L'enseignement du langage Grafcet et ses interprétations

Jean Vareille, Mireille Arnoux

L'ENSEIGNEMENT DU LANGAGE GRAFCET ET DE SES INTERPRÉTATIONS

Jean VAREILLE, Mireille ARNOUX

1 - INTRODUCTION

Dans l'enseignement technologique, l'étude des automatismes industriels occupe une place importante. Les élèves apprennent à se servir d'Automates Programmables Industriels (API) acceptant des programmes « grafcet », dont la spécification graphique est attrayante et claire. Grâce à sa simplicité elle permet de décrire aisément des fonctionnements séquentiels et parallèles. Pourtant, expliquer le fonctionnement d'un système à partir d'un GRAFCET [1], malgré la présence de règles rigoureuses, n'est pas si évident [7]. Les praticiens savent bien que le transport d'une application d'un automate à un autre n'est pas chose facile, et qu'il arrive que l'on doive retoucher un grafcet en changeant d'automate car ils n'échantillonnent pas les entrées à la même cadence, ou ne suivent pas exactement la même algorithmique. La raison fondamentale en est simple. Les API sont construits à base de systèmes microinformatiques dont les cœurs sont des microprocesseurs ou des microcontrôleurs qui opèrent des traitements séquentiels, c'est à dire une instruction après l'autre. Or les systèmes automatisés sont composés de capteurs et d'effecteurs (moteurs, résistances…) qui saisissent, mesurent et agissent simultanément. Pour étudier ces systèmes parallèles, le modèle GRAFCET est bien adapté. Mais la transformation d'un graphe qui décrit le comportement d'un système parallèle par nature, en un programme exécutable par un processeur séquentiel, entraîne obligatoirement une interprétation algorithmique du graphe. Deux familles principales d'interprétation sont acceptables, or dans certains cas elles mènent à des fonctionnements différents à partir d'un seul et même GRAFCET. Les deux interprétations classiques ont été nommées :

1 « Grafcet » désigne le langage, « grafcet » désigne un programme, et enfin « GRAFCET » désigne la spécification graphique.

interprétation Avec Recherche de Stabilité (ARS), et interprétation Sans Recherche de Stabilité (SRS). Aujourd'hui, la totalité des automates commercialisés suivent plus ou moins bien l'interprétation SRS bien qu'elle ne soit pas rigoureusement déterministe. En effet on ne peut garantir le comportement d'un logiciel qui suit l'algorithmique SRS vis à vis d'une succession d'événements captés en entrée, car il peut présenter des instabilités dont les conséquences seront imprévisibles.

L'introduction du Grafcet en tant que langage nécessite de la précision, et la définition d'une sémantique opérationnelle. L'objectif de cet article est de présenter une chaîne de développement et de vérification mettant en œuvre cette approche. Nous pensons qu'il s'agit d'une étape indispensable pour permettre le transport des programmes d'une machine à l'autre. Ce problème, qui commence à être bien résolu dans le domaine de l'informatique générale, devrait interpeller les constructeurs des Automates Programmables Industriels.

Une activité importante de notre laboratoire de recherche est centrée autour de la modélisation du comportement du Grafcet et de la vérification de leurs propriétés.

Nous avons mené ces dernières années une action dont le but était de confronter des programmes respectant l'une ou l'autre des interprétations, au fonctionnement concret de maquettes, ainsi que de soumettre au verdict de la réalité, nos outils de preuve de programmes.

Les maquettes de systèmes automatisés sont de la marque FischerTechnik. Elles sont pilotées par ordinateurs compatibles PC.

L'équipe de recherche LIMI a développé un éditeur de GRAFCET et des outils de compilation en divers langages, essentiellement les langages synchrones, ainsi que des outils de preuve. Nous nous sommes donc efforcer de lier la chaîne des outils du LIMI aux maquettes disponibles.

Le but de cet article est de présenter ce travail et de montrer qu'il contribue à l'approche rigoureuse du Grafcet, approche que nous souhaitons promouvoir dès le niveau pré-baccalauréat.

La section 2 rappelle les règles et postulats de base du modèle GRAFCET.

Les deux grandes interprétations du Grafcet et leurs différences sont présentées sur un exemple à la section 3.

La section 4 présente la chaîne de développement du LIMI.

