Rapport INRIA 1994 — Programme 2

Environnement de programmation d'applications temps réel

Projet EP-ATR

3 mai 1995

Projet EP-ATR

Environnement de programmation d'applications temps réel

Localisation : *Rennes*

Mots-clés : analyse de dépendances (1, 14), architecture multiprocesseur (1, 14), calcul formel (1, 12), environnement de programmation (1, 4), EP-ATR (1), graphe conditionnel hiérarchisé (1, 14), interface graphique (1, 4), langage synchrone (1, 4), parallélisme (1, 4, 14), preuve de programme (1, 12), programmation par contraintes (1, 4), Signal (1, 4), SynDEx (1, 14), synthèse d'architecture (1, 14), synthèse de programme (1, 12), système dynamique (1, 12), temps réel (1, 4), VHDL (1, 14).

EP-ATR est un projet commun Inria/CNRS (URA 227).

1 Composition de l'équipe

Responsable scientifique

Paul Le Guernic, DR Inria

Secrétaire

Huguette Béchu, AA Inria

Personnel Inria

Albert Benveniste, DR, projet AS
Patricia Bournai, IR, Atelier
Thierry Gautier, CR
Éric Rutten, CR

Personnel Ura 227

Loïc Besnard, IE, Atelier
Bernard Houssais, maître de conférences, université de Rennes 1
Michel Le Borgne, assistant, université de Rennes 1
Krzysztof Wolinski, maître de conférences, université de Rennes 1

Chercheurs doctorants

Tochéou Pascalin Amagbégnon, bourse MESR
Pascal Aubry, bourse MESR
Mohammed Belhadj, bourse CIES, jusqu'au 31 octobre 1994
Dominique Chauveau, bourse MESR
Apostolos Kountouris, bourse MESR, à partir d'octobre 1994
Hervé Marchand, bourse Inria

Stagiaires

Gábor Paller, bourse CIES, 9 mois à partir d'octobre 1994

2 Présentation générale et objectifs

L'objectif général du projet est de contribuer au développement de concepts, méthodes et logiciels pour la conception et la mise en œuvre d'applications dans le domaine du temps réel. Ces travaux, initiés avec le soutien du Cnet, sont conduits en relation étroite avec différents secteurs applicatifs, tant pour l'expérimentation en aval que pour l'enrichissement de la problématique en amont. Les systèmes temps réel vont des plus petits (ASICs par exemple en électronique ou dans les automatismes) aux plus grands (avionique, énergie, télécommunications, militaire). Quelle que soit leur taille, ces systèmes sont souvent critiques en terme de sûreté. Ils ont généralement un comportement temporel complexe et peuvent s'exécuter sur des architectures non standards (circuits intégrés, architectures distribuées, architectures tolérantes aux fautes...). Dans ce contexte, il est indispensable de disposer d'un ensemble intégré d'outils permettant de décomposer une réalisation en un ensemble de travaux complémentaires : conception et validation des algorithmes indépendamment de toute architecture, implantation progressive sur une architecture, validation.

Le projet contribue à cet objectif par des travaux reposant sur le modèle *flots de données synchronisés* sur lequel est construit le langage

SIGNAL. SIGNAL est un langage temps réel parallèle de style équationnel, pourvu d'une interface graphique de type bloc-diagramme. Les programmes SIGNAL peuvent subir de profondes transformations syntaxiques et architecturales préservant leur sémantique, c'est-à-dire garantissant l'équivalence. Une application peut donc être spécifiée et décrite avec une structure appropriée, puis transformée pour son exécution sur une architecture cible logicielle ou matérielle particulière. Cette reconfiguration architecturale du programme considéré n'engendre pas de surcoût qui résulterait de l'utilisation d'un mécanisme de gestion de tâches. Enfin, il est possible de vérifier formellement des propriétés à toutes les phases de la conception, tant en ce qui concerne le contrôle (synchronisation et logique) que sur les chemins de données.

Ces possibilités sont fournies grâce à l'hypothèse de base de *synchronisme,* qui peut s'exprimer selon les deux points de vue équivalents suivants :

- dans un programme SIGNAL, les calculs et les communications ne prennent pas de temps ;
- un programme SIGNAL est un système d'équations dynamiques discrètes.

Les programmes SIGNAL peuvent donc s'exécuter selon un temps discret composé de « flashs » au cours desquels calculs et communications peuvent se produire. Le temps peut ainsi être multiforme, puisque toute suite d'événements de communication (pouvant être locaux ou externes) peut être utilisée comme une horloge particulière.

La méthode que nous proposons pour la conception d'applications temps réel est alors la suivante :

- conception, spécification et test de l'application indépendamment de l'architecture cible, grâce à l'hypothèse de synchronisme ;
- raffinement progressif vers l'implémentation basé sur des simulations/vérifications à différents niveaux ; à cette étape, la cible est une architecture logicielle pouvant être vérifiée et évaluée ; cette étape peut être complètement réalisée à l'aide de SIGNAL ;
- implantation effective du schéma d'exécution obtenu sur des architectures éventuellement asynchrones et distribuées, en relâchant au besoin l'hypothèse de synchronisme, tout en garantissant une implémentation correcte ;

- génération de composants matériels ou de composants hybrides matériels/logiciels ; à cette fin, SIGNAL et VHDL sont utilisés conjointement.

3 Actions de recherche

Nos recherches abordent toutes les phases du développement d'une application temps réel, de l'expression des algorithmes à leur mise en œuvre sur architecture. Sur ce dernier point en particulier, nous travaillons en étroite collaboration avec le projet Sosso de l'Inria-Rocquencourt, sur le thème de l'adéquation algorithme–architecture pour la commande et le traitement du signal en temps réel.

Nous détaillons dans la suite les différents aspects de nos études :

- programmation temps réel en SIGNAL ;
- études sur le langage et évolutions ;
- propriétés de programmes SIGNAL ;
- méthodes d'implémentation et études d'architectures.

Les actions mises en avant cette année concernent plus particulièrement les points suivants :

- développement d'applications en SIGNAL, avec diverses collaborations (cf. 3.1.2) ;
- définition d'une nouvelle version, SIGNAL V4, en collaboration avec TNI (cf. 3.2.1) ;
- conception conjointe de systèmes matériel–logiciel (cf. 3.4.4).

3.1 Programmation temps réel en Signal

Brève description du langage

Un programme SIGNAL spécifie un système temps réel au moyen d'un système d'équations dynamiques explicites sur des suites de valeurs, les *signaux*. Les index temporels des signaux, ou *horloges,* sont définis par des systèmes d'équations implicites. L'horloge implicitement associée à chaque signal définit l'ensemble des instants où le signal possède une valeur. Les équations d'un programme peuvent être organisées en sous-systèmes (ou processus).

