Rapport d'Activité
GDR AUTOMATIQUE, pôle C2A

C. André, D. Gaffé et Marie-Agnès Péraldi

Laboratoire Informatique, Signaux, Systèmes (I3S)
Projet SPORTS, équipe ALM
Université de Nice-Sophia Antipolis / CNRS
e-mail: andre@cma.cma.fr gaffe@mimosa.unice.fr map@mimosa.unice.fr

15 juillet 1994

Résumé

Ce rapport présente les résultats obtenus par le projet SPORTS[1] *dans le cadre des recherches sur les relations asynchrones / synchrones. Les domaines d'application visés sont essentiellement les Systèmes Automatisés de Production. Les études portent sur l'usage conjoint du langage synchrone* ESTEREL *et du modèle* GRAFCET. *Les contributions se situent au niveau modélisation, sémantique, compilation et développement d'environnement de programmation synchrone.*

Mots clefs : ESTEREL, GRAFCET, systèmes réactifs temps-réel, environnements de programmation, programmation synchrone, Systèmes Automatisés de Production.

1. SPORTS (Synchronous Programming Of Real-Time Systems), ALM (Architectures Logicielles et Matérielles)

1 Introduction

Le GRAFCET est largement répandu en Automatique pour spécifier le comportement attendu de la Partie Commande des *Systèmes Automatisés de Production* (SAP). Il fait l'objet de normes francaise [AFN82] et internationale sous l'appellation "Function Charts" [IEC88].

ESTEREL [BG92] est un langage synchrone de création plus récente. Un compilateur industriel et divers outils de développement sont actuellement disponibles. L'objectif de ce langage est de programmer des applications de type "réactif" (systèmes directement couplés à leur environnement).

Nous préconisons une approche utilisant à la fois le GRAFCET et le langage ESTEREL. Ceci permet de concilier les points de vues "automatisme industriel" et "programmation réactive". Il apparaît que certains comportements s'expriment plus facilement dans l'un des deux formalismes. La coopération GRAFCET-ESTEREL, que nous avons développée, autorise ces choix syntaxiques tout en s'appuyant sur une sémantique bien établie (celle d'ESTEREL). Cette solution va dans le sens d'une programmation plus conviviale (multi-formalisme) et plus sûre (sémantique mathématique), avec possibiltés de vérifications formelles.

Domaine d'application

Nous avons essentiellement visé des applications aux SAP. Dans ce domaine, les approches réactives ESTEREL-GRAFCET apportent des contributions intéressantes, en particulier pour :

1. la *commande locale*,
2. la *surveillance du système*,
3. la *caractérisation et le contrôle des modes de marche*.

En effet, au niveau de la commande locale, les actions sont souvent de type réflexe, donc très réactives et soumises à des contraintes temporelles fortes. La surveillance fait intervenir la détection d'anomalies, que ce soient des écarts au comportement attendu, des incohérences entre la commande et l'état du système, ou bien de signaux en provenance de capteurs de surveillance. La prise en compte des fautes présente donc un caractère réactif. Le déclenchement des opérations de diagnostic, ainsi que la mise en œuvre des décisions (réessais, solution de substitution, réparations, ...) peuvent induire des comportements réactifs complexes. En ce qui concerne le dernier point, il suffit de constater que les changements de mode de fonctionnement se font souvent sur occurrence d'un événement donné. Cet événement peut être émis par l'opérateur ou peut émaner du système de surveillance. La commutation de mode provoque généralement des réactions multiples et coordonnées entre les divers sous-systèmes contrôlés.

Les utilisations des approches synchrones dans les SAP ont été introduites dans une communication invitée [AP93c]. Plus de détails sont disponibles dans le rapport d'activité [AAD+93] d'un projet inter-laboratoires [2] (soutien des programmes Sciences pour l'Ingénieur de la DRED).

