GRAFCET ET LANGAGES SYNCHRONES

ANDRE Charles, PERALDI Marie-Agnès
Université de Nice-Sophia Antipolis / CNRS
LASSY équipe de l'I3S
41 Boulevard Napoléon III
F 06041 NICE cedex

RESUME:

Le GRAFCET est un modèle de représentation des comportements attendus des systèmes logiques. Les langages synchrones sont dédiés à la programmation des systèmes réactifs. Cet article compare et rapproche ces deux approches.
La comparaison porte sur les concepts sous-jacents. Le synchronisme est le dénominateur commun. Des traductions de GRAFCET vers ESTEREL et LUSTRE sont ensuite proposées. Líexistence de telles traductions permet au GRAFCET de bénéficier des environnements logiciels développés pour les langages synchrones. En conclusion, nous soulignons les perspectives díutilisation conjointes du GRAFCET et des langages synchrones.

Mots-clés: GRAFCET, langages synchrones, ESTEREL, LUSTRE

ABSTRACT:

The GRAFCET is a convenient model for the behaviour of control systems. Synchronous languages are devoted to reactive system programming. In this paper, we compare and bring these two approaches together.
Underlying concepts are the basis for the comparison. Synchronism appears to be the commun denominator. Translations from GRAFCET into ESTEREL and LUSTRE are proposed. Such translations allow the GRAFCET to use the software (compilers and analyzers) developed for synchronous languages. Finally, we point out the future prospects of a joint use of the GRAFCET and the synchronous languages.

Keywords: GRAFCET (Function Charts), Synchronous Languages, ESTEREL, LUSTRE

I. INTRODUCTION

Le **GRAFCET** est un modèle formel de représentation des comportements attendus des systèmes logiques. Il est largement répandu dans l'industrie et l'enseignement. Il fait l'objet de normes française (GRAFCET) et européenne (Function Charts for Control Systems).
Les langages synchrones sont apparus récemment. Ils ont été conçus pour faciliter la programmation des systèmes réactifs. Certains sont impératifs (ESTEREL) et se prêtent bien aux approches "événementielles", d'autres sont déclaratifs (LUSTRE, SIGNAL) et plus adaptés aux systèmes à "flots de données".

Le modèle GRAFCET et les langages synchrones ont de nombreux **points communs**.

ï†Les **domaines d'application** se recouvrent: les systèmes logiques sont des systèmes réactifs "temps-réel" particuliers. A priori, les langages synchrones s'adressent à une classe de systèmes plus large que le GRAFCET.

ï†Les **utilisations** sont comparables: les deux types de formalisme peuvent être employés en tant que langage de commande ou en tant que modèle comportemental. Un GRAFCET qui représente un automatisme logique, selon le point de vue "partie commande", est assimilable à un programme de commande de la partie opérative. Un programme en langage synchrone peut être considéré comme une description du comportement attendu d'un système réactif, ceci indépendamment de son implémentation.

Dans cet article, après une brève présentation du GRAFCET et de certains langages synchrones, nous proposons de les comparer afin de dégager les caractéristiques communes fondamentales. Ces caractéristiques sont souvent cachées par les formalismes. Nous mettrons également en évidence les différences. Cette étude devrait conduire à une meilleure compréhension des deux approches. Dans une seconde partie nous suggèrons des "réécritures" GRAFCET vers langages synchrones. La troisième partie expose les avantages qu'on peut tirer de telles traductions: d'une part l'utilisation des environnements logiciels des langages synchrones pour le GRAFCET et d'autre part l'usage du GRAFCET comme langage d'entrée complémentaire des langages synchrones. Nous conclurons en dégageant des perspectives aussi bien du point de vue recherche que du point de vue des développements industriels.