Un signal est défini par un processus élémentaire au moyen d'opérateurs qui peuvent être :

- les extensions aux suites de valeurs des opérateurs classiques arithmétiques ou booléens,
- des opérateurs temporels spécifiques.

Un processus est défini par la composition commutative et associative de processus élémentaires ou composés.

La sémantique de SIGNAL repose sur la sémantique du langage noyau constitué des opérateurs suivants :

`Y := f(X1,...,Xn)`	fonctions ou relations
`Y := X $1`	retard (registre à décalage)
`Y := X when B`	extraction sur condition booléenne
`Y := U default V`	mélange avec priorité
`P | Q`	composition de processus
`P / X`	restriction de visibilité de variable

Une communication associe, à un instant logique du programme (transition de l'automate), une valeur à une variable. Un événement est un ensemble de communications simultanées formant une transition de l'automate. Dans un événement, une variable peut ne pas avoir de valeur associée : on dira alors que le signal est absent. La détermination de la présence d'un signal dans un événement résulte de la résolution d'un système d'équations dans $\mathcal{F}_3$, le corps des entiers modulo 3.

Un processus SIGNAL établit une relation entre les suites que constituent ses signaux d'entrée-sortie. La sémantique d'un programme SIGNAL est alors décrite par un sous-ensemble de l'espace des suites défini par l'ensemble des contraintes induites par chaque opérateur.

Une algèbre de systèmes dynamiques dans $\mathcal{F}_3$, qui décrit complètement la synchronisation d'un processus, est définie sur le langage noyau. Ce calcul, appelé calcul d'horloges, permet la vérification de la correction d'un programme vis-à-vis des synchronisations et la synthèse d'un graphe des dépendances de calculs, conditionnées par des horloges (l'horloge d'une dépendance définit les instants où la dépendance est effective).

3.1.1 Environnement de programmation

Participants : Pascal Aubry, Patricia Bournai

L'environnement de programmation de SIGNAL permet à l'utilisateur de construire ses programmes sous forme à la fois textuelle et graphique. Dans sa forme graphique, une expression SIGNAL est un ensemble de composants (représentés par des rectangles) munis de points de connexion (les ports, représentés par des triangles) joints par des liens (suites de segments connexes).

Nous avons cette année enrichi l'environnement dans la perspective d'en faire un véritable environnement d'exécution graphique de programmes SIGNAL.

Sous l'éditeur graphique et une fois son programme édité, l'utilisateur peut choisir un mode d'affichage pour chaque signal d'entrée ou de sortie. Un contrôle de cohérence est effectué entre le type d'afficheur choisi et le type du signal (boutons pour les entrées de type événementiel ou booléen, entrée clavier pour les entrées numériques, jauges, courbes, aiguilles...). Si aucun afficheur n'est associé à une entrée (ou sortie), alors la lecture (l'écriture) se fait par défaut dans un fichier. Lorsque ces choix sont terminés, un nouveau programme SIGNAL est engendré ; ce programme contient tous les appels à des fonctions externes d'affichage. Ce programme est ensuite compilé et peut être exécuté directement sous l'éditeur graphique de SIGNAL. Il est possible de ralentir la vitesse d'exécution du programme ou de l'exécuter en mode pas à pas.

Certaines améliorations sont encore nécessaires : actuellement par exemple, à une fenêtre d'affichage n'est associé qu'un seul signal ; il faudrait pouvoir visualiser plusieurs signaux dans une seule fenêtre.

Nous avons d'autre part, dans le cadre des travaux effectués autour du format GC (cf. 3.4.2), amorcé le développement d'un outil graphique de mise au point de programmes. Pour cela, un *scrutateur* générique d'arbres et de hiérarchies a été implémenté ; il permet notamment la visualisation graphique de la hiérarchie d'horloges produite par la compilation d'un programme SIGNAL et pourrait être utilisé pour visualiser les cycles et les contraintes d'horloges.

Ce *scrutateur* de hiérarchies pourrait aussi s'appliquer à l'éditeur graphique de SIGNAL. L'ajout d'une fenêtre supplémentaire, représentant l'arbre syntaxique du programme saisi par l'utilisateur, devrait per-

mettre une vision d'ensemble, des déplacements rapides et des manipulations (déplacements et copies de modèles de processus par exemple). Cette facilité nécessite l'implémentation d'un support de manipulation comprenant les fonctions de base d'édition d'arbres (copier/coller).

3.1.2 Expérimentations de Signal, méthodologie de programmation et vérification

Participants : Tochéou Pascalin Amagbégnon, Dominique Chauveau, Bernard Houssais, Paul Le Guernic, Hervé Marchand, Éric Rutten

Les diverses expérimentations de SIGNAL ont un rôle central pour nos recherches. Les applications développées sont aussi d'une grande utilité dans une perspective méthodologique [19]. Dans cette voie, nous avons entrepris la rédaction d'un manuel d'initiation à la programmation en SIGNAL, qui s'appuie sur un grand nombre d'exemples [20].

- En collaboration avec le projet Temis (É. Marchand et F. Chaumette), une application de SIGNAL et de SIGNAL*GTi* (cf. 3.2.2) est menée sur la reconstruction 3-D d'une scène constituée de plusieurs objets en utilisant une caméra embarquée sur l'effecteur d'un robot manipulateur [12, 22]. Pour chacun des types d'objets on dispose de tâches de vision active asservies sur le capteur de vision ; pour reconnaître l'ensemble de la scène, il faut les enchaîner, en commutant d'un asservissement à l'autre. L'automate parallèle hiérarchisé de ce système a été mis en œuvre en SIGNAL par des imbrications de tâches.
- En collaboration avec le projet Siames (S. Donikian, G. André et B. Arnaldi), une expérimentation concerne un environnement de simulation de systèmes physiques animés en images de synthèse, où SIGNAL intervient dans la spécification des comportements dynamiques des objets simulés ; il devra aussi intervenir dans la synchronisation des modules de calcul et dans leur synchronisation.
- L'environnement SIGNAL a été utilisé pour contribuer à une étude de cas comparative proposée par le FZI de Karlsruhe (cette étude a été traitée dans une vingtaine d'autres méthodes formelles, dont plusieurs synchrones). Un contrôleur pour une cellule robotique composée de plusieurs machines et robots a été spécifié sous l'éditeur graphique ; il a été mis en œuvre en communication avec un

simulateur graphique. La satisfaction de certaines propriétés, de sécurité et d'équité, a été vérifiée formellement à l'aide des outils de l'environnement de SIGNAL [2].

