveau 2	Niveau 3
STR	a	a				
STR	b	b				
AND	STR	a.b				
STR NOT	c	$\overline{c}$				
STR	d	d	$\overline{c}$	a.b		
AND	STR	$\overline{c}$ .d	a.b			
OR	STR	a.b + $\overline{c}$ .d				

Fig. V-1

V.III.5 Diagrammes à relais

Sur ce type de matériel, le schéma à relais est décrit avec les symboles suivants :

La visualisation du schéma peut se faire élément par élément comme sur le Promodul de la Commande Electrique, ou globalement sur un écran comme sur le Director 1001 de Crouzet.

La notion d'ouverture et de fermeture de branches peut varier d'un constructeur à l'autre. Ainsi le schéma de la Fig.V.2 ne peut pas toujours être directement implanté.

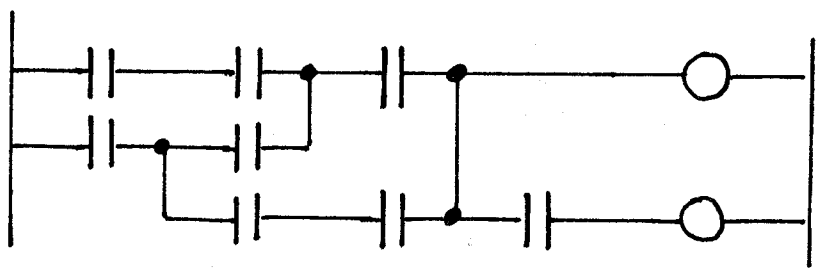


Fig. V.2

V. III. 6 Organigramme avec saut conditionnel :

Sur ce type de matériel l'utilisateur a la possibilité de réaliser une expression logique avec des primitives du type

Si variable ALORS adresse

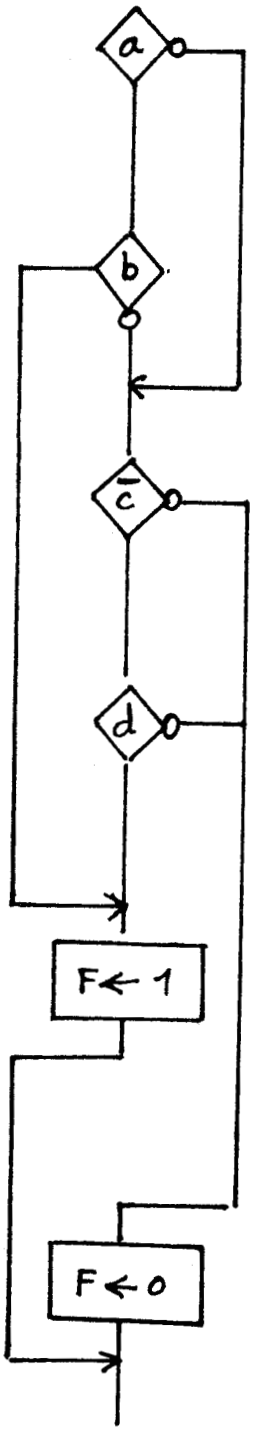
comme sur la gamme PB de Merlin Gérin ou du type

Si variable ALORS adresse SINON adresse

comme sur le TSX 80

Cette possibilité permet de réduire le temps de calcul d'une expérience logique ; en effet, si dans l'évaluation d'un produit, l'un des facteurs est à 0 le produit est nul et on peut passer au terme suivant sans examiner les autres facteurs. De même dès qu'un terme d'une somme est à 1, l'expression est à 1. La réalisation de l'expression

$F = a \cdot b + \overline{c}\, d$ est donnée par la Figure V-3.




```
C40  SI/  a        ! '10'  N a  T @'30'
C41  SAUT A03
C42  SI   b        ! '20'  P b  T @'50'
C43  SAUT A05
C44  SI   c        ! '30'  P c  T @'60'
C45  SAUT A05
C46  SI/  d        ! '40'  N d  T @'60'
C47  SAUT A03
C48  MU   F        ! '50'  S F    @'70'
C49  SAUT A02
C4A  MZ   F        ! '60'  R F
C4B  ....          ! '70'
```

gamme PB

TSX 80

Fig. V-3

Nous pouvons également remarquer que dans cette méthode il n'est pas nécessaire de disposer de variables internes pour ranger des résultats intermédiaires.

Apres avoir rappelé les différents types de primitives logiques des automates programmables, nous pouvons définir le langage intermédiaire dans lequel les expressions logiques devront être prétraduites.

V. IV. Définition du langage intermédiaire de traduction pour les expressions logiques.

Pour les automates reconnaissant les organigrammes (6), nous proposons une prétraduction des expressions logiques dans un langage qui possède les instructions suivantes :

ETIQ : si variable alors ETIQ 1 sinon ETIQ 2
ETIQ : mise à 1 variable aller à ETIQ 1
ETIQ : mise à 0 variable

Pour les autres le langage intermediaire retenu est une suite de triplets (v_1, v_2, 0) suivie d'une affectation ; v_1, v_2 sont des variables de l'expression ou des variables introduites par le compilateur pour mémoriser un résultat intermédiaire et 0 un opérateur. Nous donnons l'algorithme qui permet la traduction d'une expression logique quelconque sous cette forme.

Ce dernier est inspiré de la méthode donnée par Bauer et Samuelson pour les expressions arithmétiques. Nous utilisons deux piles de mémoire, l'une notée P_v pour les noms de variables, l'autre notée P_o pour les opérateurs. Leurs pointeurs respectifs sont désignés par PP_v et PP_o, leurs sommets par P_v (PP_v) et P_o (PP_o). Il est nécessaire de définir des variables intermédiaires, ces dernières

.../...

notées sous la forme % i, sont destinées à recevoir un resultat partiel ; le compteur qui contient le numéro de la dernière variable intermédiaire crée est notéé CV.