Enfin, la section 5 dresse un bilan du travail effectué et propose quelques perspectives.

2 - LES RÈGLES DU GRAFCET

Le GRAFCET (Graphe Fonctionnel de Commande Étapes/Transitions) [1] est, à l'origine, une méthode de spécification graphique pour les automatismes logiques. Son développement a commencé dès 1977 à l'initiative du groupe Systèmes Logiques de l'AFCET. Normalisé nationalement [8] et internationalement [2] sous le nom de *Sequential Function Chart*, il est enseigné dès le secondaire et bénéficie ainsi d'une large diffusion. Un GRAFCET est composé de trois types d'éléments :

- les étapes modélisent l'état du système. L'ensemble des étapes actives à un instant donné constitue la situation du GRAFCET et les actions associées sont exécutées,
- les transitions contrôlent le passage d'une étape à une autre, par l'intermédiaire des réceptivités qui leur sont associées,
- les liens modélisent la structure du GRAFCET. Ils relient les étapes aux transitions et les transitions aux étapes.

Le GRAFCET est basé sur deux postulats et cinq règles d'interprétation :

Postulat 1 : tous les événements sont pris en compte dès leur occurrence et pour toutes leurs incidences.

Postulat 2 : la causalité est considérée à temps nul.

Règle 1 : la situation initiale d'un GRAFCET caractérise le comportement initial de la partie commande vis à vis de la partie opérative, de l'opérateur et/ou des éléments extérieurs. Elle correspond aux étapes actives au début du fonctionnement. Elle traduit généralement un comportement de repos.

Règle 2 : une transition est dite validée lorsque toutes les étapes immédiatement précédentes reliées à cette transition sont actives. Le franchissement d'une transition se produit :

- lorsque la transition est validée,
- et que la réceptivité associée à cette transition est vraie.

Règle 3 : le franchissement d'une transition entraîne simultanément l'activation de toutes les étapes immédiatement suivantes et

la désactivation de toutes les étapes immédiatement précédentes.

Règle 4 : plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle 5 : si au cours du fonctionnement, la même étape est simultanément activée et désactivée, elle reste active.

Les deux postulats induisent un synchronisme entre les événements d'entrée et les sorties. Malgré la définition précise des règles, plusieurs interprétations existent.

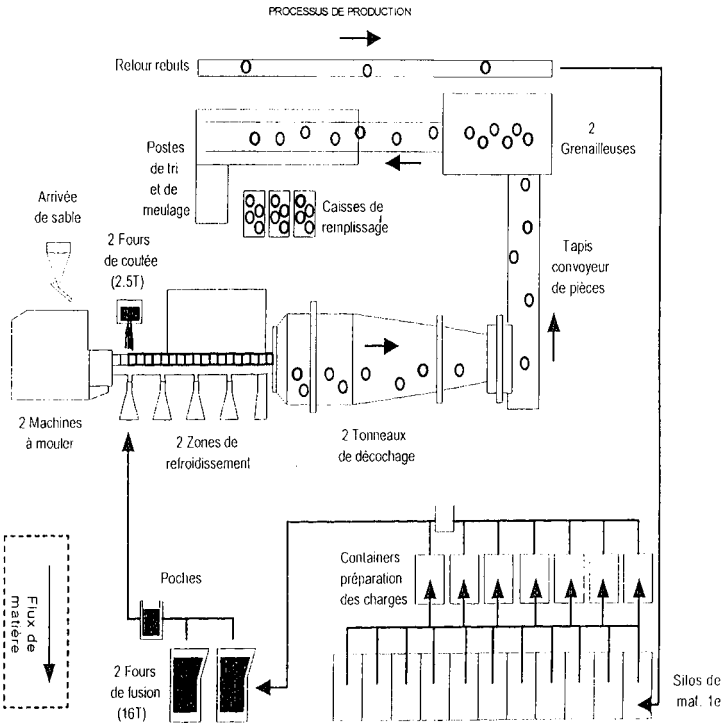

FIG.1 : Atelier de fonderie

3 - INTERPRÉTATIONS DU GRAFCET

Nous allons présenter les deux grands algorithmes d'interprétation sur un exemple tiré du sujet de l'épreuve d'automatismes de la session de 1994 de l'agrégation externe de génie mécanique [5]. Il s'agit de la spécification et de la réalisation d'une partie du système de commande d'un atelier de fonderie représenté sur la figure 1. La figure 2 donne le GRAFCET G1 de la commande de trappe d'arrivée de sable.