Plan

Dans ce rapport, nous allons reprendre les points suivants:

1. utilisation d'ESTEREL en surveillance,

2. LAIL de Lille, LAAS de Toulouse, I3S de Nice-Sophia Antipolis

2. intérêt du multi-formalisme ESTEREL-GRAFCET,
3. sémantique “à la ESTEREL” pour le GRAFCET,
4. compilation du GRAFCET,
5. contribution à l’environnement de développement ESTEREL-GRAFCET.

2 Utilisation d’Esterel en surveillance

La commande et la surveillance des *systèmes à événements discrets* est généralement une application réactive dont les réactions sont déclenchées par des occurrences d’événements sporadiques. Le langage ESTEREL est particulièrement bien adapté à la programmation de ce type d’applications. Le caractère impératif du langage permet d’exprimer des commandes surveillées assez sophistiquées. Les instructions réactives du langage sont largement utilisées. Nous allons présenter deux instances de ce problème : celui de la détection d’une faute et celui d’une commande avec reprise en cas d’erreur.

2.1 Module générique de détection de “timeout”

Généralement les actions doivent être exécutées avant une échéance donnée. Le module suivant exprime les contraintes temporelles qui existent habituellement au niveau de l’exécution d’une action. Pour surveiller temporellement une action “`Action`” il suffit de mettre les modules ESTEREL en parallèle : “`exec Action || run Timewatching`”.

Spécification des contraintes temporelles

- événement “`DoAction`” (`DoA`, en abrégé) : ordre d’exécuter l’action,
- événement “`InitialStateLeft`” (abrév. : `ISL`) : se réalise quand l’exécution commence effectivement à se manifester,
- événement “`FinalStateReached`” (abrév. : `FSR`) : se réalise quand l’exécution termine avec succès.

En ESTEREL, les événements sont représentés par des “signaux (`signal`)”.

Contraintes

- date(`ISL`) - date(`DoA`) $\leq$ `INERTIA` $\times$ unités de temps (`Top`)
- date(`FSR`) - date(`ISL`) $\leq$ `DEADLINE` $\times$ unités de temps
- Erreurs :
  - Si l’action ne démarre pas suffisamment vite, on émet `Error` valué par `Stuck`,
  - Si l’action a démarré mais ne termine pas en temps utile, on émet `Error` valué par `Lost`,
- en cas de succès, on émet `Done`.

module esterel correspondant:

```
module Timewatching:
input DoA, ISL, FSR, Top;
output Error: integer, Done;

constant INERTIA, DEADLINE: integer;
constant Stuck, Lost: integer;

    await immediate DoA;
    do
        await INERTIA Top;
        emit Error(Stuck)
    watching ISL
    timeout
        do
            await DEADLINE Top;
            emit Error(Lost)
        watching FSR
        timeout
            emit Done
        end
    end
end module
```

Bien sûr, les constantes et les signaux effectivement utilisés doivent renommer ceux du module générique. La détection d'autres types d'erreurs, en particulier celles provenant de capteurs, peuvent être détectées de façon analogue par des modules ESTEREL.

Le module `Timewatching` exploite la forme forte de la préemption (`do` ...`watching` ...`end`). Dans ce cas les signaux qui provoquent la préemption doivent être d'origine extérieure au module. La forme faible de la préemption [Ber93b] autorise la levée d'exception à l'intérieur d'un module. Elle est exploitée dans des modules de surveillance avec reprise.

2.2 Surveillance avec reprise

Nous proposons ci-dessous, un squelette de programme qui réalise la commande surveillée suivante:

- la commande (normale) est exprimée par le module `NormalCntrl`. En cas de terminaison normale, le signal `ACK` est émis.
- en cas de détection d'anomalie, la partie détection qui est non explicitée dans ce squelette, émet soit le signal `recov` (erreur récupérable), soit le signal `unrecov` (erreur irrécupérable).
- si l'erreur est récupérable, on peut faire jusqu'à `ATTEMPTS` tentatives pour exécuter le contrôle normal. Dépassé ce seuil on lance un autre contrôle (`AltControl`) qui correspond à un fonctionnement dégradé.
- si l'erreur n'est pas récupérable, on abandonne la commande et on émet `NACK` valué par un numéro d'erreur.

```
module MonitoredAction:
input RQ;
```

```
output ACK, NACK: integer;

constant ATTEMPTS: integer;

signal RQ_F in
    trap T_ACTIVE in
        run Filter % in: RQ, out: RQ_F
        ||
        trap T_DONE in
            trap T_FAIL: integer in
                signal unrecov: integer, recov in
                    trap T_ALT_CNTRL in
                        repeat ATTEMPTS times
                            trap T_CNTRL in
                                run NormalCntrl;
                                exit T_DONE
                                ||
                                % put here detection - diagnosis
                                % may exit either T_CNTRL
                                % or T_FAIL, according to the fault
                            end trap % T_CNTRL
                        end repeat;
                        exit T_ALT_CNTRL % too many attempts
                    handle T_ALT_CNTRL do
                        run AltControl
                    end trap % T_ALT_CNTRL
                end signal % recovery
            handle T_FAIL do
                emit NACK(?? T_FAIL)
            end trap % T_FAIL
        handle T_DONE do
            emit ACK
        end trap; % T_DONE
        exit T_ACTIVE
    end trap % T_ACTIVE
end signal
end module
```

2.3 Publications

Outre la communication à CompEuro'93 [AP93c] et le rapport DRED [AAD+93] déjà mentionnés, signalons une communication à la conférence internationale on Systems, Man and Cybernetics [AEGP93] et un rapport technique détaillant l'usage des structures de contrôle ESTEREL en programmation de commande/contrôle de SAP [AP93b].

3 Multi-formalisme ESTEREL-GRAFCET

Les modules `NormalCntrl` et `AltControl` du programme précédent expriment les comportement attendus du système. Lorsqu'il s'agit de systèmes séquentiels industriels, le GRAFCET est préférable au langage ESTEREL. Deux raisons militent

FIG. 1 - *Grafcet du Contrôle Normal*

en faveur de ce choix:

- le GRAFCET est un langage "métier" pour ce type d'applications,
- le GRAFCET exprime plus simplement qu'ESTEREL certains évolutions parallèles qui ne sont pas strictement "fork–join".

Le grafcet Fig. 1 illustre le second argument : les deux branches d'une évolution parallèle sont synchronisées (l'`Usinage 2` ne peut être exécuté que si les opérations `Dégraissage` et `ChangementOutil` sont terminées).

Dans la description multi-formalisme, ESTEREL exprime les comportements réactifs de supervision, alors que le GRAFCET décrit essentiellement la commande (normale ou de substitution). Les utilisateurs du GRAFCET ont étendu le modèle en introduisant des hiérarchies de grafcets, ainsi que la notion de forçage d'un grafcet par un autre. La solution que nous proposons a deux avantages :

- elle ne fait pas appel à ces extensions du GRAFCET dont la sémantique n'est pas encore stabilisée,
- les instructions réactives d'ESTEREL permettent d'exprimer des comportements complexes, et ceci sans ambiguïté.

Un autre exemple d'usage conjoint ESTEREL-GRAFCET est décrit dans le rapport [AG94c] et a été présenté au colloque AGI'94 [AG94a].

4 Une sémantique pour le GRAFCET

Pour bénéficier à la fois de la convivialité du GRAFCET et des fondements théoriques des langages, on a intérêt à traduire le GRAFCET en un langage ayant une sémantique rigoureuse. Cette démarche a déjà été suivie par différents auteurs (GRAFCET vers SIGNAL [ML92]), (GRAFCET vers ESTEREL [AP93a]), (GRAFCET vers

ELECTRE [RR94]). Nous avons choisi une traduction GRAFCET vers ESTEREL pour plusieurs raisons :