La prétraduction de l'expression logique s'effectue en un seul balayage de la chaine de caractères correspondante. Pour chaque opérateur ρ ainsi rencontré, on place d'abord dans la pile P_V le nom $_V$ de la variable qui le précède, puis on le compare au sommet de la pile opérateur, la matrice de décision donnée par la figure V.4 précise alors les diverses possibilités pour l'opérateur O.

a) empiler l'operateur rencontré dans la pile P_O

b) passer à l'opérateur suivant de l'expression

opérateur examiné ρ \ $P_o(PP_o)$ opérateur du sommet de pile P_o	pile vide	+	•	(
+	0	1	1	0
•	0	0	1	0
(	0	0	0	0
)		1	1	3
fin	2	1	1	

Fig. V-4 - Matrice de décision

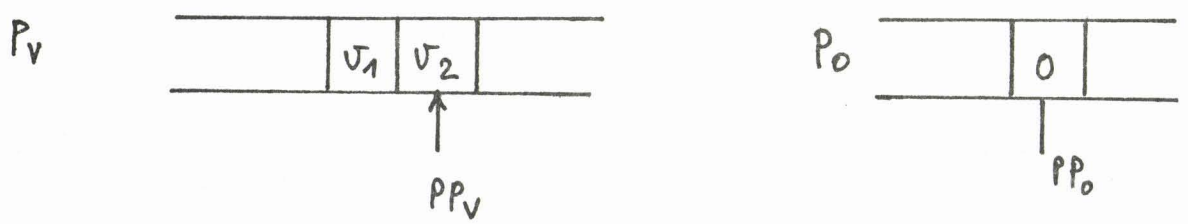
.../...

pour 1 a) générer un triple $(v_1, v_2, 0)$ où 0 est le sommet de la pile opérateur P_o et v_1, v_2 les sommets de la pile variable P_v les pointeurs PP_o et PP_v sont décrementés

b) Déterminer la variable intermédiaire où sera éventuellement rangé le résultat correspondant au triplet précédemment généré.

si v_1 ou v_2 désignent des variables intermédiaires le compteur CV est décrementé; le nom de la variable intermédiaire ou sera éventuellement rangé le résultat est alors % CV, ce dernier est rangé dans la pile variable.

La figure V.5 précise la structure des piles avant et après la génération d'un triplet.



a) avant génération

b) après génération

Fig. V-5

c) l'opérateur examiné est à nouveau comparé au sommet de la pile P_o

Pour 2 fin

générer l'affectation

pour 3 l'opérateur) examiné est une parenthèse fermante et le sommet de pile est une parenthèse ouvrante

a) dépiler la parenthèse ouvrante (décrementer PP_o)

b) si le caractère qui suit immédiatement la parenthèse fermante est l'opérateur de négation noté ', on doit placer dans la pile variable % i ' pour indiquer que la variable intermédiaire contient le complément du résultat.

c) on doit ensuite passer à l'opérateur suivant de l'expression

L'organigramme correspondant est donné à la Fig. V.6

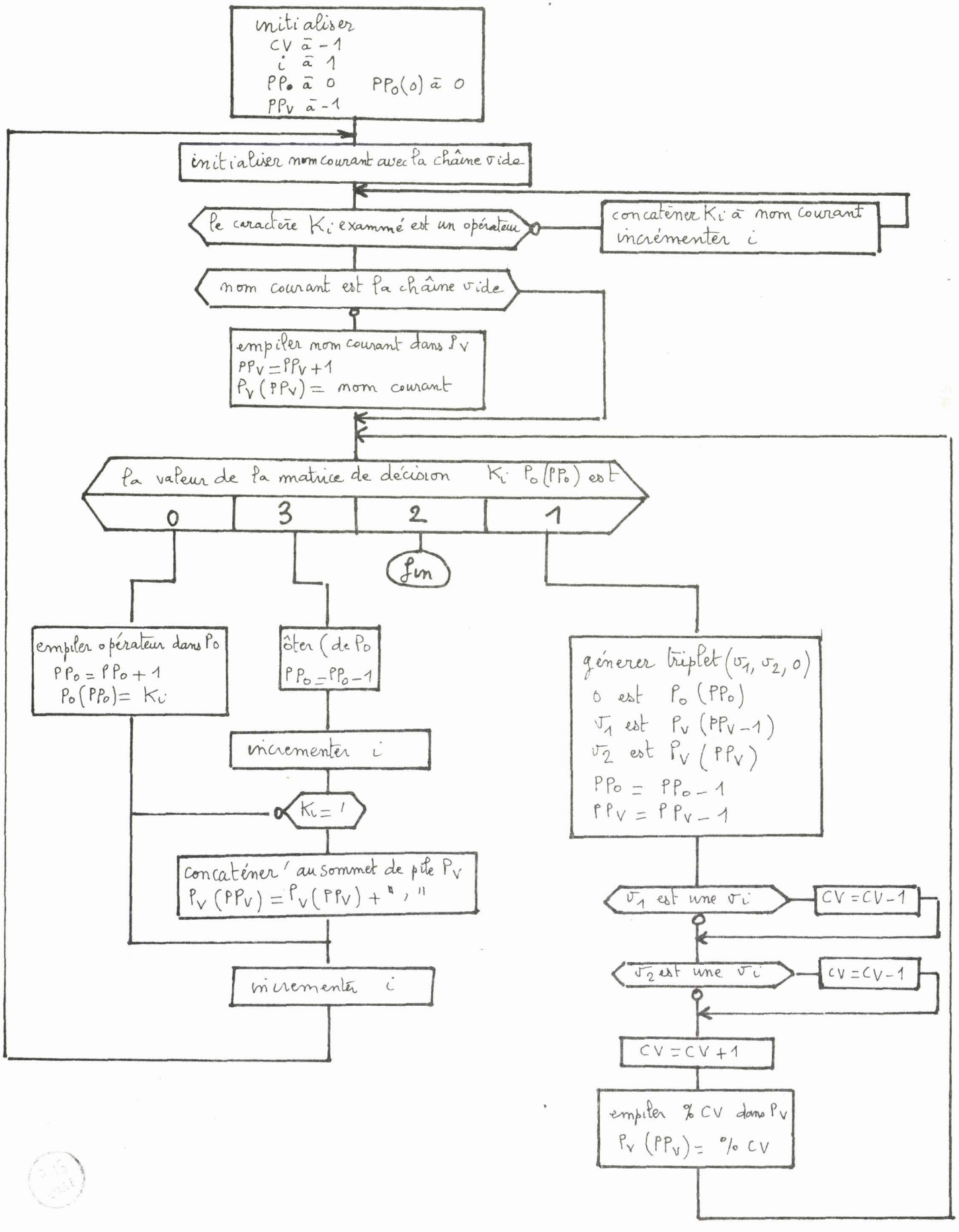

initialiser
CV ← -1
i ← 1
PP0 ← 0
PPV ← -1
PP0(0) ← 0
initialiser nom courant avec la chaîne vide
le caractère Ki examiné est un opérateur
concaténer Ki à nom courant
incrémenter i
nom courant est la chaîne vide
empiler nom courant dans PV
PPV = PPV + 1
PV(PPV) = nom courant
la valeur de la matrice de décision Ki P0(PP0) est
0
3
2
1
fin
empiler opérateur dans P0
PP0 = PP0 + 1
P0(PP0) = Ki
ôter (de P0
PP0 = PP0 - 1
incrementer i
Ki = '
concaténer ' au sommet de pile PV
PV(PPV) = PV(PPV) + " , "
incrementer i
générer triplet (σ1, σ2, 0)
0 est P0(PP0)
σ1 est PV(PPV - 1)
σ2 est PV(PPV)
PP0 = PP0 - 1
PPV = PPV - 1
σ1 est une σi
CV = CV - 1
σ2 est une σi
CV = CV - 1
CV = CV + 1
empiler % CV dans PV
PV(PPV) = % CV

Fig. V-6

La traduction de l'expression logique

F = ((a + b)' . (b + c . (d.e.(f+g) + (h +j)'))). i est donnée sur la figure V-7.