Il peut être vu comme un simple cycle d'ouverture *AC_1* - fermeture *AC_2* d'une trémie, *a* et *b* étant des capteurs de fin de course du volet, *a = fermé, b = ouvert*. Le front montant de *m* provoque l'enclenchement du cycle. En fin de cycle la trémie est fermée et *a=1 (vrai)*. L'exécution des actions associées à ce GRAFCET (*AC_1* et *AC_2*) peut être gérée de deux manières.

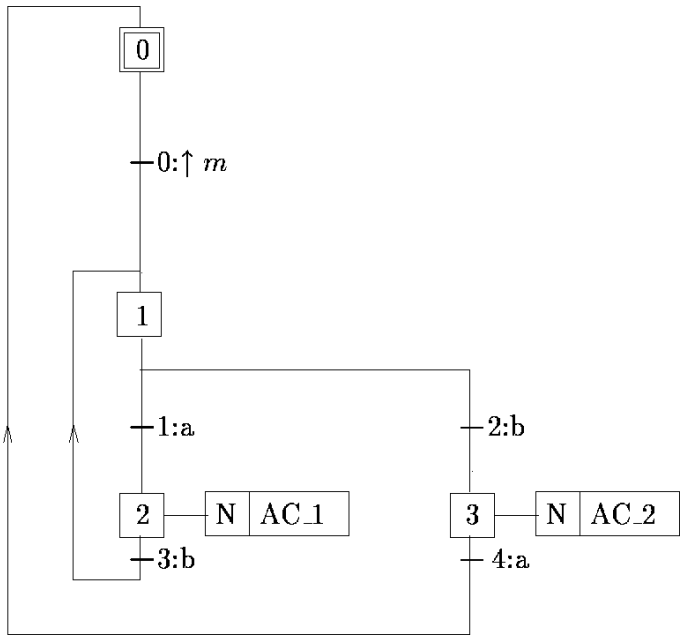

FIG.2 : GRAFCET G1

La première consiste à exécuter uniquement les actions associées aux étapes actives lorsque la partie commande est stable. La figure 3 présente cette algorithmique, appelée Avec Recherche de Stabilité (ARS).

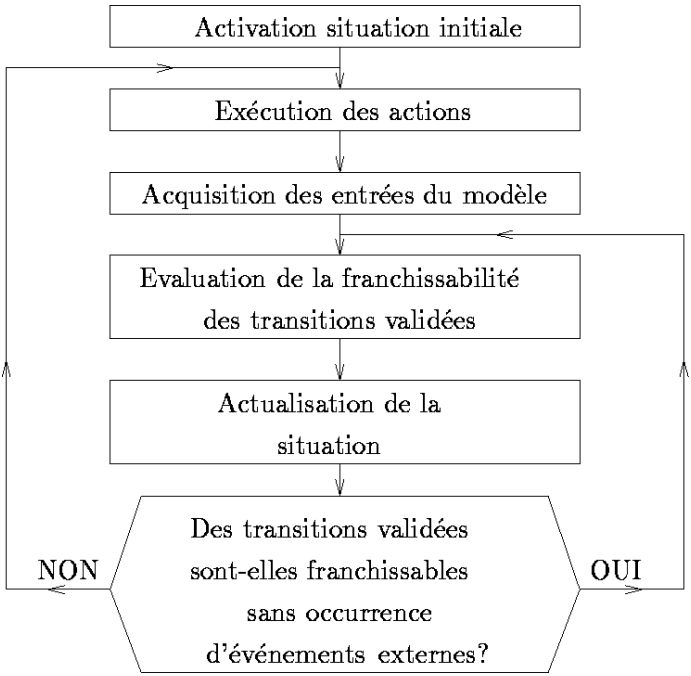

FIG.3 : Algorithme avec recherche de stabilité

Une situation est dite stable si aucune transition du grafcet n'est franchissable sans changement d'état d'une variable d'entrée. L'instabilité de la partie commande peut être causée par le dysfonctionnement d'un capteur. Imaginons qu'au cours du dernier cycle achevé *b* reste bloqué à *1 (vrai)*. Ce cycle connaît une fin normale. Par contre au début du cycle suivant le démarrage sera instable, car *a* et *b* seront simultanément à *1*. Or l'instabilité met en péril l'actionneur d'ouverture (ou son préactionneur).