- La spécification du comportement d'une cellule de transfert de courant électrique a été réalisée, à partir de descriptions d'Électricité de France, en SIGNAL*GTi* ; les déclenchements et réenclenchements successifs en réaction à des défauts de courant, et en vue de leur identification, confirmation et élimination, ont été décrits en termes de tâches préemptibles et des intervalles de temps sur lesquels elles sont actives.
- Une autre collaboration avec EDF nous a conduits à développer en SIGNAL un simulateur d'appareils électro-domestiques. Cette application utilise une bibliothèque de génération de variables aléatoires permettant de modéliser le comportement physique de certains des composants, mais aussi, de simuler de façon aléatoire des scénarios de mise en fonctionnement des différents appareils constituant le circuit.

3.2 Études sur le langage et évolutions

3.2.1 Nouvelle version de Signal : Signal V4

Participants : Thierry Gautier, Paul Le Guernic

Nous avons terminé, en collaboration avec la société TNI, qui commercialise l'environnement Sildex issu des travaux sur SIGNAL, la définition d'une nouvelle version du langage : SIGNAL V4 [18].

Dans la version actuelle de SIGNAL (H2), l'effort de conception a surtout porté sur la structure temporelle des traitements. Afin de promouvoir la pénétration des techniques synchrones dans des domaines nouveaux tels que le traitement temps réel d'images, nous nous sommes attachés dans la version V4 à étendre la puissance d'expression de SIGNAL en direction de la structure spatiale des traitements. Une première extension est la définition de faisceaux de signaux, dont les types structurés des langages traditionnels constituent un cas particulier : des propriétés de synchronisation peuvent être spécifiées entre les composants d'un faisceau. Une seconde extension porte sur le traitement des tableaux. Dans la version actuelle de SIGNAL, il est possible de décrire des architectures régulières

de traitement avec des fonctions de dépendance très simples (translation). On peut aussi utiliser des instructions séquentielles, palliatif au problème connu de manipulation de tableaux dans les langages à flots de données. En SIGNAL V4, nous déplaçons la séquentialité du traitement (permettant de garantir le caractère fonctionnel des calculs sans contrôle à l'exécution) vers les données : la séquence s'exprime à travers la première dimension d'une structure de données de type tableau. Ces occurrences de valeurs étant totalement ordonnées, cela permet de modéliser ainsi des signaux prenant plusieurs valeurs à un instant donné (séquences). Les tableaux sont alors définis ou utilisés au moyen d'index considérés comme séquences d'entiers. L'opérateur

`T := I : S`	restructuration

définit le tableau `T` comme une restructuration de la séquence `S` selon l'index `I`.

On obtient ainsi une définition séquentielle de tableaux dans un cadre flots de données, sans utilisation de structures de contrôle impératives. Il peut s'agir là d'une étape vers l'utilisation de structures de dépendance plus générales, dès lors qu'elle sera possible dans le cadre du temps réel.

3.2.2 Tâches flots de données et intervalles de temps en Signal : Signal GTi

Participants : Éric Rutten, Paul Le Guernic

Des domaines d'applications tels que le traitement de signal et le contrôle/commande de procédés industriels (par exemple la robotique) requièrent la possibilité de pouvoir spécifier des comportements hybrides. Il s'agit, d'une part, de comportements « continus » (équations, éventuellement dynamiques) et d'autre part de comportements discrets (enchaînement de tâches sur l'occurrence d'événements, ainsi que leur préemption : suspension ou interruption). Ces deux modèles des systèmes à contrôler doivent être combinés de façon à pouvoir spécifier la *commutation* entre des *modes* d'interaction continue avec l'environnement, qui plus est de façon hiérarchisée.

Ce besoin suggère la définition d'un langage mixte issu des modèles à flots de données (équationnels) et de séquencement. Or les langages synchrones sont généralement soit déclaratifs, soit impératifs ; qui plus est, la gestion de tâches non instantanées n'est envisagée qu'à l'extérieur

du cadre synchrone. SIGNAL*GTi* (Gestion de Tâches et d'*i*ntervalles) introduit dans le langage SIGNAL la notion d'*intervalle de temps,* défini par ses événements de début et de fin, et désignant un état qui alterne entre les valeurs *à l'intérieur* et *à l'extérieur* [11]. Par exemple l'intervalle `I`, dans lequel on entre sur l'occurrence d'un événement `B`, et dont on sort sur l'occurrence de `E` est désigné par `I := ]B,E]`.

Associer un tel intervalle de temps et un processus à flots de données `P` définit une *tâche*, c'est-à-dire une activité non instantanée et son intervalle d'exécution (à l'*intérieur,* la tâche a le comportement du processus ; à l'*extérieur,* elle est inexistante : on lui a coupé l'horloge). Nous donnons deux façons de construire une tâche, qui sont aussi deux structures de préemption : dans `P on I`, le comportement de `P` reprend à l'entrée dans `I` à l'état courant de ses variables d'état, alors que dans `P each I`, il reprend depuis son état initial (d'après les déclarations). Au moyen des ces éléments simples, on peut spécifier des séquencements et structures de préemption sur des tâches à flots de données, de façon déclarative, par des contraintes sur les intervalles et des imbrications hiérarchiques de tâches.

Une première version de cette extension à SIGNAL a été mise en œuvre sous la forme d'un préprocesseur au compilateur, qui traduit les structures de SIGNAL*GTi* en SIGNAL.

Ces résultats ont été utilisés dans le cadre d'une étude menée en collaboration avec le projet Temis, ainsi que pour la spécification de comportements dynamiques complexes (cf. 3.1.2).

3.2.3 Extension de Dignal à des modèles probabilistes

Participants : Albert Benveniste, Dominique Chauveau

Le langage SIGN*alea*, défini comme une extension de SIGNAL, permet la spécification et la manipulation de systèmes (partiellement) aléatoires. SIGN*alea* permet en particulier de décrire des automates stochastiques multi-horloges, des réseaux de Petri stochastiques, temporisés ou non, mais aussi des réseaux de neurones (ou réseaux d'automates aléatoires, ou encore champs de Gibbs sur des graphes, selon le vocabulaire utilisé) tels qu'on aime à les utiliser en reconnaissance des formes. Il possède un modèle mathématique, qui permet de fonder l'algorithme de compilation (c'est-à-dire de simulation dans notre cas).