0	a	b	+	%0	contient :	a + b
1	d	e	.	%1	"	d . e
2	f	g	+	%2		f+g
3	%1	%2	.	%1		d.e (f+g)
4	h	j	+	%2		h+j
5	%1	%2'	+	%1		d.e.(f+g)+(h+j)'
6	c	%1	.	%1		c.(d.e.(f+g)+(h+j)')
7	b	%1	+	%1		b+ c.(d.e.(f+g)+(h+j)')
8	%0'	%1	.	%0		(a+b)'. (b+c.(d.e.(f+g)+(h+j)'))
9	%0	i	.	%0		F
10	=	F				

Fig. V-7

.../...

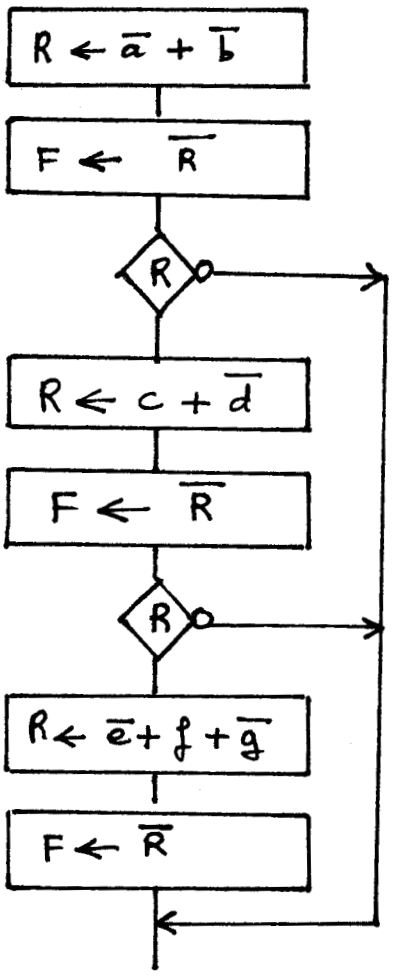
Nous donnons dans le paragraphe suivant les règles qui permettent la transcription dans le langage de l'automate retenu.

V.V Règles de transcription du langage intermédiaire dans le langage de l'automate.

Pour les machines dont les primitives logiques sont du type organigramme avec saut conditionnel, la transcription dans leurs langages ne pose pas de problèmes (cf exemple paragraphe V.III.6)

Ces méthodes peuvent être utilisées sur des machines qui disposent d'instructions de mise à 1 et à 0 conditionnelles. Le temps d'évaluation de l'expression n'est pas réduit, mais il n'est alors plus nécessaire de disposer de variables internes pour sauver les résultats intermédiaires. La figure V.8 donne l'implantation de l'expression logique :

$F = ab + \bar{c}d + e\bar{f}g$ sur le 5 TI et le 14500 B

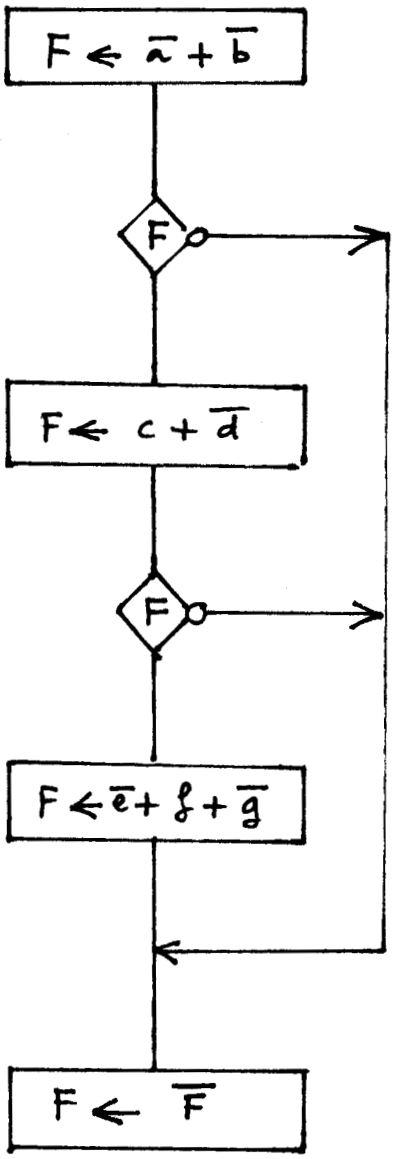



```
STR   NOT   a
OR    NOT   b
OUT   NOT   F

JMP         2
STR         c
OR    NOT   d
OUT   NOT   F

JMP         1
STR   NOT   e
OR          f
OR    NOT   g
OUT   NOT   F
```

5 T I




```
LDC     a
ORC     b
STO     F
OEN     F
LD      c
ORC     d
STO     F

OEN     F
LDC     e
OR      f
ORC     g
STO     F
ORC     RR
OEN     RR
LD      F
STOC    F
```

(14500B)

Fig. V-8

Examinons les règles qui permettent la transcription d'un triplet (v_1, v_2, 0) dans le langage de l'automate. Cette dernière peut d'abord conduire au rangement du résultat précédent. Pour cela certaines machines disposent d'une instruction de sauvegarde.

STO	%i	pour le 14500 B
OUT	%i	pour le 5 TI
= % i		pour le module SPRINT LOGIC

pour la gamme PB, on doit suivant l'opérateur du triplet précédent générer.

ET	%i
OU	%i

Pour le 14500 cette sauvegarde doit être effectuée si le triplet n'est pas le premier et si aucune variable du triplet n'est variable intermédiaire, ou si la dernière variable intermédiaire est complémentaire.

x	y	+ •	en cours de transcription
x	%i'	+ •	
%i'	y	+ •	

Sur la gamme PB, on doit en plus effectuer une sauvegarde, si l'opérateur est différent de celui du triplet precédent.

Sur le 5 II la pile évite de stocker les résultats partiels pour les variables intermédiaires avec i $\leq$ 5. Dans le cas où la dernière variable intermédiai-

.../...

re crée est complémentée on doit cependant effectuer une sauvegarde. Après un rangement du résultat précédent, la transcription conduit ensuite à un appel du premier opérande. Dans le cas contraire il suffit de générer une instruction du type

o v

(v représente la variable v_1 ou v_2). Le tableau de la fig.V.9 illustre ces règles pour 4 automates programmables.