Le graphe des situations atteignables est présenté sur la figure 4. Les situations stables sont symbolisées par des rectangles, les instables par des ellipses, */a* signifie *non a* et */b* signifie *non b*. On peut remarquer une boucle de cycle perpétuel (états (0,1) et (0,2,3)). Dans cette boucle instable, aucune action n'est exécutée. L'intégrité de la partie opérative est donc préservée.

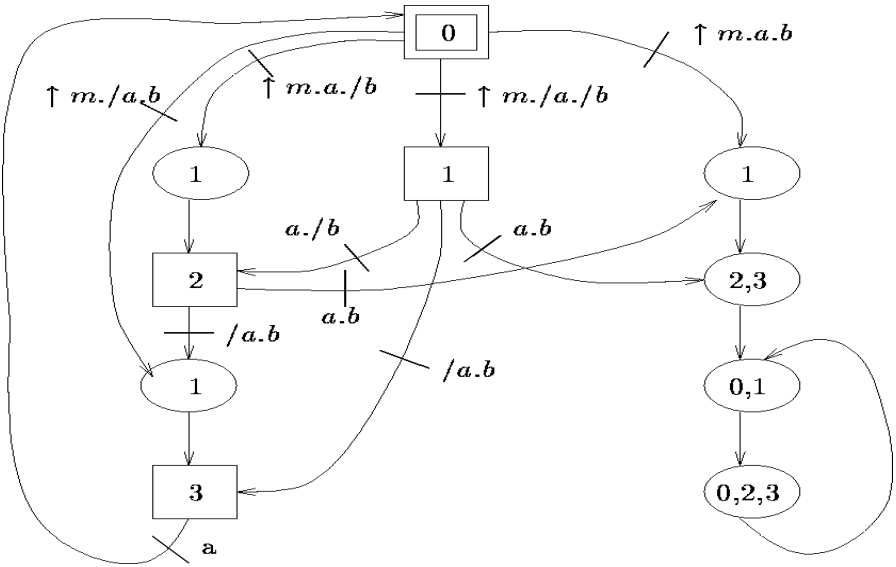

FIG. 4 : Graphe des situations atteignables

La seconde manière consiste à exécuter les actions associées aux étapes actives sans souci de stabilité de la partie commande. Elle correspond à l'algorithmique sans recherche de stabilité (SRS) qui est présentée sur la figure 5. A chaque cycle, les actions sont exécutées et les entrées sont scrutées. Contrairement à l'algorithmique ARS, il n'y a pas de boucle de cycle perpétuel. En revanche, les actions associées aux états du cycle instable précédent sont exécutées, alors qu'elles sont contradictoires (on tente de faire fonctionner le moteur simultanément dans les deux sens).

Dans le sujet original, la première manière était proposée, dans un souci de sécurité.

Les constructeurs d'Automates Programmables Industriels utilisent l'algorithmique SRS. Pour éviter toute contradiction entre les ordres, ils proposent un traitement postérieur au Grafcet, avant l'affectation des sorties. Ce post - traitement sort du modèle GRAFCET.

Il est clair que l'exemple est structuré de façon à ce que l'instabilité y soit inéluctable. Une légère modification des réceptivités peut la faire disparaître. En général, les risques de contradiction sont beaucoup moins flagrants.