L'extension consiste en l'ajout d'un seul opérateur au langage noyau de SIGNAL :

`potential U(X1,...,Xn)`	énergie de liaison

Cet opérateur requiert que les signaux mentionnés aient tous la même horloge. Considéré isolément, il spécifie une suite de variables aléatoires $(X_1, \cdots, X_n)$ indépendantes et de même loi

$$\frac{1}{Z} e^{-U(X_1,\cdots,X_n)}$$

où Z est une constante appropriée effectuant la normalisation. La combinaison d'une telle équation avec un programme SIGNAL est à interpréter comme suit : un programme SIGNAL établissant une contrainte dynamique entre les signaux $(X_1, \cdots, X_n)$ spécifie un sous-ensemble de comportements légaux de ces signaux, et la distribution résultante est simplement la loi

$$\left(\frac{1}{Z} e^{-U(X_1,\cdots,X_n)}\right)^{\otimes N}$$

conditionnellement à ce que la contrainte spécifiée par le programme SIGNAL *soit satisfaite.*

Par exemple, si l'on prend

$$U(y, x) = -\log \pi(y, x)$$

où $\pi(y, x)$ est une probabilité de transition (c'est-à-dire une famille de probabilités en x paramétrées par y), le programme

```
(| potential U(y,x)
 | y := x $1 init x0
 |)
```

spécifie une chaîne de Markov de probabilité de transition π et de condition initiale x_0.

Cette année, grâce à la visite de Bernard C. Levy, de U.C. Davis, au sein du projet AS, et en particulier, à partir de son travail sur l'estimation linéaire gaussienne, on a pu dégager les primitives pour la compilation et les traitements formels autour de SIG, un sous-ensemble de SIGN*alea* sans traitement automatique du temps. Ces primitives permettent de résoudre automatiquement les problèmes suivants :

- estimer la trajectoire d'un état caché d'un programme SIG ;
- calculer la vraisemblance d'une suite d'observations, par rapport à un modèle spécifié en SIG.

On dispose ainsi d'outils théoriques pour, à partir d'une description SIG, engendrer automatiquement des algorithmes de Viterbi pour des modèles de type *Hidden Markov Model,* ou, plus généralement, pour engendrer des algorithmes de diagnostic temps réel.

Les primitives dégagées correspondent à une généralisation des notions de vraisemblance et de statistique suffisante pour des programmes SIG, ainsi qu'à une formule générale permettant leur calcul incrémental [8, 15].

Ces méthodes de simulation et d'estimation incrémentale se rapprochent assez fortement des travaux de A.P. Dempster et G. Shafer en Statistique et Intelligence Artificielle sur les fonctions de croyance et réseaux de croyance.

3.3 Propriétés temporelles des programmes SIGNAL

Un processus SIGNAL établit une relation entre les suites que constituent ses signaux d'entrée–sortie. L'étude formelle des synchronisations induites est nécessaire d'une part à la génération d'un code exécutable et d'autre part à la vérification des propriétés du processus. Cette étude est conduite, au travers d'un homomorphisme associant à un programme SIGNAL un autre programme SIGNAL restreint aux suites booléennes, dans les anneaux de polynômes sur $\mathcal{F}_3$. Un processus y est codé par un système de la forme :

$$\begin{aligned} Q(M_n, X_n, Y_n) &= 0 \\ M_{n+1} &= P(M_n, X_n) \\ Q_0(M) &= 0 \end{aligned}$$

où M_n, M_{n+1}, X_n ,Y_n représentent des multivariables, Q, Q_0 des ensembles finis de polynômes, et $Q(M_n, X_n, Y_n) = 0$ (partie statique) l'ensemble des états admissibles. Une étude particulière de cet ensemble est intéressante à deux titres : elle est nécessaire à la génération de code (sachant que pour ce faire l'étude du système complet est très coûteuse) ; il est de plus bien connu que les systèmes de traitement symbolique sont très sensibles à l'ordre des variables ; une étude préliminaire de la structure de l'ensemble des états admissibles permet d'offrir un ordre partiel adéquat au traitement complet.

3.3.1 Structure de l'ensemble des états admissibles

Participants : Tochéou Pascalin Amagbégnon, Loïc Besnard, Paul Le Guernic

Le calcul statique repose sur l'arborescence d'inclusion des variables booléennes décrivant l'ensemble des états admissibles : l'extraction sur condition *libre* (signal booléen d'entrée ou expression booléenne sur signaux non booléens) induit une partition des horloges et conduit à la construction d'une forêt dont les racines sont les horloges de ces conditions libres. Toute expression d'horloge peut alors être progressivement placée dans cette forêt représentant un ou plusieurs arbres d'inclusion. Chaque expression ainsi placée est mise sous forme normale en tant que somme (borne supérieure) de produits (borne inférieure) d'extractions.

L'objectif poursuivi en cherchant à faire de cette forme normale une *forme canonique* est de donner à chaque formule d'horloge une représentation unique de façon à pouvoir vérifier efficacement l'équivalence de deux horloges.

Cet objectif a été atteint en utilisant des BDD (Binary Decision Diagrams) pour effectuer les calculs sur les formules d'horloge. Chaque nœud de l'*arbre d'horloges* est désormais décoré avec un BDD faisant ainsi de cet arbre une forme canonique [14]. L'utilisation d'une bibliothèque logicielle de fonctions de manipulation de BDD a permis une mise en œuvre de la forme canonique arborescente.

L'apport principal de cette structure de données est l'accroissement des possibilités de preuves au cours de la compilation et une plus grande efficacité du code généré, et ce en conservant des temps de compilation du même ordre de grandeur qu'avant.

3.3.2 Étude des propriétés de synchronisation

Participants : Michel Le Borgne, Hervé Marchand

L'étude des systèmes dynamiques repose sur l'utilisation de techniques algébriques sur les corps de Galois. Elle vise à exprimer les propriétés des systèmes dynamiques et à donner une solution algorithmique pour leur vérification et pour la synthèse de systèmes satisfaisant certaines spécifications.

L'étude systématique des problèmes de contrôle a été poursuivie cette année. La synthèse d'un contrôleur dans le cas où la spécification du contrôle est donnée à partir d'un langage qui n'est pas localement testable a été élucidée.

Par contre les progrès réalisés dans le domaine de la synthèse modulaire et en particulier sur le problème de réalisation lié à l'abstraction, ne permettent pas encore de proposer des techniques de calcul d'une abstraction d'un système dynamique polynômial (en dehors, bien entendu, des techniques directement transposables des automates).