	AVEC SAUVEGARDE DU RESULTAT PRECEDENT		SANS SAUVEGARDE DU RESULTAT PRECEDENT	
5 T.I	OUT STR [NOT] {AND / OR} [NOT]	% CV $\dot{v}_1$ $\dot{v}_2$	{AND / OR} [NOT]	$\dot{v}_1$ ou $\dot{v}_2$ ou STR
14500B	STO LD [c] {AND / OR} [c]	% CV $\dot{v}_1$ $\dot{v}_2$	{AND / OR} [c]	$\dot{v}_1$ ou $\dot{v}_2$
PB	{ET / OU} SI [/] SI [/]	% CV $\dot{v}_1$ $\dot{v}_2$	SI [/]	$\dot{v}_1$ ou $\dot{v}_2$
SPRINT LOGIC	= % CV {ET / OU} [PAS] $\dot{v}_1$ [PAS] $\dot{v}_2$		l'opérateur du triplet est identique au précédent [PAS] $\dot{v}_1$ ou $\dot{v}_2$	l'opérateur du triplet n'est pas identique au précedent * {ET / OU} [PAS] $\dot{v}_1$ ou $\dot{v}_2$

Fig. V-9

v_i désigne la variable v_i non complémentée

La figure suivante donne la transcription correspondant à l'expression

F = ((a +b)' . (b + c . (d . e.(f+g) + (h+J)'))). i

sur ces 4 machines.

	5 TI		14500B		PB		SPRINT LOGIC
a b +	STR	a	LD	a	SI	a	OU
	OR	b	OR	b	SI	b	a
							b
d e •	STR	d	STO	%0	OU	%0	=
	AND	e	LD	d	SI	d	%0
			AND	e	SI	e	ET
							d
							e
f g +	STR	f	STO	%1	ET	%1	=
	OR	g	LD	f	SI	f	%1
			AND	g	SI	g	OU
							f
							g
%1 %2 •	AND	STR	AND	%1	OU	%2	* ET
					SI	%1	%1
					SI	%2	
h j +	STR	h	STO	%1	ET	%1	=
	AND	j	LD	h	SI	h	%1
			AND	j	SI	j	OU
							h
							j
%1 %2' •	OUT	%2	STO	%2	ET	%2	=
	STR	%1	LD	%1	SI	%1	%2
	AND NOT	%2	ANDC	%2	SI/	%2	ET
							%1
							PAS
							%2
c %1 •	AND	c	AND	c	SI	c	c
b %1 +	OR	b	OR	b	ET	%1	* OU
					SI	b	b
					SI	%1	

%0 0' %1 •	AND NOT STR	AND C %0 0	OU %1 SI / %0 0 SI %1	* ET PAS %0 0
%0 i •	AND i	AND i	SI i	i
= F	OUT F	STo F	ET F	= F

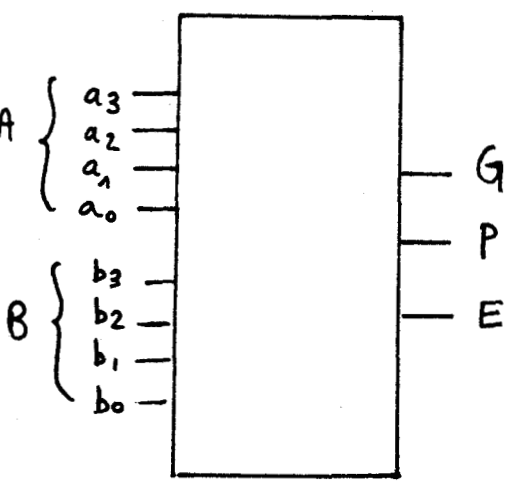
Fig. V-10

V. VI Implantation des prédicats numériques .

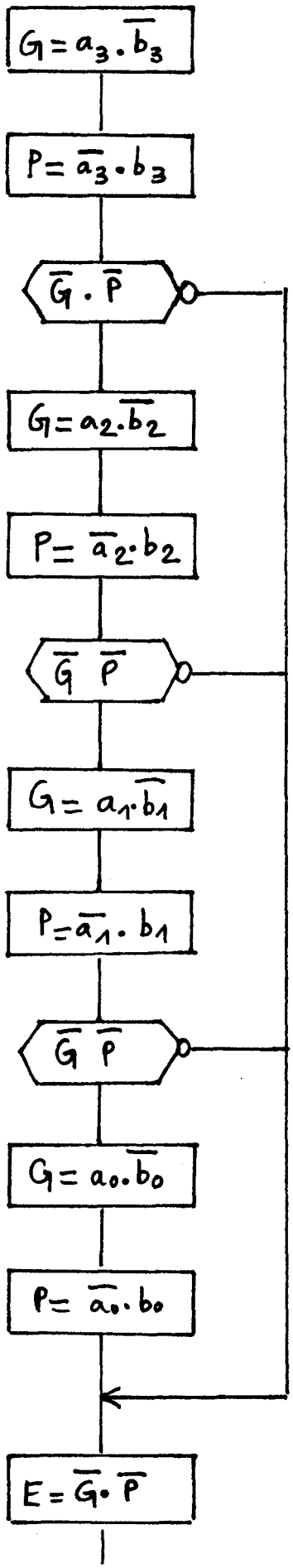
L'implantation des prédicats numériques ne pose aucun problème si la machine dispose d'instruction qui permettent de les réaliser. (cas de la gamme PB et du TSX 80). Sur les autres elle doit être confiée au traducteur spécialisé final. Nous donnons l'implantation type d'un module qui permettra de réaliser la comparaison de nombres de 4 bits A et B pour le 5 TI et le 14500.

$$A = a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

$$B = b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0$$



	G	E	P
A > B	1	0	0
A = B	0	1	0
A < B	0	0	1




```
STR      a3      LD      a3
AND NOT  b3      ANDC    b3
OUT      G       STO     G

STR  NOT a3      LDC     a3
AND      b3      AND     b3
OUT      P       STO     P

STR  NOT G       LDC     G
AND  NOT P       ANDC    P
JMP      6       STO     VI
                 OEN     RR
STR      a2      LDC     a2
AND  NOT b2      ANDC    b2
OUT      G       OUT     G

STR  NOT a2      LDC     a2
AND      b2      AND     b2
OUT      P       OUT     P
                 LDC     G
STR  NOT G       ANDC    P
AND  NOT P       AND     VI
JMP      4       STO     VI
                 OEN     RR
STR      a1      LD      a1
AND  NOT b1      ANDC    b1
OUT      G       STO     G

STR  NOT a1      LDC     a1
AND      b1      AND     b1
OUT      P       STO     P
                 LDC     G
STR  NOT G       ANDC    P
AND  NOT P       AND     VI
JMP      2       STO     VI
                 OEN     RR
STR      a0      LD      a0
AND  NOT b0      ANDC    b0
OUT      G       STO     G

STR  NOT a0      LDC     a0
AND      b0      AND     b0
OUT      P       STO     P


                 ORC     RR
STR  NOT G       OEN     RR
AND  NOT P       LDC     G
OUT  E           ANDC    P
                 STO     E
```

Fig. V-11

Il reste à déterminer les commandes des dispositifs particuliers des automates programmables à savoir les temporisations et les compteurs décompteurs.