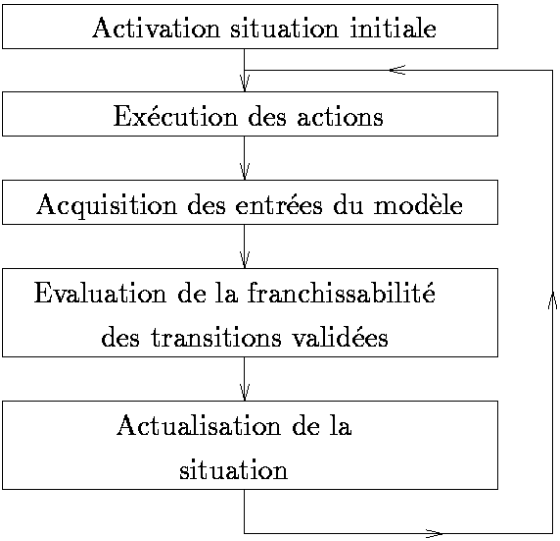

FIG. 5 : Algorithme sans recherche de stabilité

4 - CHAÎNE DE DÉVELOPPEMENT GRAFCET DU LIMI

La chaîne de développement présentée ici permet de générer un programme grafcet à partir d'une spécification GRAFCET de façon plus sûre. Le choix entre les deux interprétations SRS et ARS, doit être précisé. Les problèmes d'atteignabilité des étapes et les bouclages sont signalés par les outils de preuve.

Cette chaîne est composée par un ensemble de logiciels interconnectés, qui sont présentés sur la figure 6.

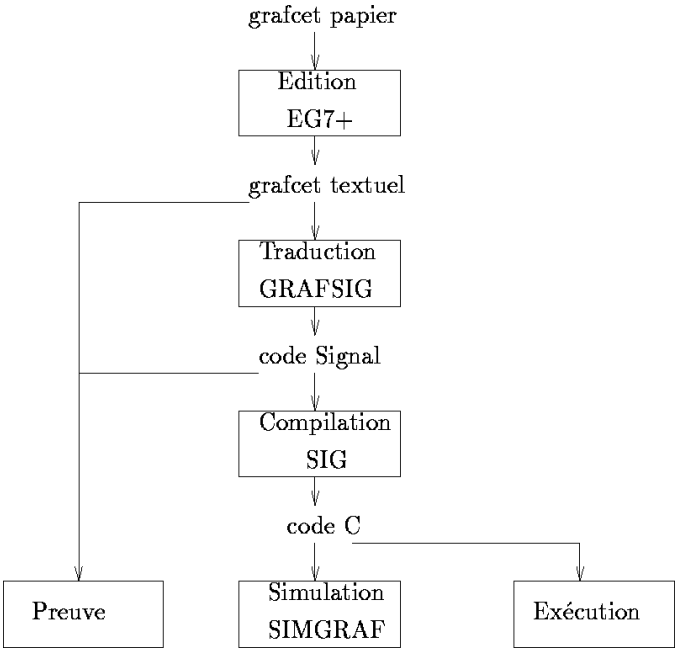

FIG. 6 : Chaîne de développement grafcet

4.1 - L'éditeur de GRAFCET

Le premier élément de la chaîne est un éditeur de GRAFCET, nommé EG7+.

La figure 7 montre le grafcet de l'exemple saisi sous l'éditeur EG7+, *moteur1h*, *moteur1t*, *capteur3*, *capteur5* et *capteur7* sont des noms reconnus par l'outil qui adapte le code généré par Signal à l'interface Fischer-Technik.

4.2 - Les outils de preuve du grafcet textuel

Ces outils de preuve permettent de démontrer des propriétés sur le grafcet. Le lecteur intéressé peut se référer à [6]. Leur utilisation débute par une génération automatique des états comme celle de la figure 4, grâce à l'outil TEMPOS. Nous pouvons générer les états de la partie commande dans le cas des deux algorithmiques, ARS et SRS. Il en sort un fichier textuel, retraité ensuite par un outil de preuve, qui permet de vérifier des propriétés du graphe d'états. La conformité du comportement de la partie commande à son cahier des charges peut ainsi être prouvée dès la fin de sa conception.

4.3 La traduction en langage SIGNAL

Le GRAFCET textuel, avec son interprétation SRS ou ARS choisie par l'utilisateur, est traduit par GRAFSIG en langage SIGNAL [3], qui est un langage synchrone du type flot de données. Il a pour but la programmation des systèmes réactifs. Ils est fondé sur l'hypothèse dite de synchronisme fort :

1. L'ensemble des instants où les valeurs des signaux d'entrée sont disponibles est ordonné. Seul l'ordre d'arrivée est important. On se réfère donc à un temps logique et non pas métrique.
2. Les sorties sont présentes simultanément avec les entrées qui les ont engendrées, mais elles dépendent fonctionnellement des entrées. Les calculs et les communications internes sont donc instantanés.