Du côté des applications, le travail s'est poursuivi dans le cadre de la thèse de H. Marchand (contrat EDF). Après une phase d'étude des diffèrentes représentations des systèmes à événements discrets proposées dans la littérature [23], la modélisation des automatismes de commande des postes de transformation a commencé. Nous avons pour cette modélisation, tiré avantage des travaux sur les intervalles menés dans l'équipe. Nous disposons actuellement d'un modèle en SIGNAL pour chaque type de cellule, constituant élémentaire des postes de transformation. Un simulateur configurable est en cours d'élaboration (cf. 3.1.2).

Les automatismes des postes de transformation faisant intervenir des compteurs bornés, nous étudions actuellement leur intégration dans la théorie du contrôle des systèmes dynamiques. Le but est d'une part d'implémenter le calcul formel sur ces compteurs en utilisant une technique inspirée des BDD, d'autre part de calculer des contrôleurs comportant des compteurs.

Signalons enfin qu'une nouvelle version du système de calcul formel Sigali est en cours de développement. Elle se distingue de la précédente par la gestion mémoire et a été conçue pour servir de base à l'implémentation des compteurs.

3.4 Méthodes d'implémentation et études d'architectures

3.4.1 Outils de manipulation de graphes

Participants : Loïc Besnard, Thierry Gautier, Paul Le Guernic

La représentation interne d'un programme SIGNAL sous la forme d'un graphe hiérarchisé, conditionné par les horloges, est centrale pour un

grand nombre d'outils : génération de code, optimisation, évaluation quantitative, implémentation parallèle, etc.

Un outil de base pour la méthodologie d'implémentation parallèle que nous proposons [9, 10, 19, 21] est le calcul d'horloges et de chemins. Dans ce cadre, une nouvelle stratégie a été mise en œuvre pour le calcul de la fermeture transitive du graphe (réduite aux entrées/sorties). Jusque là, le calcul des horloges des dépendances était réalisé à la volée» (c'est-à-dire au cours du parcours du graphe). Dans la nouvelle technique, les équations sont collectées et résolues globalement après le parcours du graphe. Le placement des horloges dans la hiérarchie n'est effectué qu'après l'application de réductions booléennes. Ces modifications ont permis de réduire considérablement le temps de calcul de fermeture transitive.

D'autre part, des outils de base pour la manipulation du graphe des programmes SIGNAL ont été intégrés au compilateur. Un graphe est constitué de nœuds horloges, de nœuds signaux et de dépendances qui expriment les contraintes de séquencement. Cette représentation d'un programme SIGNAL a été généralisée afin de permettre une manipulation du graphe produit. L'objectif est de permettre à l'utilisateur de réorganiser le graphe par des transformations automatiques (calcul de *lignées, granules...*) ou semi-automatiques par édition du graphe produit sous forme graphique. Certaines transformations automatiques ont été mises en œuvre.

L'objet manipulé devient une hiérarchie définie comme une succession de graphes condensés du graphe initial dans des partitions qui s'ordonnent totalement. Actuellement, la partition ne porte que sur des sous-graphes représentant des calculs mono-cadencés. La mise en œuvre du partitionnement sur entrées, effectué sur les graphes de signaux synchrones, utilise cette nouvelle implémentation.

3.4.2 Format commun des langages synchrones

Participants : Pascal Aubry, Albert Benveniste, Thierry Gautier, Paul Le Guernic

Dans le cadre du projet Synchrone (CP2I et Groupement C2A), les équipes engagées dans le modèle synchrone (autour des langages Esterel, Lustre et SIGNAL et du système d'aide à la répartition SynDEx)

ont défini un socle commun à la programmation synchrone, constitué de trois formats [7] :

IC est un format parallèle de type *impératif.*

GC est un format parallèle de type *graphe flots de données.*

OC est un format automate» séquentiel.

Les formats communs des langages synchrones sont au cœur du projet Eureka SYNCHRON.

Le document initial de définition a été traduit en anglais et quelques révisions ont été effectuées. La première définition des formats est maintenant confrontée à l'expérience des mises en œuvre.

À l'Irisa, nos expérimentations ont porté essentiellement sur le format GC. Un générateur de GC à partir de programmes SIGNAL est intégré dans l'environnement (option du compilateur SIGNAL). Les sources produits sont une image du graphe hiérarchisé conditionné issu du calcul d'horloges.

Les codes ainsi engendrés peuvent être validés par un compilateur de sources GC, qui accepte les sources produits par les autres participants au projet SYNCHRON générant du GC.

En tant que *format,* à la différence d'un langage, GC est difficilement lisible (à cause essentiellement d'une grammaire d'index, par opposition à une grammaire d'identificateurs). Afin de faciliter la mise au point des passerelles et de visualiser le format GC de manière plus conviviale, un *scrutateur* générique d'arbres et de hiérarchies a été développé. Il est actuellement intégré aux compilateurs GC et SIGNAL (via l'option de génération de GC).

La maîtrise du format GC va permettre la mise en place de nouveaux outils, tels que des partitionneurs de graphes et autres répartiteurs de code.

3.4.3 Évaluations quantitatives

Participants : Patricia Bournai, Paul Le Guernic

L'environnement SynDEx, développé à l'Inria-Rocquencourt (Yves Sorel et Christophe Lavarenne, projet Sosso), fournit une aide à l'implantation temps réel multi-processeur des algorithmes spécifiés avec SIGNAL

en déchargeant au maximum l'utilisateur des tâches lourdes de programmation bas niveau (système). Il permet notamment une distribution et un ordonnancement des processus SIGNAL optimisant le temps de réponse de l'algorithme sur une machine multi-processeur.

Cette année, l'interface Sisy [16] entre SIGNAL et SynDEx a pu être consolidée au vu des utilisations qui en ont été faites, aux LER Thomson notamment.

3.4.4 Synthèse de circuits et conception de systèmes matériel–logiciel

Participants : Mohammed Belhadj, Thierry Gautier, Paul Le Guernic, Krzysztof Wolinski

La conception d'architectures en utilisant SIGNAL et VHDL et d'une manière générale, la conception conjointe de systèmes matériel–logiciel deviennent un thème majeur du projet. C'est sur ce thème que porte la thèse de Mohammed Belhadj [1].

Les axes suivants ont été étudiés :

- synthèse de circuits synchrones à partir de descriptions SIGNAL ;
- synthèse de circuits GALS (globalement asynchrones, localement synchrones) ;
- vérification d'architectures décrites en VHDL en utilisant les possibilités de preuves associées à SIGNAL ;
- génération automatique d'un noyau générique pour chacun des processeurs d'une machine parallèle composée de processeurs TMS320C40.