II. RAPPELS SUR LE GRAFCET ET LES LANGAGES SYNCHRONES

1 GRAFCET

La norme [AFNOR] constitue notre référence. Des ouvrages à vocation pédagogique ont montré les possibilités du modèle [GREPA, TheTou]. Des réflexions sont en cours sur des extensions du modèle [Groupe AFCET- sous groupe GRAFCET].
Nous nous limitons à rappeler les règles d'évolutions ainsi qu'un postulat et une note, car nous y faisons référence dans la suite.

Règle 1: situation initiale (G1)
La situation initiale du GRAFCET caractérise le comportement initial de la partie Commande vis-à-vis de sa partie Opérative et correspond aux étapes actives au début du fonctionnement de la partie Commande.

Règle 2: franchissement d'une transition (G2)
Une transition est **validée** si et seulement si toutes les étapes immédiatement précédentes reliées à cette transition sont actives.
Une transition est **franchissable** si et seulement si elle est **validée** et la **réceptivité** associée à cette transition est vraie.
Une transition qui devient franchissable est **obligatoirement** franchie.

Règle 3: évolution des étapes actives (G3)
Le franchissement d'une transition provoque **simultanément**
ï†la désactivation de toutes les étapes immédiatement précédentes,
ï†l'activation de toutes les étapes immédiatement suivantes.

Règle 4: évolutions simultanées (G4)
Plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle 5: activation et désactivation simultanées (G5)
Si au cours du fonctionnement de l'automatisme une même étape doit être simultanément activée et désactivée, elle reste active.

Postulat: simultanéité (GP)
Le modèle GRAFCET exclut formellement la simultanéité d'occurrence de deux événements externes non corrélés.

Note: durée des transitions (GD)

La durée de franchissement d'une transition est aussi petite que l'on veut sans être nulle.

2 LANGAGES SYNCHRONES

Ces langages, récemment introduits, permettent de programmer des applications à la fois parallèles et déterministes, ceci de façon élégante, sûre et efficace. Un numéro spécial des Proceedings of the IEEE vient de leur être consacré [IEEE].

A. ESTEREL [BouSim]

Ce langage a été proposé, entre autre, pour répondre au défit posé par la programmation "temps-réel". Il a fait des choix qui sont parfois diamétralement opposés à ceux retenus dans les approches classiques. Il est basé sur les hypothèses d'exécution infiniment rapide des actions internes et de diffusion instantanée des informations. Le langage est de style impératif: il possède les structures de contrôle classiques incluant le parallélisme, un traitement des exceptions et des instructions "temporelles" originales et puissantes.
ESTEREL est particulièrement apte à programmer les systèmes à événements discrets qui doivent réagir à des séquences de stimuli.

B. LUSTRE [HCRP]

Ce langage est de style déclaratif. On peut concevoir un programme LUSTRE comme un réseau d'opérateurs sur lequel circulent des flots de données. Le caractère synchrone apparait dans l'hypothèse que les opérateurs sont synchrones et font leurs calculs instantanément.
La caractéristique flot de données fait que LUSTRE est plutôt adapté aux systèmes nécessitant des calculs en temps-réel.

III. COMPARAISON

Les comparaisons qui suivent portent sur le modèle GRAFCET et les langages synchrones au niveau sémantique uniquement.

1 TEMPS

Les systèmes réactifs doivent interagir en permanence avec leur environnement physique et ceci à un rythme imposé par cet environnement. Généralement, les contrôleurs de systèmes réactifs doivent réagir "assez vite" pour ne pas perdre d'informations en provenance de l'extérieur et pour présenter les réactions appropriées en temps utile. Il s'agit là de contraintes classiques pour les systèmes dits **temps-réel**.
Le GRAFCET, qui est capable de représenter de tels comportements pour les systèmes logiques, admet l'existence d'un temps physique extérieur au modèle. Les règles d'évolution supposent, implicitement, une référence temporelle globale. Un postulat dit pour le GRAFCET que l'on peut découper le temps en intervalles "aussi petits que l'on veut". Ceci semblerait indiquer qu'on ait besoin d'un temps continu, dans ce cas la notion de simultanéité serait à définir proprement. Ce postulat pourrait être à l'origine de difficultés d'interprétation de certains comportements temporels du GRAFCET. Pour ESTEREL on peut donner une sémantique dénotationnelle avec temps continu, mais ceci est fait avec beaucoup de précautions [Gonthier].
En fait, ce qui est important pour les systèmes à événements discrets considérés, c'est de pouvoir dire si "quelque chose" se passe avant, simultanément ou après "autre chose". Il faut avoir une échelle de "temps". A priori ce "temps" n'est qu'un **temps logique**. Dans les langages synchrones toute entrée physique répétitive peut être utilisée comme base de temps (notion de temps multiforme). La liaison avec le temps physique se fait, si nécessaire, au niveau de l'interprétation: certaines entrées (signaux en ESTEREL, flots booléens en LUSTRE) matérialisent l'écoulement du temps.