- Les systèmes considérés sont *pilotés par occurrences d'événements* (event-driven systems). Ces occurrences s'expriment en termes de signaux d'entrée en ESTEREL et d'événements (fronts) dans les réceptivités GRAFCET.
- Une deuxième raison est que différents modes d'évolution ont été proposés pour le GRAFCET. Le plus complet est le mode dit "Avec Recherche de Stabilité" (ARS). Ce mode introduit des *évolutions internes instantanées* s'apparentant à la sémantique comportementale d'ESTEREL (système de réécritures multiples instantanées).
- Un dernier argument important est la *maîtrise du langage et de sa compilation.* ESTEREL a été créé et développé au CMA (Centre de Mathématiques Appliquées de l'École des Mines de Paris), à Sophia Antipolis. Nous travaillons en collaboration directe avec le CMA et nous avons accès aux versions les plus récentes du langage.

La traduction a fait apparaître des insuffisances sémantiques au niveau du GRAFCET, essentiellement en ce qui concerne la dynamique des états instables. Des choix ont dû être faits pour lever les ambiguïtés et assurer la cohérence avec la sémantique d'ESTEREL.

Un article à paraître dans APII [AG94b] explique les difficultés rencontrées. Les notions d'événements et conditions, ainsi que la caractérisation des évolutions internes ont été reformulées sous les hypothèses de synchronisme introduites par ESTEREL. Un algorithme pour calculer la situation suivante après évolutions internes a été proposé. Il fait intervenir des calcul de points fixes [And94]. Cet algorithme admet comme cas particulier celui qui était utilisé classiquement. Il est pleinement compatible avec la notion de réaction définie par le langage ESTEREL. Ceci garantit la cohérence sémantique des approches mixtes ESTEREL-GRAFCET.

5 Compilation de GRAFCET

Pour traduire le GRAFCET en ESTEREL, deux types de solutions ont été explorés :

1. approche structurelle,
2. approche équationnelle.

La première solution, a consisté à identifier dans un grafcet, des structures (séquence, alternative, itération, parallélisme, ...) qui sont directement supportées par des constructeurs du langage ESTEREL. Cette approche convient pour des grafcets "bien structurés", mais est insuffisante dans le cas général. En particulier certaines synchronisations (voir Fig. 1), ainsi que les évolutions "pipe-line", induisent des traductions lourdes et difficilement généralisables. La prise en compte des états initiaux est également un problème délicat.

La seconde solution s'éloigne délibérément de la structure du GRAFCET. Elle conçoit le modèle comme un ensemble d'objets réactifs communicants. A chaque étape est associée une entité réactive. Les étapes interagissent par dialogues instantanés. Les dialogues (communications synchrones) expriment la topologie du graphe ainsi que le mode d'évolution retenu (évolutions avec Recherche de Stabilité). Notons que les transitions n'apparaissent plus en tant qu'entités distinctes mais sont regroupées au sein d'un unique "*moteur de transition*".

D'un point de vue sémantique, un grafcet est considéré comme un transformateur déterministe d'histoires d'entrée en histoire de sortie. Chaque "étape" peut être dans un état actif ou inactif. L'ensemble des étapes actives caractérise la "situation" courante du grafcet (état global). Pour un environnement donné, une situation nouvelle *unique* doit être trouvée par dialogues instantanés. Le problème se ramène donc à la résolution d'un système d'équations de communication, dont LA solution est l'ensemble des signaux présents dans l'instant.

Une solution approchée est fournie par le compilateur ESTEREL, grâce à un algorithme efficace [Gon88]. Cet algorithme rejète les programmes incorrects (en particulier ceux n'admettant pas de solution unique). Toutefois il peut rejeter à tort, certains programmes corrects, par présomption de cycle de causalité (théorie des potentiels de Gonthier). Les grafcets, qui utilisent plutôt les conditions booléennes que les événements, induisent de nombreux rejets de la part du compilateur ESTEREL. La traduction que nous proposons a sû contourner ce problème et de plus, rejète les grafcets instables (cycle d'évolution instantanée).

FIG. 2 - *Chaîne de compilation de Grafcet et d'Esterel*

La compilation du GRAFCET s'inclut donc dans celle d' ESTEREL [CIS88] comme le montre le schéma de la figure 2.