Le premier thème correspond à la mise en œuvre matérielle de SIGNAL en utilisant les outils de synthèse VHDL [5]. Nous avons défini une génération à plusieurs niveaux d'abstraction de VHDL, suivant le niveau d'abstraction du programme SIGNAL de départ. Ainsi, un programme SIGNAL peut être traduit en VHDL de niveau comportemental, ce qui permet l'utilisation d'outils de synthèse architecturale tels que Synopsys. La traduction peut aussi être de niveau transfert de registres ou de niveau portes. Le résultat de la synthèse est un circuit synchrone correspondant à la totalité d'une application SIGNAL.

Dans le deuxième thème, nous avons poursuivi les travaux concernant la génération de spécifications en VHDL correspondant à une architecture GALS [13]. Les avantages de cette approche sont les suivants :

- élimination du problème de l'horloge commune ;
- possibilité de partitionnement d'une application et synthèse séparée.

Dans le cadre de ce thème, un processeur correspondant à un nœud d'un graphe d'exécution a été réalisé en technologie FPGA.

Le troisième thème concerne la vérification de programmes VHDL dans l'environnement SIGNAL. La vérification formelle de programmes VHDL passe avant tout par une définition formelle d'un sous-ensemble du langage [6]. Deux sous-ensembles de VHDL ont été définis en SIGNAL : un sous-ensemble « synchrone » et un sous-ensemble « temporisé » qui inclut des constructeurs temporels et devrait permettre de raisonner sur des programmes VHDL utilisant les différents modèles de temps.

Le quatrième thème correspond à la génération du noyau générique pour les processeurs formant une machine parallèle composée de TMS320C40. Un générateur de noyau a été défini et réalisé. Le graphe d'exécution d'application SIGNAL utilisé pour la génération du code est le même que pour la synthèse de circuits GALS. Cela permettra prochainement d'exécuter une partie de l'application SIGNAL sur la machine, l'autre partie étant exécutée par le composant synthétisé.

Nous sommes engagés d'autre part dans le projet Asar de mise en place autour de CENTAUR d'un atelier d'accueil générique pour la synthèse architecturale [3, 4]. Un des principaux objectifs du projet est de permettre la conception conjointe matériel–logiciel en utilisant au besoin différents formalismes et les approches associées : VHDL, langages synchrones SIGNAL et Lustre, langage Alpha pour la description de parties régulières, outils de synthèse Osys et Gaut, etc. Pour cela, il est indispensable de disposer d'un format commun de représentation et de communication. L'ensemble des participants au projet se sont mis d'accord pour utiliser le format GC, lequel doit pouvoir être adapté aux besoins spécifiques du domaine.

4 Actions industrielles

TNI

Nous collaborons étroitement avec la société TNI, qui assure l'industrialisation de SIGNAL à travers l'environnement Sildex. Un axe essentiel de cette collaboration concerne la diffusion du synchrone en général, et en particulier des outils développés d'un côté et de l'autre autour de SIGNAL (cf. 6.3).

D'autre part, la définition de la nouvelle version de SIGNAL, SIGNAL V4, a été l'objet d'un travail en commun important. Si les travaux portant sur la sémantique de cette version sont pour l'essentiel effectués à l'Irisa, François Dupont, de TNI, a participé activement à l'élaboration de la syntaxe en fonction des besoins exprimés par les utilisateurs de Sildex, priorité nous semblant devoir être donnée sur ce plan aux acteurs industriels des domaines d'application. Le document V4 [18], cosigné par F. Dupont, présente ces évolutions.

Projet SYNCHRON

Le projet Eureka SYNCHRON, dont la mise en place a été conduite dans le cadre du groupement C2A, a démarré cette année. Ce projet regroupe des chercheurs et des industriels : Inria (centres de Rennes, Rocquencourt, Sophia-Antipolis et Grenoble), CP2I, TNI, Ilog, Verilog, Schneider-Électrique, Saab et Logikkonsult. Les deux derniers cités sont des partenaires suédois du projet. De nouveaux partenaires pourraient entrer dans le projet à partir de 1995.

L'objectif du projet SYNCHRON est de développer, de standardiser et de promouvoir la technologie synchrone. Le format SYNCHRON », qui regroupe les formats communs des langages synchrones, est au centre de cette action. En particulier, un effort de standardisation du format est entrepris.

Divers outils ont été développés autour du format. Les codes engendrés (en GC, IC et OC) par les différents participants du projet SYNCHRON sont accessibles sur un compte *ftp* géré à l'Irisa par le projet EP-ATR. Il permet notamment d'assurer la cohérence entre les équipes de développement. À ce jour, le compte *ftp* contient, en plus de la documentation liée au projet, des codes IC engendrés par Ilog et de nombreux codes GC engendrés par TNI et par l'Irisa.

Parallèlement, l'action de développement Inria Speak a été mise en place (début effectif : novembre 1994). Le but de cette action est de développer une version « domaine public » du format.

Thomson

Notre collaboration avec Thomson se poursuit avec les LER d'une part et devrait s'élargir dans le cadre de l'accord Inria–Thomson :

- Une convention de mise à disposition du compilateur SIGNAL H2 a été signée avec Thomson-CSF LER ; SIGNAL y est utilisé pour décrire le séquencement d'applications de traitement d'images, et générer les objets nécessaires à leur exécution sur des architectures parallèles hétérogènes.

  Une application est d'abord décrite sous l'environnement graphique dédié à la machine P3I. À partir de cette description, le programme SIGNAL correspondant est engendré puis compilé. Si des erreurs d'horloges sont détectées, elles sont analysées et remontées au niveau de l'environnement graphique. Dans le cas contraire, le code Sisy (interface SIGNAL-SynDEx) engendré est analysé pour construire les tables nécessaires au noyau d'exécution réparti de P3I.

- Un accord cadre a été signé entre Thomson-CSF et l'Inria. Des discussions sont actuellement conduites avec Thomson Sintra-ASM (en concertation avec le projet Api de l'Irisa) et les LER en vue de définir des coopérations particulières dans les domaines des traitements radar/sonar et du traitement d'images.

EDF

Nous avons une collaboration avec EDF (Mazen Samaan, groupe Acsar de la direction des études et recherches) concernant la conception d'automates de contrôle de postes de transformation électrique. Dans ce cadre a été établi un état de l'art sur les méthodes de synthèse de contrôleurs de systèmes à événements discrets. Nous avons aussi modélisé en SIGNAL*GTi* les comportements d'un poste de transformation électrique pour la reprise en cas de défaut de courant. De son côté, EDF conduit une expérimentation lourde sur SIGNAL et Sildex.