Dans la suite nous considéreront un temps discret.

Les systèmes réactifs sont généralement constitués de plusieurs sous-systèmes interagissants. Pour les décrire il faut absolument un moyen d'exprimer le parallélisme, les synchronisations et les communications. Pour les aspects temps réel, il faut également être capable d'exprimer les préemptions et déroutements d'exécutions.

2 SYNCHRONISME

Lorsqu'on modélise un système par un GRAFCET, ou lorsqu'on programme une application avec un langage synchrone, on exprime de façon formelle les réactions attendues d'un système à des stimuli. La plupart de ces réactions doivent être "immédiates" (type réflexe), d'autres doivent être différées mais en respectant des contraintes temporelles précises.
L'hypothèse de synchronisme, et plus encore celle de synchronisme fort, facilitent l'expression de ces contraintes rapidement inextricables avec les approches asynchrones. L'**hypothèse de synchronisme** est celle qui permet de parler de façon non ambiguë d'instant et de simultanéité. Dans les approches synchrones tous les événements simultanés sont traités globalement, de façon indivisible. Une réaction synchrone engendre, à partir d'un ensemble de données simultanées, un ensemble de résultats simultanés. L'hypothèse de **synchronisme fort** va encore plus loin, elle considère que la réaction est instantanée, si bien que les réponses apparaissent dans le même instant que les stimuli, comme si la réaction était calculée par une machine infiniment rapide.
Le langage ESTEREL fait l'hypothèse du synchronisme fort, ce qui confère à ce langage une grande richesse dans l'expression des contraintes temporelles, une sémantique mathématique élégante, mais aussi certaines caractéristiques qui peuvent surprendre de prime abord.
Un langage synchrone flot de données peut être interprété comme un système d'équations dynamiques. A une séquence de vecteurs d'entrée, un programme synchrone répond de façon synchrone par une séquence de vecteurs de sortie, solution du système d'équations associé.
Par les règles G3 et G4, le GRAFCET se rapproche d'ESTEREL. On doit tout de même noter deux différences. ESTEREL, par son hypothèse de synchronisme fort conçoit très bien qu'on puisse effectuer séquentiellement plusieurs actions dans le même instant (durée nulle). Le GRAFCET, par contre, considère qu'une transition a une durée non nulle (GD). La deuxième différence porte sur les entrées. En ESTEREL on admet la simultanéité de signaux d'entrée (stimuli), alors que le GRAFCET rejète cette hypothèse pour des événements externes non corrélés (GP). Il est aisé dans un langage synchrone d'exprimer que certaines occurrences d'événements sont exclusives, par exemple à l'aide des relations en ESTEREL. A notre sens, le fait de savoir si deux événements externes sont corrélés ou non, relève de l'interprétation et suppose des connaissances sur l'environnement. L'hypothèse d'exclusivité des événements externes non corrélés a été introduite pour exprimer l'asynchronisme des interactions. Il semble que la simultanéité des entrées pourrait être admise en GRAFCET, sans remise en cause fondamentale du modèle.