V. VIII Les temporisations et les compteurs décompteurs.

La grande diversité des solutions retenues pour la réalisation de ces éléments par les différents constructeurs d'automates programmables rend délicat une mise en oeuvre générale indépendante du matériel.

V.VIII.1 les temporisations:

La temporisation la plus simple est un dispositif à une entrée et une sortie; un front positif sur son entrée fait passer sa sortie à 1 au bout d'un temps τ précisé

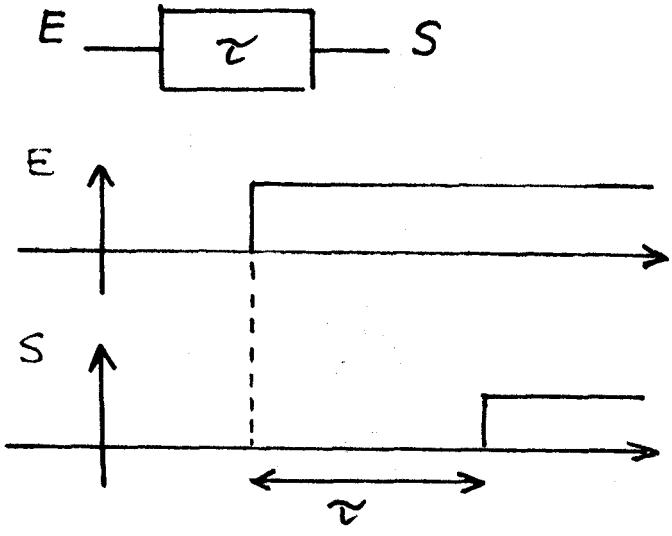


Fig. V-12

Cette temporisation du type retard à l'enclenchement permet d'élaborer tous les autres types.

Des possibilités de remise à zéro et de gel ont été ajoutées sur certains matériels.

Le fonctionnement d'une temporisation la plus générale peut être décrit par le Grafcet de la figure V.13

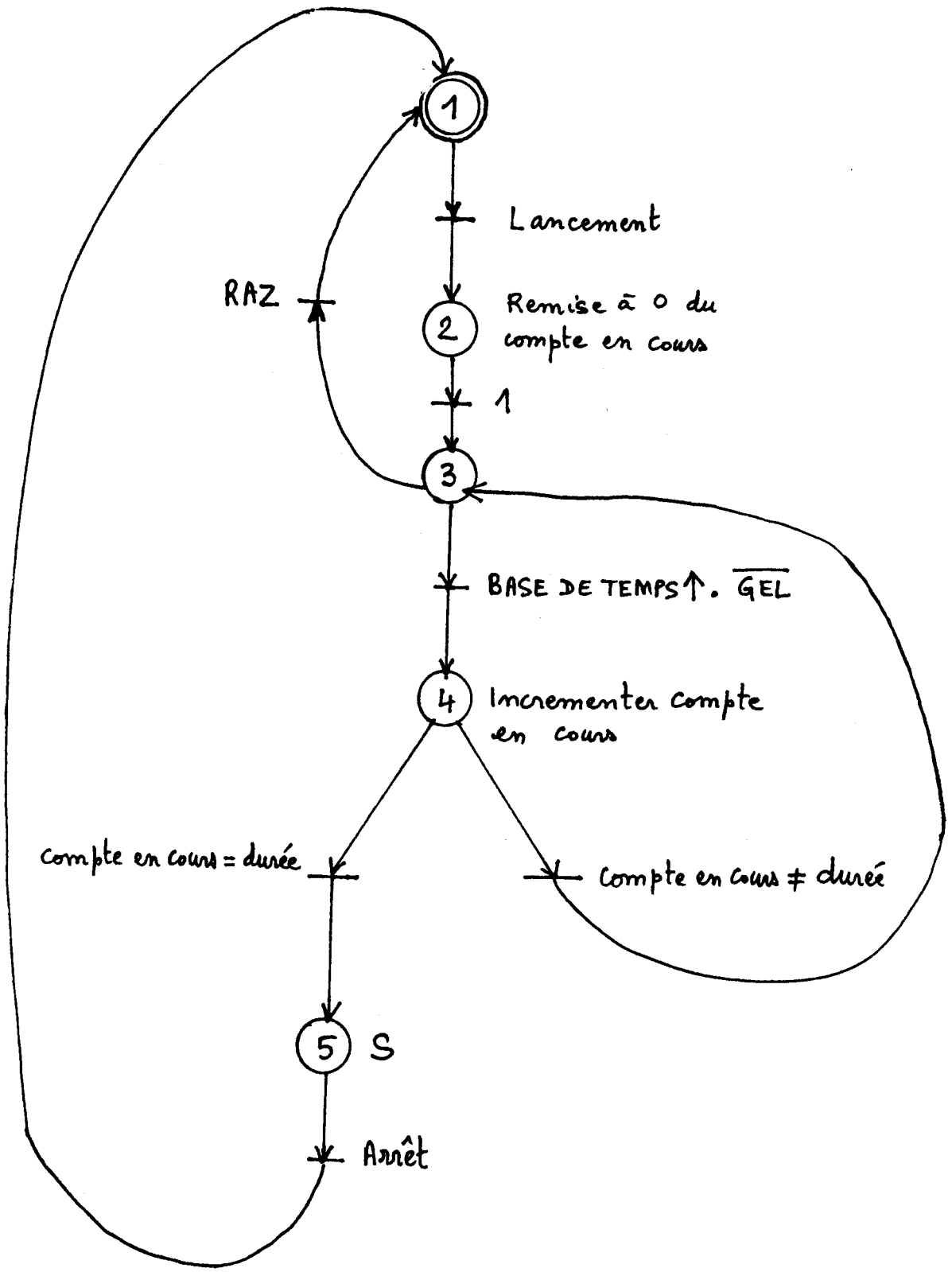
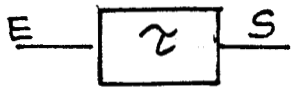