Alcatel Alsthom Recherche

A. Benveniste et P. Le Guernic, en coopération avec J.F. Abramatic et G. Kahn, conduisent une action prospective dans le cadre d'une convention de conseil avec Alcatel Alsthom Recherche.

France Télécom – Cnet

Dans le cadre des consultations thématiques de France Télécom – Cnet, nous avons mis en place, sur le thème Techniques et outils pour la conception conjointe logiciel/matériel (codesign)», l'action concertée Cairn, regroupant les projets Api et EP-ATR. Ce projet concerne l'utilisation conjointe des langages SIGNAL et Alpha pour la spécification, la simulation et la réalisation matérielle et logicielle d'une application. Le travail portera notamment sur l'intégration SIGNAL–Alpha et l'étude théorique du problème d'interfaçage entre structures régulières et irrégulières. L'accent sera également porté sur l'intégration de ces langages dans des environnements de conception standards, à base de VHDL.

5 Actions nationales et internationales

Groupement C2A

Le projet assure l'animation du groupement C2A, pôle 4 du GDR Automatique du CNRS. Ce groupement fédère une grande partie de la recherche effectuée en France autour de la programmation synchrone. Il comporte en particulier de nombreux partenaires industriels, tels que Schneider-Électrique, Dassault-Aviation, Thomson, EDF, Snecma, etc. Ce groupement joue un rôle stratégique dans la diffusion de la technologie synchrone.

Le groupement C2A a notamment été le cadre de la définition et des premières évaluations des formats communs des langages synchrones. Il a aussi été le lieu de mise en place du projet Eureka SYNCHRON.

GDR 134

Nous participons au GT6 du GDR134 TDSI, groupement de recherche coordonné du CNRS dans le domaine du traitement du signal et de l'image. Ce groupe de travail étudie les problèmes liés à l'adéquation logiciel/matériel dans ce domaine ainsi que les méthodologies de

programmation. Il s'appuie sur l'utilisation conjointe de SIGNAL et SynDEx.

Projet inter–PRC ASAR

Ce projet regroupe différents laboratoires de recherche français impliqués dans la synthèse architecturale : projets EP-ATR et Api à l'Irisa, projet Croap à l'Inria-Sophia, Lasti à l'Enssat de Lannion et à l'IUP de Lorient, Cert-Onera Deri à Toulouse, I3S à Nice, Lami à Évry. L'objectif du projet est de mettre en place un atelier d'accueil générique pour la synthèse architecturale bâti autour de Centaur. Le format GC servira de représentation intermédiaire pour l'ensemble des formalismes utilisés.

Projet inter–PRC VIA (vision intentionnelle et action)

Ce projet est le cadre d'une expérimentation menée en collaboration avec le projet Temis : des tâches de vision, au caractère flots de données intrinsèque, sont spécifiées et mises en œuvre en SIGNAL, tandis que leur séquencement l'est avec SIGNAL*GTi*.

Collaboration avec Siemens

Les contacts avec le centre de recherche Siemens à Munich se sont poursuivis en vue d'une collaboration effective. Nous envisageons en particulier la génération à partir de SIGNAL d'automates exprimés dans le langage de description d'automates de Siemens.

Collaboration franco–indienne

Une collaboration avec l'IISc (Bangalore) et le TIFR (Bombay) a été mise en place sous l'égide du centre franco-indien pour la promotion de la recherche avancée. Il s'agit d'utiliser des variantes du paradigme synchrone pour modéliser et spécifier les systèmes de production automatisés. Une visite de deux semaines dans le projet de R.K. Shyamasundar, du *Tata Institute of Fundamental Research* (Bombay), a permis de dégager des thèmes de recherche communs.

6 Diffusion des résultats

6.1 Colloques et congrès

On pourra se reporter à la bibliographie pour la liste des colloques et congrès auxquels les membres de l'équipe ont participé.

Nous avons présenté l'environnement SIGNAL, ainsi que SynDEx, à l'exposition internationale CEBIT, sur le stand de l'Inria, à Hanovre (RFA), du 16 au 18 mars 1994.

6.2 Actions d'enseignement

P. Le Guernic et B. Houssais ont donné un cours de DEA et un cours de 5^{e} année Insa et DESS-Isa consacrés à la programmation du temps réel.

Les membres de l'équipe participent à divers titres à la formation d'étudiants à l'université et à l'Insa.

Dans le cadre des formations de troisième cycle ou da stages d'été, nous avons assuré l'encadrement d'étudiants stagiaires :

- Florent Martinez (DEA informatique), SIGNAL*GTi : Intervalles de temps et préemption de tâche en* SIGNAL ;
- Mora Cheav (DEA informatique), *Un environnement de simulation graphique pour le langage* SIGNAL ;
- Jean-Pascal Bodin (Maîtrise informatique), *Programmation en* SIGNAL *d'une cellule d'un poste de transformation électrique ;*
- Samuel Ketels (Maîtrise informatique), *Programmation en* SIGNAL *de tâches de vision active pour la robotique ;*
- Sylvain Machard (Maîtrise informatique), *Représentation graphique d'arbres sous X11R5 : Implémentation générique orientée objet, Applications.*

P. Le Guernic et Th. Gautier ont donné des cours sur SIGNAL à Supelec (Rennes et Gif-sur-Yvette).

É. Rutten a présenté SIGNAL et SIGNAL*GTi* dans le cadre du DESS de Productique de l'UBO (Brest).

6.3 Diffusion de logiciels

La version Inria de SIGNAL n'est pas diffusée par *ftp* mais peut, lorsque des raisons spécifiques justifient son emploi plutôt que celui de la version commerciale vendue par TNI (Sildex), être obtenue dans le cadre d'une convention de mise à disposition signée pour un an renouvelable. Au terme de cette mise à disposition, un rapport d'utilisation est demandé par l'Inria. Cette gestion qui préserve les intérêts de notre partenaire industriel entraîne une activité importante qui nous paraît néanmoins nécessaire. Une trentaine de conventions ont été passées. De son côté, TNI a distribué une trentaine de licences Sildex.

7 Publications

Thèses

[1] M. BELHADJ, *Conception d'architectures en utilisant* SIGNAL *et VHDL*, thèse de doctorat, université de Rennes 1, décembre 1994.

Articles et chapitres de livre

[2] T. P. AMAGBEGNON, P. LE GUERNIC, H. MARCHAND, E. RUTTEN, *Case Study "Production Cell", A Comparative Study in Formal Software Development*, Springer-Verlag, 1994, ch. SIGNAL – The Specification of a Generic, Verified Production Cell Controller, Lecture Notes in Computer Science, à paraître.