3 PARALLELISME

En ESTEREL, le parallélisme est explicite: le langage possède l'opérateur || (parallèle). Notons que ce parallélisme est assez restrictif; il est du type "Fork and Join". Le GRAFCET offre la même possibilité par l'emploi de transitions de divergence et de convergence. Toutefois il n'est pas obligatoire d'utiliser les convergences-divergences par paires, comme une programmation très structurée l'imposerait.
Dans les langages flot de données, le parallélisme est implicite, il résulte de l'assemblage de divers blocs (équations en SIGNAL, nœuds en LUSTRE). Il s'agit d'un modèle parallèle à grain fin: les opérateurs sont sensés prendre les données dès qu'elles sont disponibles.

4 HIERACHISATION

Les descriptions hiérarchiques sont un moyen d'aborder la complexité des spécifications. Les langages synchrones offrent des facilités dans ce sens.
En LUSTRE, par exemple, la définition d'un nœud peut faire appel à d'autres nœuds. En ESTEREL, l'unité de programmation est le module. Un module est généralement construit à l'aide d'autres modules.

En GRAFCET, la hiérarchisation n'est pas incluse dans la norme, mais les groupes de réflexion sur ce modèle ont très vite proposé des extensions. La macro-étape est le premier concept, mais il s'agit essentiellement d'une commodité syntaxique. Par contre l'introduction des GRAFCET partiels qui interagissent est d'un apport plus important.

5 INTERRUPTIONS

Nous donnons ici à ce terme un sens très général. Il s'agit de pouvoir arrêter l'exécution d'un ou plusieurs sous-systèmes.
C'est dans les approches impératives que l'on explicite ces notions.
Le langage ESTEREL semble être le plus souple dans ce domaine. Il est possible de programmer aussi bien les **préemptions** que les **déroutements** après exécutions. La sémantique de ces opérations ne pose pas les mêmes difficultés que pour les approches asynchrones, grâce à l'hypothèse de synchronisme fort. La suspension n'est pas une primitive du langage. Sa mise en œuvre en ESTEREL demande actuellement des solutions moins directes.
En GRAFCET (ou tout au moins dans ses extensions), il existe la notion de **forçage**: l'activation d'une étape peut activer tout un GRAFCET partiel; la désactivation d'une étape peut causer l'avortement d'exécutions. La suspension peut-être réalisée par sauvegarde puis restauration ultérieure de situation dans un GRAFCET partiel.
En langages flot de données on peut avoir des suspensions implicites. C'est le cas en LUSTRE lorsqu'on utilise la notion de filtrage: une expression filtrée n'est calculée que lorsque l'horloge associée est vraie.

6 COMMUNICATIONS

Les interactions entre sous-systèmes s'accompagnent généralement d'échanges d'informations. Le langage ESTEREL a été le premier à considérer que ses agents communiquaient par **diffusion instantanée**. Ceci signifie que tous les agents ont exactement la même information et au même instant. Cette hypothèse de visibilité cohérente et délocalisée de l'information est implicitement utilisée dans les autres approches synchrones. En langages flot de données, toute occurrence du nom d'un signal (ou d'une variable) dans son domaine de visibilité, désigne exactement la même information. En GRAFCET, si le nom d'une variable apparaît dans différentes réceptivités, elle a la même valeur dans toutes les réceptivités, à un instant donné.