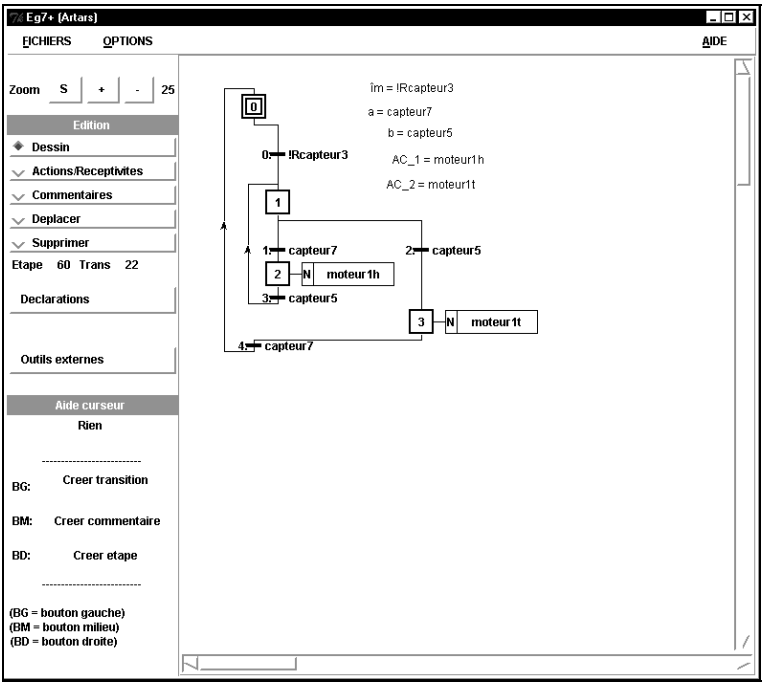

FIG. 7 : Exemple d'édition de Grafcet avec EG7+

Ce langage est proche du GRAFCET. En particulier, on retrouve dans le point 2 les postulats de base du GRAFCET. Son aspect déclaratif en flot de données permet de définir facilement les relations qui lient

chaque étape avec son environnement proche, et d'exprimer aisément le parallélisme implicite inhérent au Grafcet [4].

Ce sont des langages très formels, qui reposent sur une sémantique mathématique précise et disposent d'outils de preuve. Cela les oppose au Grafcet, langage graphique et « confortable ».

Nous proposons une méthodologie où le Grafcet est utilisé comme langage utilisateur et où la puissance des langages synchrones sert la formalisation. C'est la traduction en langage SIGNAL qui fixe la sémantique du GRAFCET.

Une spécification GRAFCET, dans le cas d'une interprétation Sans Recherche de Stabilité, produit un programme Signal dont l'architecture générale est présentée sur la figure 8.

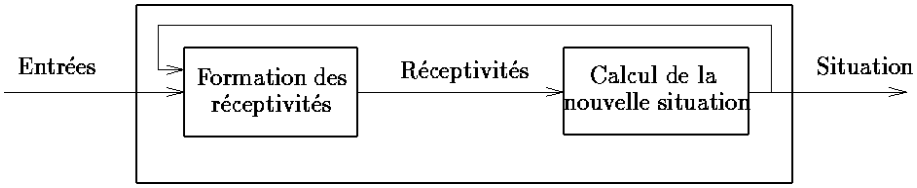

FIG. 8 : Architecture générale SRS

Une situation définit l'état d'activité des étapes du GRAFCET. Le calcul de la nouvelle situation se fait par un ensemble d'équations qui expriment simplement que l'activité de chaque étape dépend de la réceptivité actuelle de certaines transitions et de l'activité, au top d'horloge précédent, de certaines étapes. Les équations sont calculées en parallèle, et les pas de calcul sont rythmés par des tops d'horloge.

L'interprétation ARS est plus compliquée à modéliser. Il faut en effet être capable de déterminer si une situation est stable et être capable d'isoler le système du monde extérieur pendant cette recherche de stabilité. Du point de vue du monde externe (accessible à l'utilisateur), l'acquisition des entrées et l'émission des sorties sont simultanées. Du point de vue du monde interne, on bloque la lecture des entrées et l'écriture des sorties tant qu'une situation stable n'est pas atteinte. Il y a donc deux mondes à synchroniser en fonction du type de situation atteinte.