Communications à des congrès, colloques, etc.

[3] P. ASAR (M. AUGUIN, M. BELHADJ, J. BENZAKKI, C. CARRIÈRE, G. DURRIEU, T. GAUTIER, M. ISRAËL, P. LE GUERNIC, M. LEMAÎTRE, E. MARTIN, P. QUINTON, L. RIDEAU, F. ROUSSEAU, O. SENTIEYS), «Towards a Multi-Formalism Framework for Architectural Synthesis: the Asar Project», *in : Proceedings of the Third International Workshop on Hardware/Software Codesign*, IEEE Computer Society Press, p. 25–32, Grenoble, septembre 1994.

[4] P. ASAR (P. AUBRY, M. AUGUIN, M. BELHADJ, J. BENZAKKI, T. BOUGUERBA, C. CARRIÈRE, G. DURRIEU, T. GAUTIER, M. ISRAËL, P. LE GUERNIC, M. LEMAÎTRE, E. MARTIN, P. QUINTON, L. RIDEAU, F. ROUSSEAU, O. SENTIEYS), «Framework and Multi-Formalism: the

Asar Project», *in : Proceedings of the 4th International Ifip 10.5 Working Conference on Electronic Design Automation Frameworks*, Gramado (Brésil), novembre 1994.

[5] M. Belhadj, «Using VHDL for Link to Synthesis Tools», *in : North Atlantic Test Workshop NATW'94*, Nîmes, 1994.

[6] M. Belhadj, «VHDL & Signal: A Cooperative Approach», *in : International Conference on Simulation and Hardware Description Languages, Western Simulation Multi-Conference*, Society for Computer Simulation, p. 76–81, Tempe, Arizona (usa), 1994.

[7] A. Benveniste, T. Gautier, P. Le Guernic, G. Berry, F. Mignard, P. Caspi, N. Halbwachs, P. Couronné, F. cois Dupont, C. Le Maire, J.-P. Paris, Y. Sorel, «Synchronous technology for real-time systems», *in : RTS'94, Actes des conférences*, Teknea, p. 105–122, 1994.

[8] A. Benveniste, B. C. Levy, É. Fabre, P. Le Guernic, «A calculus of stochastic systems for the specification, simulation, and hidden state estimation of hybrid stochastic/nonstochastic systems», *in : 3rd International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Springer-Verlag, p. 149–169, septembre 1994. Lecture Notes in Computer Science 863.

[9] O. Maffeïs, P. Le Guernic, «Distributed Implementation of Signal : Scheduling & Graph Clustering», *in : 3rd International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Springer-Verlag, p. 547–566, septembre 1994. Lecture Notes in Computer Science 863.

[10] O. Maffeïs, P. Le Guernic, «From Signal to fine-grain parallel implementations», *in : Int. Conference on Parallel Architectures and Compilation Techniques*, Ifip A-50, North-Holland, p. 237–246, août 1994.

[11] E. Rutten, P. Le Guernic, «Sequencing data flow tasks in Signal», *in : Proceedings of the ACM SIGPLAN Workshop on Language, Compiler and Tool Support for Real-Time Systems*, Orlando, Florida (usa), juin 1994.

[12] E. Rutten, E. Marchand, F. Chaumette, «The sequencing of data flow tasks in Signal: application to active vision in robotics», *in : Proceedings of the 6th Euromicro Workshop on Real Time Systems*, IEEE Publ., p. 80–84, university of Maelardalen, Västerås (Sweden), juin 1994.

[13] K. Wolinski, M. Belhadj, «High Level Synthesis of Globally Asynchronous Locally Synchronous Circuits», *in : North Atlantic Test Workshop NATW'94*, Nîmes, 1994.

Rapports de recherche et publications internes

[14] T. Amagbegnon, L. Besnard, P. Le Guernic, «Arborescent Canonical Form of Boolean Expressions», *rapport de recherche n°2290*, Inria, juin 1994.

[15] A. Benveniste, B. C. Levy, E. Fabre, P. Le Guernic, «A calculus of stochastic systems for the specification, simulation, and hidden state estimation of hybrid stochastic/nonstochastic systems», *publication interne n°837*, Irisa, juillet 1994.

[16] P. Bournai, C. Lavarenne, P. Le Guernic, O. Maffeïs, Y. Sorel, «Interface Signal–SynDEx», *rapport de recherche n°2206*, Inria, mars 1994.

[17] B. Dutertre, M. Le Borgne, «Control of Polynomial Dynamic Systems: an Example», *rapport de recherche*, Inria, janvier 1994.

[18] T. Gautier, P. Le Guernic, F. Dupont, «Signal V4 : manuel de référence», *publication interne n°832*, Irisa, juin 1994.

[19] T. Gautier, P. Le Guernic, O. Maffeïs, «For a New Real-Time Methodology», *rapport de recherche n°2364*, Inria, octobre 1994.

[20] B. Houssais, M. Le Borgne, P. Le Guernic, «Cours de programmation en langage temps-réel Signal», *rapport de recherche*, Irisa, juin 1994.

[21] O. Maffeïs, P. Le Guernic, «From Synchronous-Flow Dependence Graphs to Reliable and Efficient Implementations», *research report n°02/94-R029*, Ercim, février 1994.

[22] E. Marchand, E. Rutten, F. Chaumette, «Applying the Synchronous Approach to Real Time Active Visual Reconstruction», *rapport de recherche n°2383*, Inria, novembre 1994.

[23] H. Marchand, M. Le Borgne, «Modèles des systèmes à événements discrets contrôlés», *rapport d'avancement*, Irisa, 1994, contrat EDF.

[24] H. Marchand, M. Le Borgne, «Typage des graphes de décisions ternaires», *rapport de recherche n°2185*, Inria, mars 1994.

Divers

[25] A. Benveniste, «Synchronous languages provide safety in reactive systems design», *Control Engineering*, septembre 1994, pp. 67–69.

8 Abstract

The objective of the project is to develop a method to design and implement real time algorithms onto multi-processors. The synchronous language SIGNAL has been defined for this purpose. Main topics are: real-time and parallel processing semantics; program distribution and hierarchical automata; asynchronous processing of synchronous programs. Control of industrial processes, speech and image coding and real time processing, telecommunication and videocommunication are in the scope of the project.

Scientific Context:

- A language for real-time: SIGNAL
- A graphic environment for SIGNAL
- Automatic verification of temporal properties of programs
- Implementation on multi-processors
- Hardware/software codesign

Applications:

- Active vision in robotics
- Electrical power transfer cell
- Specification, implementation and verification of a production cell

Table des matières