6 Environnement logiciel

Par l'intermédiaire du langage ESTEREL, le GRAFCET peut bénéficier des environnements logiciels développés pour ESTEREL. Ceci participe à l'intégration du GRAFCET dans la *plateforme commune synchrone*. Cet objectif rejoint celui d'autres recherches faites au sein de C2A : intégration du GRAFCET via SIGNAL [Lep94], via ARGOS et LUSTRE, via ELECTRE [RR94].

ESTEREL étant particulièrement adapté à la programmation des systèmes pilotés par événements, il a été doté de simulateurs événementiels interactifs. XSIMUL (extension graphique du C-SIMUL [CIS88]) est une application conviviale, tournant sous X11, et générée automatiquement à partir du source ESTEREL (voir chaîne de compilation Fig. 2). Cette possibilité a été étendue au GRAFCET. On peut simuler les comportements de programmes mixtes ESTEREL-GRAFCET à l'aide de panneaux de contrôle.

Une aide très appréciée des concepteurs est la capacité de visualiser dynamiquement l'état interne des programmes (remontée dans le source en ESTEREL). En

GRAFCET, l'équivalent de la remontée dans le source est une visualisation de la situation courante du grafcet. La figure. 3 est une copie d'écran d'une session de simulation mixte.

Les logiciels relatifs au GRAFCET (Éditeur, générateur de simulateur, exécution dynamique) ont été développés au sein de l'équipe SPORTS [Pas94]. Ils utilisent largement les possibilités graphiques de Tcl/Tk.

FIG. 3 - *Écran de simulation*

ESTEREL offre de plus la possibilité de générer du code pour certains outils de preuves formelles sur les automates (citons Auto [dV89]). Avec le GRAFCET, on peut par exemple dénombrer les actions jamais émises, prouver l'équivalence comportementale de deux grafcets ou montrer l'incidence sur le système complet de la panne de tel ou tel capteur.

7 Conclusion

La coopération GRAFCET-ESTEREL présentée dans ce rapport est intéressante à plus d'un titre. Les deux approches sont syntaxiquement complémentaires (graphique/textuelle) et nous avons pu leur donner une unité sémantique (basée sur celle d'ESTEREL). Dans ces conditions il est possible *d'exprimer les comportements attendus dans le formalisme le plus adéquat, sans perte de rigueur.* Ces représentations mixtes peuvent être compilées en automates à états finis déterministes. Il en résulte des possibilités d'implantation efficace et de connexion aux outils de preuves basés sur les automates, répondant ainsi à la demande de programmation sûre des systèmes "temps-réel".

L'intégration du GRAFCET dans les environnements de développement synchrone devrait rendre de précieux services dans la mise au point des applications. Le simulateur mis au point, qui permet interactions et remontées dans le source aussi bien en ESTEREL qu'en GRAFCET est un outil important pour les développements à venir. La mise au point des applications proposées au niveau des Systèmes Automatisés de Production devrait s'en trouver facilitée.

D'un *point de vue plus fondamental*, la traduction GRAFCET vers ESTEREL a nécessité de repenser la sémantique du GRAFCET, avec la rigueur qui caractérise les langages synchrones. Une traduction directe du GRAFCET en OC, basée sur cette sémantique, devrait permettre de dépasser certaines limitations dues aux compilateurs ESTEREL actuels, en tirant un meilleur parti des spécificités du GRAFCET. Notons que la version actuelle permet déjà de rejeter tout grafcet *instable* dès la compilation.

En fait, nous visons, à terme, un cadre unifié pour étudier des *processus réactifs communicants* introduits dans [Ber93a]. Nous étudions actuellement les *objects réactifs* qui encapsulent données, traitements et comportements. Dans ce cadre, le GRAFCET devrait devenir un moyen de description, à part entière, pour les objets réactifs.