1
Lancement
RAZ
2
Remise à 0 du compte en cours
1
3
BASE DE TEMPS↑. $\overline{GEL}$
4
Incrementer compte en cours
compte en cours = durée
compte en cours ≠ durée
5
S
Arrêt

Fig. V-13

A titre d'exemple, nous citons les temporisations de la gamme PB de Merlin Gérin et du 5 TI de Texas (Fig.V.13)



0	SI	E
1	TP	S
2	BT	choix de la base de temps
3	Adresse durée temporisation	
4	Adresse compte en cours	

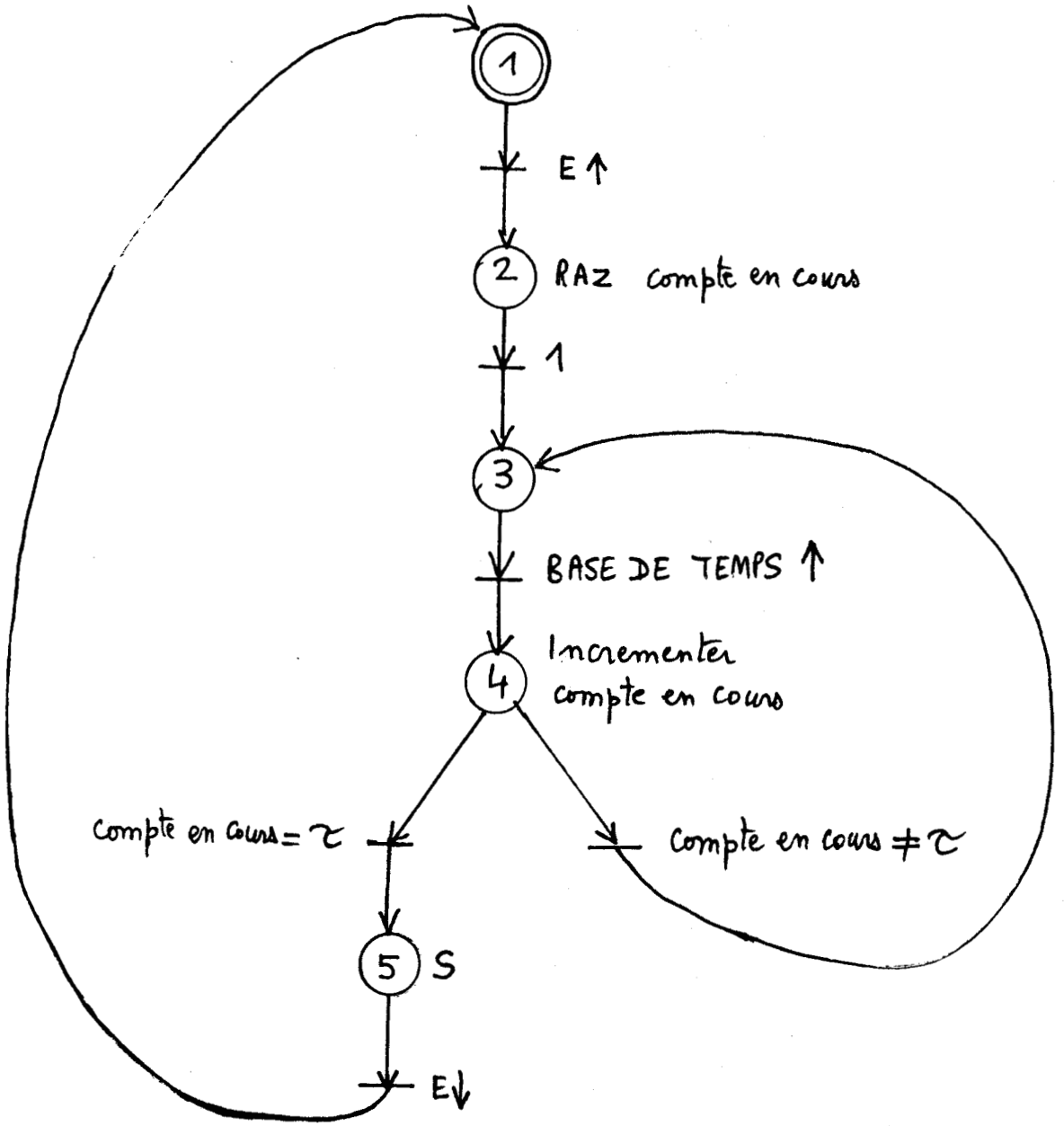
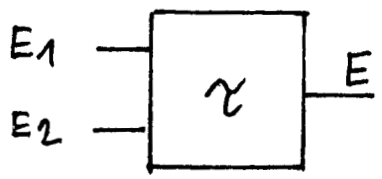