7 DETERMINISME

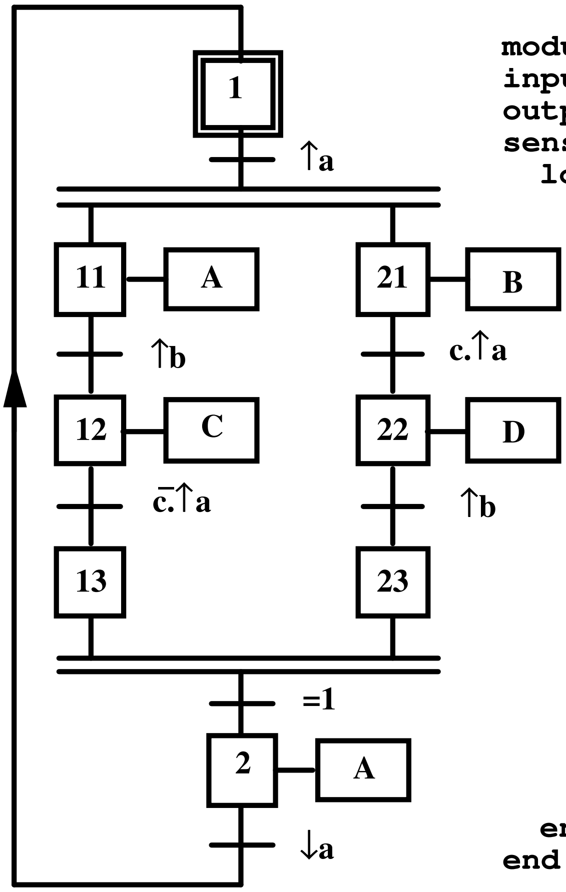
L'hypothèse de synchronisme a permis de définir une composition parallèle simple. Contrairement à son homologue des approches asynchrones, cette composition est déterministe. En ESTEREL, on a fait en sorte que le déterminisme soit assuré pour toutes les instructions, même pour les sélections multiples. Tout programme pouvant conduire à des solutions multiples est rejeté par le compilateur ESTEREL. Donc à une histoire d'entrée correspond une et une seule histoire de sortie. Le comportement du système est entièrement **prévisible**. Notons qu'ESTEREL interdit tout partage de variables entre branches parallèles. Il ne se pose plus ainsi le problème d'accès concurrents à une même variable. Le GRAFCET est lui aussi déterministe. Le franchissement d'une transition produit toujours les mêmes effets. Dans certaines extensions du modèle, il existe des variables qui peuvent être modifiées par des actions associées aux étapes. Il est certain que dans ce cas là on ne peut plus assurer le déterminisme, puisqu'on peut très bien avoir des variables partagées affectées en parallèle. Cette extension ne fait pas partie de la norme. Notons que SIGNAL permet, si on le désire, de faire certains choix non déterministes.

IV. TRADUCTIONS GRAFCET VERS LANGAGES SYNCHRONES

Etant données les similitudes entre GRAFCET et langages synchrones on peut envisager une traduction du GRAFCET vers les langages synchrones. Cette traduction doit respecter la sémantique GRAFCET et être automatisable. Les méthodes de traductions sont très techniques et utilisent largement les spécificités des langages ESTEREL et LUSTRE. Dans la suite, nous ne donnons que les principes de la traduction.

1 ASPECTS SEQUENTIELS

Dans la mesure où le GRAFCET représente des séquences éventuellement en parallèle, ESTEREL parait tout indiqué comme langage cible. La traduction n'est simple que pour des GRAFCET "bien structurés" et qui ont des réceptivités du type événementiel (considérés comme signaux d'entrée ESTEREL). La figure 1 contient une traduction partielle díun tel GRAFCET.




```
module Grafcet1:
input Upa, Upb, Dna;
output A, B, C, D;
sensor c: boolean;
  loop
    await Upa;
    [
     [do
        sustain A
      watching Upb;
      trap X12 in
        loop
          present Upa then
            if ?c else exit X12 end if
          end present;
          emit C
        each tick
      end trap]
    ||
     [% branche de droite]
    ];
    do
      sustain A
    watching Dna
  end loop
end module
```