Le corps de la boucle d'évolution du grafcet est le suivant (en langage commun) :

si situation stable *alors*
    acquisition des entrées
finsi

calcul de la nouvelle situation
calcul de la stabilité
si situation stable *alors*
émission des sorties
finsi

L'architecture générale du programme SIGNAL obtenu est présentée sur la figure 9.

Les calculs sont effectués comme précédemment en parallèle.

Dans les deux cas, ARS et SRS, on modélise également les actions en SIGNAL. On a donc une traduction complète du GRAFCET en langage synchrone. On peut parler alors de langage Grafcet.

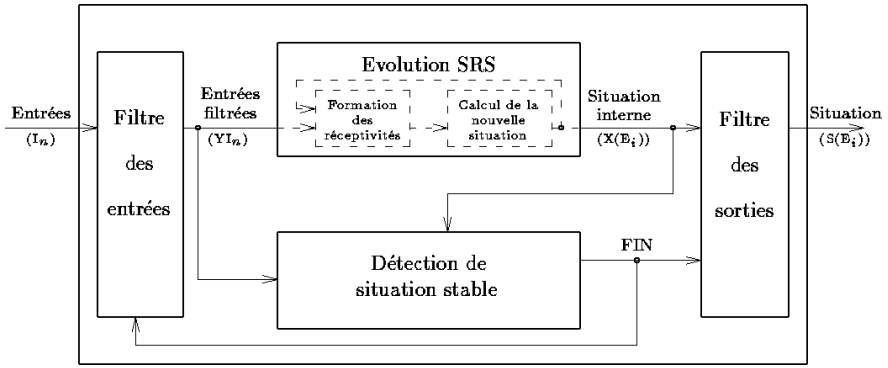

FIG. 9 : Architecture générale ARS

4.4 - La simulation du grafcet

SIG est le compilateur SIGNAL développé à l'IRISA de Rennes (il existe un environnement commercial développé par la société TNI [2]). Il génère une procédure d'initialisation du grafcet, une procédure d'évolution, une fonction de lecture dans un fichier pour chaque entrée et une fonction d'écriture dans un fichier pour chaque sortie.

SIMGRAF permet une simulation graphique et interactive du grafcet, déconnectée de toute partie opérative et sans prise en compte du temps physique.

2 Technique Nouvelle Informatique (TNI) - rue Descartes - Technopôle Brest Iroise - Site de la Pointe du Diable - 29200 Brest

4.5 - L'exécution du grafcet

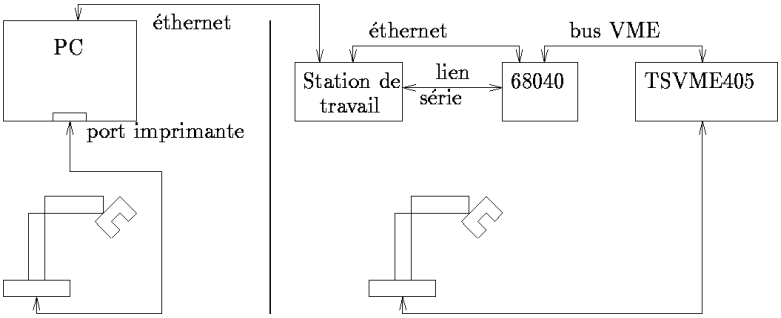

FIG. 10 : Environnements matériels

Dans ce cadre, nous avons expérimenté deux environnements matériels :

- un micro-ordinateur de type PC sous différents systèmes d'exploitation (depuis MS-DOS 5.0 jusqu'à Windows NT),
- une carte 68040 exécutant le système d'exploitation temps-réel VxWORKS, reliée à une station de travail sous UNIX via Ethernet et une liaison série [9].

La partie opérative est un bras manipulateur FischerTechnik. Elle est connectée au PC via le port imprimante ou à la carte 68040 via un bus VME et une carte de gestion d'entrées/sorties logiques. La figure 10 décrit sommairement les deux environnements matériels.

Notre chaîne est entièrement portable sur ordinateur de type PC sous Windows 95, ou sur station de travail SUN, liée à un rack VME. Certaine parties de la chaîne de développement peuvent être portées sur des PC sous d'autres systèmes d'exploitation.