Fig. V-14

P B



0 STR E1
1 STR E2
2 TMR *
3 durée
4 compte en cours
5 OUT S

* ou TMR 1 suivant la base de temps

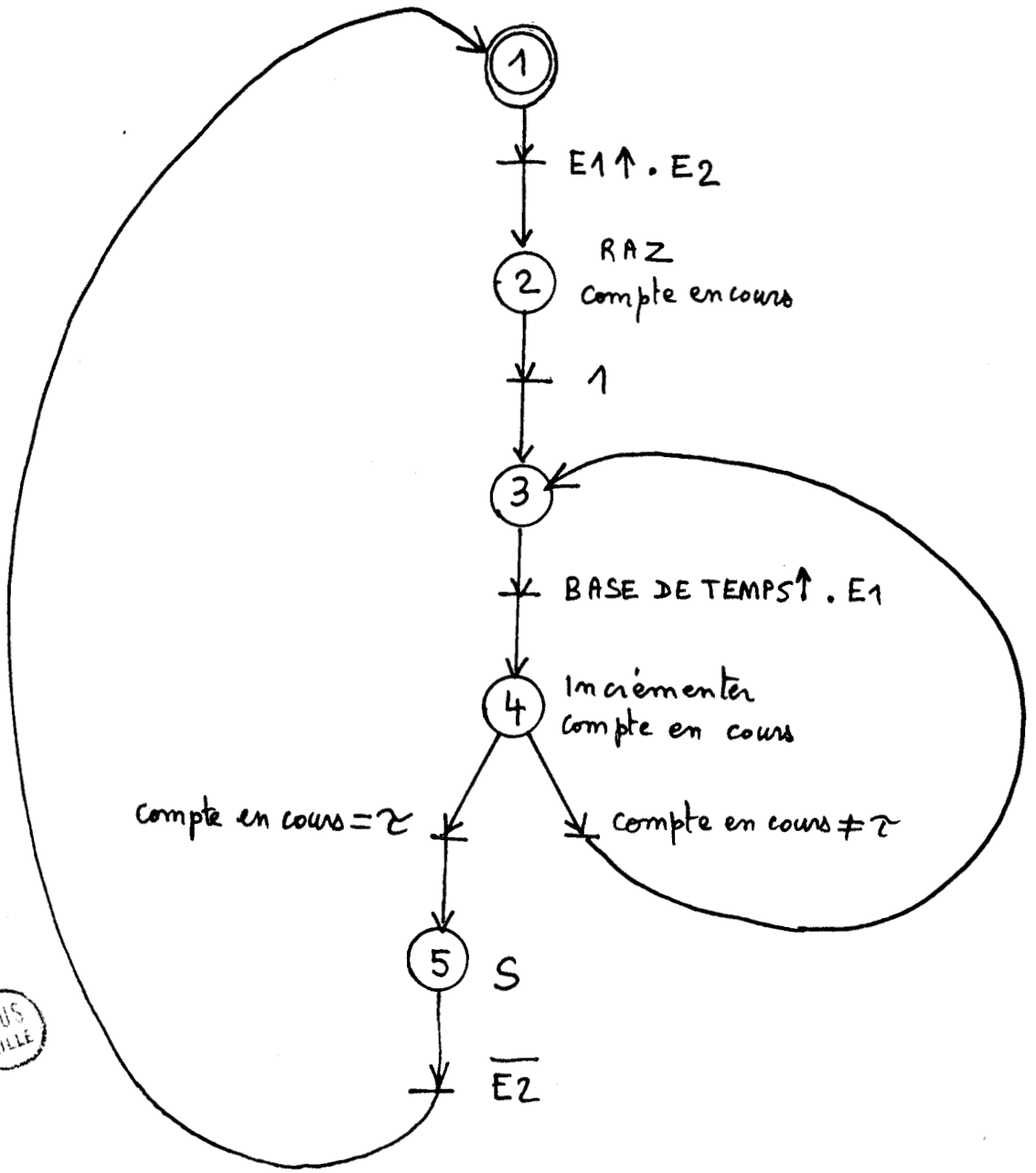


Fig. V-15

Les automates qui disposent d'instructions de traitement de mots offrent la possibilité d'accèder au compte en cours et ainsi de contrôler le temps écoulé.

V. VII 2 Les compteurs décompteurs

La plupart des automates programmables permettent de compter ou décompter les fronts montants d'une variable

La fonction comptage -décomptage ainsi proposée, est analogue à la fonction temporisation ; car dans le premier cas on compte ou décompte les variations d'une variable d'entrée ou interne, dans le second celles d'une base temps.

On trouvera ainsi, une commande de comptage décomptage, une remise à zéro. validation et l'entrée dont on traite les variations. La détection des fronts à prendre en compte peut être réalisée directement par l'automate (cas du 5 TI) ou non (cas du PB). Dans ce cas l'entrée du compteur doit être gérée en variable monscoup.

La sortie passe à un lorsqu'on a atteint le nombre fixé, ou la valeur zéro. Les automates qui disposent d'instructions de traitement sur mots permettent de tester la valeur du compte atteint.

Au contraire, certains matériels de bas de gamme ne disposent pas directement de possibilités de comptage décomptage. Nous proposons un module réalisant cette deuxième fonction, pour le TEC 16 (automate réalisé à partir du microprocesseur 14500). La valeur de départ est :

$$a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

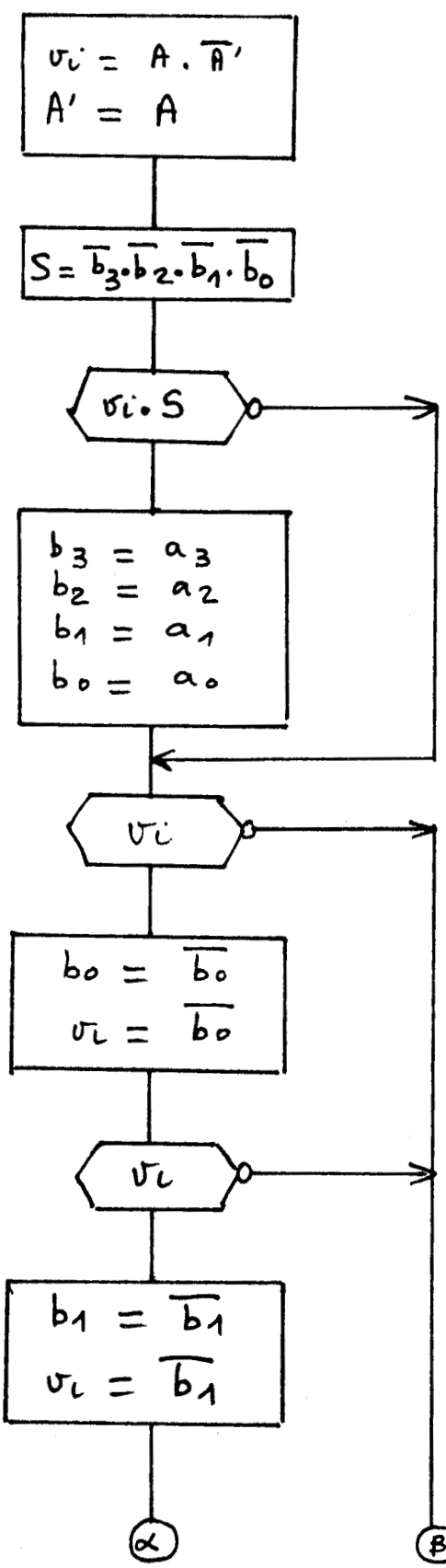
le compte en cours :

$$b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0$$

.../...

l'entrée à prendre en compte A , la sortie S (Fig. V.16)

Ce module peut alors être utilisée comme un sous programme.




```
LD    A
ANDC  A'
STO   vi
LD    A
STO   A'
LDC   b0
ANDC  b1
ANDC  b2
ANDC  b3
STO   S

AND   vi
OEN   RR

LD    a3
STO   b3
LD    a2
STO   b2
LD    a1
STO   b1
LD    a0
STO   b0

OEN   vi

LDC   b0
STO   b0
STO   vi

OEN   vi

LDC   b1
STO   b1
STO   vi
```