FIGURE 1

2 RECEPTIVITES

Une réceptivité peut être vraie avant que la transition ne soit validée. Ceci interdit, dans le cas général, la solution simple précédente. Il faut alors considérer les étapes comme des états locaux. LUSTRE peut traiter de façon efficace les états, qui ne sont que des variables booléennes particulières. Nous avons donc développé des techniques de traduction du modèle algébrique associé au GRAFCET vers LUSTRE. Nous avons également fait des traductions vers ESTEREL en nous efforçant de tirer le meilleur partie des séquentialités.
En LUSTRE, on définit un nœud de type ETAPE qui simule le comportement des étapes díun grafcet en respectant les règles G3 et G5.

```
node ETAPE (Init, Set, Reset: bool) returns(X, XS: bool);
-- X est vrai ssi l'étape est active
-- XS est l'état suivant
  let
```

```
    X = Init -> pre(XS);
    XS = if Set then true else
            if Reset then false else X;
  tel;
```

Le grafcet complet est traduit en exprimant les conditions díactivation et de désactivation de chaque étape à líaide díéquations. Pour líexemple figure 1 on obtient (extrait du programme):

```
node GRAFCET1 (a, b, c: bool) returns (A, B, C, D;bool);
  var X1,XS1,X2,XS2,…,SetX1,ResetX1,SetX2,ResetX2,…:bool;
  let
    (X1,XS1) = ETAPE (true, SetX1, ResetX1);
    (X2,XS2) = ETAPE (false, SetX2, ResetX2);
    …
    SetX12 = X11 and (b and not pre(b)); -- X11.↑b

    ResetX12 = X12 and (not c) and (a and not pre(a)); -- X11.c,‾ .↑a
    …
  tel;
```

3 EVOLUTIONS INTERNES

Les traductions faites, aussi bien en ESTEREL qu'en LUSTRE, ont implémenté correctement le caractère synchrone des évolutions. Le mode de simulation dit "sans recherche de stabilité" se programme assez naturellement dans ces langages. Il s'agit en fait à chaque cycle de faire un pas avec parallélisme maximal. Le mode de simulation dit "avec recherche de stabilité" est plus délicat car il faut effectuer dans le même instant toutes les transitions possibles parmi lesquelles certaines décrivent des évolutions internes. La solution que nous avons adoptée, actuellement, consiste à décomposer un instant en micro-instants. Les évolutions à un instant sont le résultat de líexécution de plusieurs pas avec parallélisme maximal, jusquíà la stabilité (si elle existe !). Nous avons eu recours à des artifices de programmation (horloges logiques multiphases) ce qui rend les programmes encore plus lourds.

V. DEVELOPPEMENT EN APPROCHE SYNCHRONE

1 LES ENVIRONNEMENTS POUR LANGAGES SYNCHRONES

A. Généralités

Les **équipes de recherche** ayant créé les langages ESTEREL, LUSTRE et SIGNAL coopèrent dans leurs actions scientifiques (GdR C^3, GdR Automatique (pôle C^2A)). Les **industriels** sont également concernés (C^2A). Des accords d'industrialisation de ses langages ont été signés. Des compilateurs industriels existent pour chacun des langages. Des environnements sont disponibles ou en cours de développement. Un projet actuel est de développer le maximum d'**outils communs**. Ceci passe par l'établissements de "ponts" entre les divers formalismes. Le GRAFCET pourrait, à terme, être un des formalismes d'entrée.
La figure 2 résume les dépendances. Les arcs en traits pleins sont des traductions existantes. Les arcs en tirets sont des traductions en cours de développement ou à venir.