5 - CONCLUSION

Nos expériences sur l'exemple choisi sont conformes aux résultats attendus :

- l'algorithme SRS fonctionne bien tant que les capteurs associés à a et b (cf. figure 2) ne sont pas enfoncés simultanément. Mais dans le cas d'un dysfonctionnement de ces capteurs les amenant à être vrais tous les deux en même temps, alors le comportement de la partie commande devient instable entraînant des mouvements incohérents de la partie opérative.

- l'algorithme ARS élimine les commandes dans les situations instables, ce qui a l'avantage de ne pas solliciter la partie opérative. Nous constatons effectivement aucun mouvement de la partie opérative, lors d'une demande de début de cycle. Pour pouvoir reprendre un cycle normal il faut d'abord régler le problème à l'origine de l'instabilité, puis relancer le logiciel.

L'effet concret de l'algorithmique choisie ARS ou SRS est assez spectaculaire. Or ces deux interprétations aussi divergentes sont en accord avec les règles d'évolution et les postulats temporels, malgré leur apparence de rigueur formelle.

Nos réalisations démontrent la différence d'évolution du même grafcet selon l'algorithme employé. Nous y trouvons un grand intérêt pédagogique pour nos étudiants.

Les API respectent en général l'algorithme SRS. Cependant quelques uns ont un comportement intermédiaire entre les deux algorithmes, ce qui surprend toujours lors des premières utilisations.

Nos travaux sur la preuve de programmes ont pour but de déceler les dysfonctionnements potentiels des parties commandes avant même leur implantation, en repérant les bouclages infinis, les situations jamais atteintes, etc. Dans le futur nous souhaitons réaliser un post-processeur pour les automates TSX07 de la société Schneider-Télémécanique permettant de compiler les fichiers grafcets générés par EG7+ en fichiers ASCII assimilables par le langage PL-707.

L'extension des réseaux de communications informatiques et des outils de communications universels que sont les navigateurs de réseaux et les chargements FTP, permet d'envisager le développement de travaux d'expertises à distance sur des grafcets. Les outils de preuve, la simulation et les tests pourraient être mis à disposition d'utilisateurs.

Jean VAREILLE*, Mireille ARNOUX**
* I.U.T. dépt G.M.P. ** Département d'Informatique,
Université de Bretagne Occidentale,
EA 2215, équipe LIMI
(Langages et Interfaces pour Machines Intelligentes),
6 av. V. Le Gorgeu, BP 809, 29285 Brest cedex
e-mail : arnoux@univ-brest.fr
vareille@univ-brest.fr
http://doelan-gw.univ-brest.fr/8080/limi

BIBLIOGRAPHIE

[1] BOUTEILLE N., BRARD P., COLOMBARI G., COTAINA N. et RICHET D. : Le grafcet. In : *Cepadues Editions.*

[2] IEC, IEC 848 : *Preparation of function charts for control systems.*

[3] LEGUERNIC P., Gautier T. et BESNARD L. : Functional programming languages and computer architectures. In : *SIGNAL a declarative language for synchronous programming of real time systems.*- LNCS 274. Springer-Verlag, 1987.

[4] LEPARC P. : Apports de la méthodologie synchrone pour la définition et l'utilisation du langage grafcet. In : *Thèse de doctorat, LIMI, Brest.* - Universite de Rennes I, janvier 1994.

[5] LESAGE J.J. : Epreuve d'automatismes industriels du concours externe de l'agrégation de génie mécanique. In : *Rapport de jury d'agrégation, Centre National de Documentation Pédagogique.*

[6] L'HER D., LEPARC P. et MARCÉ L. : Modélisation et vérification du grafcet. In : *Modélisation des systèmes réactifs.* - Brest, France, mars 1996.

[7] MERLAUD C. : Construire un grafcet : une question de comportements... In : *Technologies et formations.* - N°47. Tâches et comportements. In : *Technologies et formations.* - N°48. Des grafcets en Prépas. In : *Technologies et formations.* - N°69 & N°71.

[8] NFC-82 - Norme NF C 03-190. Schémas, diagrammes, tableaux : diagramme fonctionnel grafcet pour la description des systèmes logiques de commande.

[9] SCHARBARG J.L., BATTUT J. et MARCÉ L. : Un automate programmable fondé sur l'approche synchrone du grafcet. In : *Modélisation des systèmes réactifs.* - Brest, France, mars 1996.