Références

[AAD+93] C. André, C. Ausfelder, D. Delfieu, S. Elkhattabi, D. Gaffé, J.C. Gentina, L. Kermad, J.P. Maik, R. Moisand, M.A. Péraldi, and A.E.K. Sahraoui. Approches asynchrones / synchrones des systèmes flexibles de production. Technical Report DRED Soutien des Programmes SPI, I3S/LAAS/LAIL, Sophia-Antipolis, France, Septembre 1993. Rapport Final, I3S TR 93–01.

[AEGP93] C. André, S. Elkhattabi, J.C. Gentina, and M.A. Péraldi. Esterel programming in fms. In *Int. Conf. on Systems, Man and Cybernetics*, pages 252–257, Le Touquet (France), 17–20 October 1993. IEEE/SMC.

[AFN82] AFNOR. *Diagramme fonctionnel* GRAFCET *pour la description des systèmes logiques de commande*, juin 1982. Norme Française, NF C 03 190.

[AG94a] C. André and D. Gaffé. Coopération grafcet/estérel. In *Colloque AGI'94*, pages 221–224, Poitiers, Juin 1994. Association AGI.

[AG94b] C. André and D. Gaffé. Evénements et conditions en GRAFCET. Technical Report RR 94–06, I3S, Sophia-Antipolis, France, Février 1994.

[AG94c] C. André and D. Gaffé. Modélisation et coopération GRAFCET/ESTEREL. Technical Report RR 94–27, I3S, Sophia-Antipolis, France, Mai 1994.

[And94] C. André. Une sémantique pour le GRAFCET. Technical Report RR 94–07, I3S, Sophia-Antipolis, France, Février 1994.

[AP93a] C. André and M-A. Péraldi. GRAFCET and synchronous languages. *A.P.I.I.*, 27(1):95–105, 1993.

[AP93b] C. André and M-A. Péraldi. Synchronous approach to industrial process control. Technical Report RR 93–10, I3S, Sophia-Antipolis, France, February 1993.

[AP93c] C. André and M-A. Péraldi. Synchronous programming: Introduction and application to industrial process control. In 7^{th} *Annual European Computer Conference*, pages 461–470, Evry (France), May 1993. IEEE.

[Ber93a] G. Berry. Communicating reactive processes. In *Proc. 20th ACM Conf. on Principles of Programming Languages, Charleston, Virginia*, 1993.

[Ber93b] G. Berry. Preemption in concurrent systems. In W. Brauer, editor, *FSTTCS 93*, volume 761 of *Lecture Notes in Computer Science*, pages 72–93. Springer-Verlag, 1993.

[BG92] G. Berry and G. Gonthier. The ESTEREL synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.

[CIS88] CISI Ingénierie, Valbonne (FRANCE). ESTEREL C *Simulation Manual*, 1988.

[dV89] R. de Simone and D. Vergamini. Aboard AUTO. Technical Report 111, INRIA, October 1989.

[Gon88] G. Gonthier. *Sémantiques et modèles d'exécution des langages réactifs synchrones. Application à* ESTEREL. PhD thesis, Université de Paris-sud, Centre d'Orsay, Mars 1988.

[IEC88] IEC, Genève (CH). *Preparation of functions charts for control systems*, december 1988. International standard IEC 848.

[Lep94] P. Leparc. *Apports de la méthodologie synchrone pour la définition et l'utilisation du langage Grafcet.* PhD thesis, Université de Rennes 1, Janvier 1994.

[ML92] L. Marcé and P. Le Parc. Modélisation de la sémantique du GRAFCET à l'aide de processus synchrones. In *Conférence Grafcet'92*, pages 101–110, Paris, Mars 1992. Afcet.

[Pas94] L. Pastorelli. Machine d'exécution pour le formalisme synchrone GRAFCET. Technical report, Ecole Doctorale SPI – DEA Informatique, June 1994. Rapport de stage de DEA.

[RR94] O. Roux and V. Rusu. Etude de la traduction du GRAFCET en langage réactif asynchrone ELECTRE. *à paraître*, 1994.