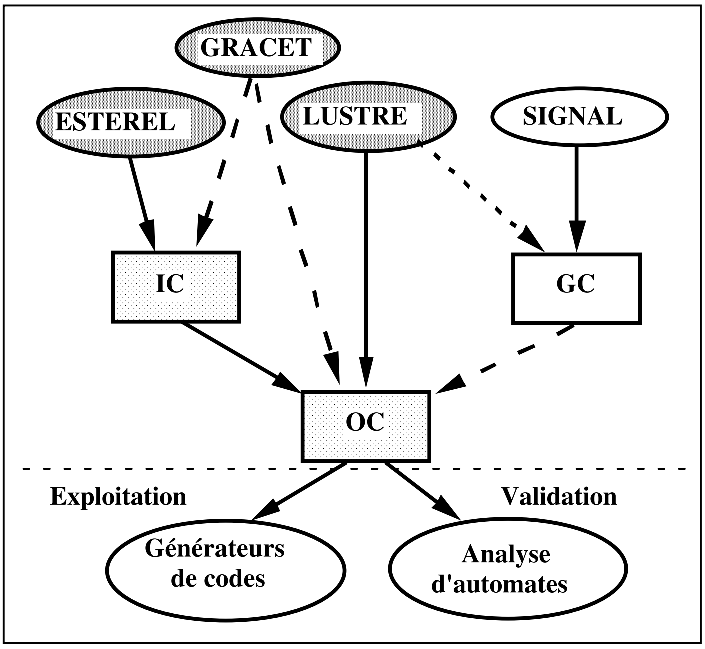


FIGURE 2

B. Code intermédiaire (IC)

IC est un code intermédiaire utilisé par ESTEREL. Il traduit un ou plusieurs modules. Un fichier ic contient des tables (pour les objets manipulés dans le programme) et des instructions dans un langage noyau. Le parallélisme contenu dans le source est conservé dans cette traduction intermédiaire.

C. Code commun (OC)

OC était à l'origine le code objet ESTEREL. Il est maintenant commun à LUSTRE et ESTEREL. On retrouve des tables, mais cette fois les instructions sont remplacées par un automate. Le parallélisme a été mis à plat. Ce travail délicat est, en fait, le plus gros travail effectué par le compilateur.

D. Environnements communs

A partir du format OC (donc d'un automate fini, dont les transitions sont décorées par le code associé à une réaction), on peut faire soit des générations de codes vers des langages standards (C, ADA,…), soit des traitements et validations sur automates (logiciels Auto, Autograph, …).

2 PERSPECTIVES POUR LE GRAFCET

A. Vérification

Les compilateurs de langages synchrones peuvent faire des contrôles intensifs de typage et díusage correct de variables. Ils sont également capables de détecter des erreurs de causalité (existence de cycle de causalité). Une traduction automatique díun GRAFCET en langage synchrone permet au GRAFCET de bénéficier de ces contrôles; il est possible, en particulier, de détecter des instabilités. Plus intéressant encore, le résultat de la compilation peut être analysé avec les logiciels de validation díautomate. Ceci permet de vérifier si la programmation correspond bien aux spécifications.

B. Implémentation

Les traductions proposées dans le paragraphe précédent paraissent très lourdes. Il convient de noter que la longueur du programme source díun langage synchrone níest pas significative de la complexité du code généré. En ESTEREL, par exemple, on a intérêt à créer de nombreux signaux locaux. Pendant la phase de compilation, les signaux redondants sont éliminés. Cíest pourquoi dans les traductions proposées nous níavons pas cherché à minimiser le code produit. Nous avons surtout recherché des traductions génériques. Les expansions de GRAFCET en langage synchrone ne sont pas destinées à des utilisateurs humains, mais à des compilateurs.
Rappelons que la traduction suivie de la compilation est essentiellement faite à des fins de vérification. Si líautomate engendré est de taille raisonnable, rien níempêche de produire une version compilée du GRAFCET dans les langages cibles disponibles. On bénéficie ainsi díune plus grande portabilité.

C. Sémantique

La sémantique des langages synchrones est définie de façon mathématique. Le comportement díun programme correct est parfaitement déterminé. En GRAFCET, les 5 règles díévolutions ne suffisent pas à garantir cette unicité díinterprétation, comme en témoignent les divers modes díinterprétation existants. Nous étudions actuellement la sémantique díun **GRAFCET NOYAU**, avec des techniques héritées des langages synchrones. Un GRAFCET noyau est une forme très dépouillée de GRAFCET et à partir duquel on peut obtenir des GRAFCET standards par expansions structurelles.