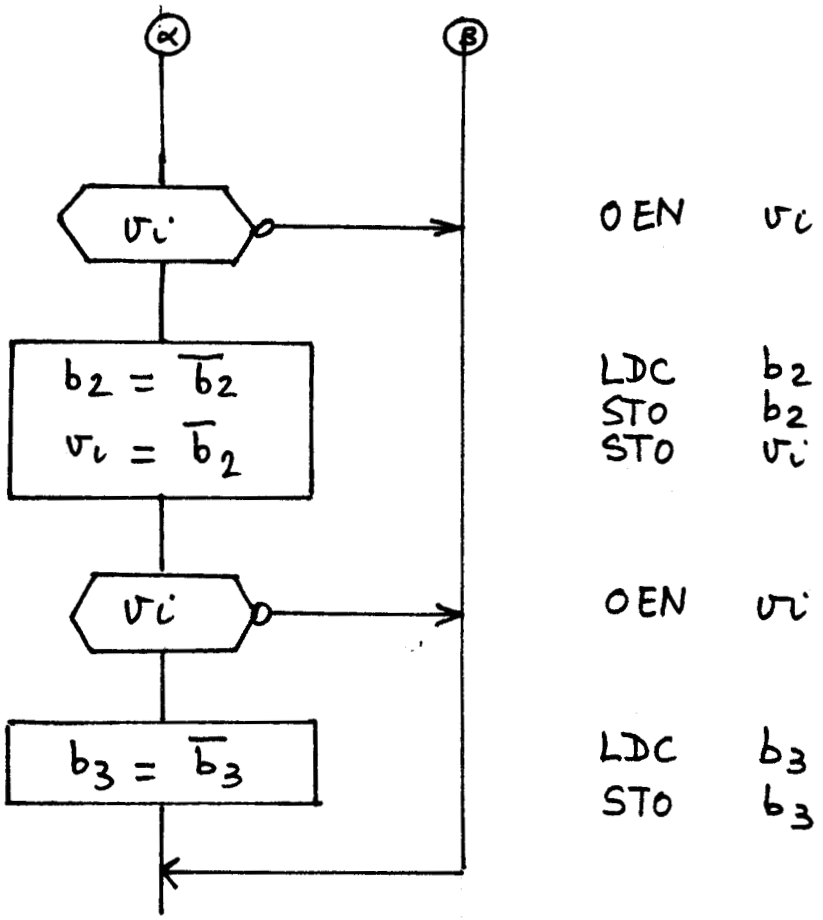


Fig. V - 16

Le système traducteur réalisé permet l'implantation directe d'un Grafcet sur tous les automates programmables et le panneau de programmation Prograf.

Nous avons étudié le module permettant une implantation sur le microprocesseur monochip 8748 de Intel. Une étude sur la définition d'un langage standard pour microprocesseur est en cours, elle conduira à une implantation sur tous les microprocesseurs. Nous prévoyons également une extension pour les dispositifs semi programmés (FPAL, PAL, ...) et câblés.

CONCLUSION

Ce travail montre la possibilité de réaliser un outil d'aide à la conception en logique industrielle sur un matériel d'un coût modeste. Les possibilités graphiques que nous avons testé sur ce type de matériel ne nous paraissent pas encore satisfaisantes, mais l'évolution actuelle des processeurs graphiques devrait permettre à cours terme des réalisations éventuellement couleur de grande qualité. La structure modulaire permet toute forme d'adaptation, modification et extension en particulier à tous les types de microprocesseurs.

BIBLIOGRAPHIE

(1) AFCET. Rapport de la "commission de normalisation de la représentation du cahier des charges d'un automatisme logique" (Groupe de travail AFCET "systèmes logiques") Présentation des travaux par M. BLANCHARD. Revue automatisme tome XXII n° 3-4 Mars-Avril pp 66-83

(2) C. ANDRE. Sur une méthode de conception assistée par ordinateur des systèmes logiques à évolutions simultanées. Thèse 3ème cycle. Nice.Juin 1975

(3) M. AUGUIN. Conception des systèmes de commande à l'aide de réseaux logiques programmables. Thèse 3ème cycle. Nice. Novembre 1978

(4) F.L. BAUER et K. SAMELSON. Sequential formula translation. C.A.C.M. n° 3 1960 pp 76-83

(5) M. BLANCHARD. Comprendre, maîtriser et appliquer le GRAFCET. CEPADUES EDITIONS 1979

(6) J.L BOUSSIN. Systèmes de conception assistée par ordinateur et de validation des automatismes logiques. Journées d'études: Les méthodes modernes de réalisation des automatismes. Paris. Février 1978

(7) R. CLEAZ. Monitoring and Control Language. Langage adapté à la transcription du Grafcet. AFCET. Décembre 1979

(8) J.DEFRENNE. Implantation des réseaux de Pétri sur automate bi-processeur à haute sûreté de fonctionnement. Thèse 3ème cycle. Lille. Juin 1979

(9) C.GELE et D. MANSION. Panorama des automates programmables: 150 modèles présentés. Le nouvel automatisme n° 9. Novembre 1979 pp 29-38 et n° 10. Décembre 1979 pp 29-45

(10) G. MICHEL C. LAURGEAU B.ESPIAU. Les automates programmables industriels. DUNOD 1979

(11) A. POULAIN J.M. TOULOTTE. Implantation du Grafcet et des graphes fonctionnels sur panneau de programmation. Le nouvel automatisme Mars 1980. pp 43-50

(12) F. PRUNET A.REBOUL B.CARRIERE. Un exemple d'outil d'implantation de cahiers des charges. Journées AFCET Décembre 1979

(13) C.RAMCHANDANI. Analysis of asynchronous concurrent systems by Petri nets. MAC TR-120 Février 1974 MIT

(14) J. REINALIER. Analyse et simulation en APL des systèmes de commandes décrits par les réseaux de Pétri. Thèse 3ème cycle. Toulouse 1977

(15) S. SILVA SUAREZ. Contribution à la synthèse programmée des automatismes logiques. Thèse de Docteur ingénieur. INP Grenoble. 1978

(16) B. TACONNET B. CHOLLOT. Programmation du Grafcet sur automates à langage logique, à relais ou booléen. Le nouvel automatime n° 4 Janvier-Février 1979 pp 41-45

(17) S. THELLIEZ J.M. TOULOTTE. Grafcet et logique industrielle programmée. EYROLLES 1980

(18) J.M. TOULOTTE. Réseaux de Pétri et automates programmables. automatisme Tome XXII n° 7-8 Juillet-Aout 1978 pp 200-211

(19) J.M. TOULOTTE. J.P PARSY.A method for decomposing interpreted Petri nets and its utilization. Digital processes 5 (1979) 3-4

(20) J.M TOULOTTE. La fiche idéale d'un automate programmable. Le nouvel automatisme. Septembre-Octobre 1979 pp 22-26

(21) R. VALETTE. Sur la description, l'analyse et la validation des systèmes de commande parallèles. Thèse de Doctorat d'état. Toulouse. Novembre 1976

(22) UNION TECHNIQUE DE L'ELECTRICITE. Diagramme fonctionnel "Grafcet" pour la description des systèmes logiques de commande. Projet de norme NF C 03-190 (27 Octobre 1980)