D. Utilisation conjointe GRAFCET - langages synchrones

Jusquíà présent nous níavons envisagé la traduction GRAFCET - langage synchrone que pour engendrer le code OC (automate + tables). Il existe une autre possibilité: si on est capable de générer du code IC directement à partir díun GRAFCET alors on pourra décrire directement les applications en adoptant un style de programmation multi-langage. Il est certain que le GRAFCET permet díexprimer certaines synchronisations de façon beaucoup plus agréables quí ESTEREL; de plus le comportement de sous-systèmes peut être déjà modélisé par un GRAFCET. Un langage comme ESTEREL permet par contre díexprimer des comportements plus complexes (mélanges de parallélisme et díexceptions). Líutilisation conjointe du GRAFCET avec les langages synchrones est prometteuse mais elle passe nécessairement par la définition díune sémantique formelle (§ V.2.C).

VI. CONCLUSION

Le GRAFCET et les langages synchrones ont de nombreuses **caractéristiques communes**. La plus significative est le synchronisme des évolutions. Il existe tout de même des différences qui sont à l'origine de difficultés rencontrées lors de nos traductions du GRAFCET vers les langages synchrones.
Une telle **traduction est possible et automatisable**. Dans le cas général la traduction conduit à des programmes cible lourds. Ceci n'est pas un obstacle majeur car les programmes traduits ne sont pas nécessairement à usages humains directs, ce sont souvent des intermédiaires utilisés par d'autres logiciels. Nos traductions ne sont pas uniquement un exercice de style. Nous voyons plusieurs avantages à **établir des ponts entre divers formalismes**.
Tout d'abord l'**aspect multi-formalisme** est intéressant. Dans la conception de systèmes de contrôle sophistiqués, des parties peuvent être typiquement des automatismes séquentiels, donc modélisées de façon adéquate par un GRAFCET, d'autres parties plus informatiques sont plus facilement décrites par un langage.
Le deuxième point est celui de la **validation.** La figure 2 montre que les divers langages synchrones peuvent bénéficier d' environnements d'analyse communs (en particulier les logiciel d'analyse de propriétés d'automates). Par traduction, un GRAFCET peut indirectement hériter de ces environnements.
Enfin, il existe la possibilité d'**implémenter** des contrôleurs grâce aux divers générateurs de codes. On aura ainsi la possibilité de "matérialiser", après traduction, un GRAFCET représentant le point de vue réalisateur.
Afin d'exploiter ces potentialités, nous pensons qu'il serait intéressant de développer une compilation directe du GRAFCET vers OC. Ce travail est certainement lourd et pas nécessairement innovateur. Par

contre, dans un but de meilleure exploitation du parallélisme nous pensons qu'il serait judicieux d'avoir une compilation GRAFCET vers IC. Ceci permettrait des descriptions mixtes ESTEREL - GRAFCET, dans lesquelles on utiliserait les avantages de chacun. De plus on bénéficierait de toute la chaîne de compilation ESTEREL.

VII. BIBLIOGRAPHIE

[AFNOR] **AFNOR**. *Norme C03-190* . Paris, 1982.

[BouSim] **F.BOUSSINOT, R. DE SIMONE**. *The ESTEREL language* , in [IEEE], pp1293-1304.

[Gonthier] **G.GONTHIER**. *Sémantiques et modèles d'exécution des langages réactifs synchrones; application à Estérel* , Thèse, spécialité informatique, Orsay, Mars 1988.

[GREPA] **GREPA**. *Le GRAFCET, de nouveaux concepts* , Cepadues éditions, 1985.

[HCRP] **N.HALBWACHS, P.CASPI, P.RAYMOND, D.PILAUD**. *The Synchronous Data Flow Programming Language LUSTRE* , in [IEEE], pp1305-1320.

[IEEE] (Special section on) *Another look at Real-Time programming*. A.Benveniste , G.Berry Eds., Proceedings of the IEEE, Vol 79, n°9, September 91, pp1268-1336).

[TheTou] **S.THELLIEZ, J.M.TOULOTTE**, *Applications industrielles du GRAFCET* , Eyrolles, 1983.