# Rapport d'activité scientifique 1997

## Action Vasy-RA

Validation de Systèmes – Recherche et Applications

Thème Inria 1c

Réseaux et systèmes

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# Table des matières

# Action Vasy-RA

**Validation de Systèmes – Recherche et Applications**

---

**Localisation** : *Rhône-Alpes*

**Mots-clés** : application critique, application répartie, compilation, concurrence, génération de code, génie logiciel, logique temporelle, modélisation, méthodes formelles, parallélisme asynchrone, protocole de communication, spécification formelle, synchronisation, système distribué, vérification de programme.

## 1 Composition de l'équipe

**Responsable scientifique**

Hubert Garavel, CR1 Inria

**Assistante de projet**

Béatrice Claudio

**Ingénieur expert**

Mark Jorgensen, ingénieur Dyade

**Chercheurs extérieurs**

Ghassan Chehaibar, ingénieur Bull
Massimo Zendri, ingénieur Bull

**Chercheurs post-doctorants**

Charles Pecheur, bourse Inria Rhône-Alpes

**Chercheurs doctorants**

Radu Mateescu, bourse MENESR
Mihaela Sighireanu, bourse MENESR

**Stagiaires**

David Jacquemin, élève-ingénieur ENSIMAG, juillet–août 1997
Bruno Vivien, élève-ingénieur CNAM, octobre 1996–septembre 1997
Patrick Wendel, élève-ingénieur ENSIMAG, juillet–août 1997

# 2 Présentation générale et objectifs

## 2.1 Introduction

Créée au 1er janvier 1997, l'action VASY "Recherche et Applications" s'inscrit dans la problématique de la conception de systèmes sûrs par l'utilisation de méthodes formelles.

Plus précisément, nous nous intéressons à tout système (matériel, logiciel, télécommunications) faisant intervenir du parallélisme *asynchrone*, une modélisation du parallélisme basée sur la sémantique d'entrelacement (*interleaving semantics*) et bien adaptée à la description de systèmes répartis.

Pour la conception de systèmes sûrs, nous préconisons l'utilisation de techniques de description formelles, complétées par des outils informatiques adaptés, offrant des fonctionnalités de simulation, prototypage rapide, validation et vérification formelle.

Parmi les différentes approches existantes pour la vérification, nous concentrons nos efforts sur la vérification "basée sur les modèles" (*model-checking*) qui recouvre un grand nombre de techniques spécialisées (vérification énumérative, à la volée, symbolique...). Ces techniques, bien que moins générales que les approches par preuves (*theorem proving*), possèdent pourtant l'avantage de permettre une détection rapide et économique des erreurs de conception dans des systèmes complexes.

Nos travaux se situent au confluent de deux grandes approches en méthodes formelles : l'approche basée sur des *modèles* (très répandue en Amérique du Nord) et l'approche basée sur des *langages* (plus développée en Europe) :

- Sous le terme de *modèles*, on désigne diverses représentations de programmes parallèles (automates, réseaux d'automates communicants, réseaux de Petri, diagrammes de décision binaire, etc.) ainsi que les algorithmes de vérification qui s'y appliquent. D'un point de vue théorique, il importe de rechercher des résultats généraux, donc indépendants de tout langage de description particulier, ce qui incite à la recherche de modèles mathématiques simples et généraux.
- En pratique, ces modèles sont souvent trop rudimentaires pour servir à la description directe d'un système complexe (une telle approche est fastidieuse et comporte un fort risque d'erreur). C'est pourquoi il est indispensable de s'appuyer sur des formalismes de plus haut niveau (c'est-à-dire des *langages*) permettant de décrire des problèmes réels et complexes sous forme de programmes. Ces programmes sont ensuite analysés et traduits automatiquement vers des modèles sur lesquels opèrent les algorithmes de vérification.

Pour mener à bien la vérification de systèmes complexes (de taille "industrielle"), il nous semble nécessaire de maîtriser simultanément la technologie des modèles et celle des langages.

## 2.2 Technologie des modèles – vérification

Par vérification, on entend la comparaison d'un système avec ses *propriétés*, qui décrivent les services rendus par le système et son fonctionnement attendu, à un certain niveau d'abstraction.

Les techniques de vérification que nous mettons en œuvre reposent en grande partie sur le modèle des *systèmes de transitions étiquetées* (ou, plus simplement, *automates*, ou encore *graphes*) composés d'un ensemble d'états, d'un état initial, et d'une relation de transition entre ces états. Ces techniques consistent à engendrer automatiquement, à partir de la description du système à vérifier, un graphe fini qui en modélise le comportement, puis à vérifier les propriétés sur le graphe grâce à une procédure de décision.

Selon le formalisme utilisé pour exprimer les propriétés, on distingue deux approches :

**propriétés comportementales :** elles décrivent le fonctionnement du système sous forme d'automates (ou bien en utilisant un langage de plus haut niveau que l'on traduit ensuite vers des automates). Compte-tenu que le système à vérifier et ses propriétés comportementales peuvent tous deux être représentés par des automates, la vérification consiste à les comparer au moyen de *relations d'équivalence ou de préordre*.

Concernant la vérification de propriétés comportementales, nous n'effectuons pas de recherches dans ce domaine, mais nous collaborons avec d'autres projets, notamment Spectre/Verimag et Meije, qui développent des outils basés sur les relations d'équivalence et de préordre.

**propriétés logiques :** elles caractérisent des propriétés essentielles du système, telles que l'absence de blocage, l'exclusion mutuelle ou l'équité. Parmi les formalismes utilisés, les *logiques temporelles* et le *$\mu$-calcul modal* s'avèrent bien adaptés pour décrire l'évolution du système dans le temps. Dans ce cas, la vérification consiste à s'assurer que l'automate modélisant le système à vérifier satisfait les propriétés logiques.

Concernant la vérification de propriétés logiques, nos travaux dans ce domaine portent sur l'extension du $\mu$-calcul arborescent par des variables typées, afin de prendre en compte les données contenues dans les états et les transitions du graphe. Cette extension (dont nous avons mis en évidence l'utilité sur de nombreux exemples, notamment industriels) permet d'exprimer des propriétés qu'il n'est pas possible d'écrire en $\mu$-calcul standard comme, par exemple, le fait qu'une variable donnée soit toujours croissante sur un chemin d'exécution. Notre objectif est également de proposer et d'implémenter des algorithmes d'évaluation efficaces pour cette extension du $\mu$-calcul.

Bien que ces techniques soient efficaces et complètement automatisables, leur principale limitation est leur complexité, en général exponentielle par rapport à la taille du système. Ceci explique le problème de l'*explosion d'états* qui survient lorsque le nombre d'états du système à vérifier dépasse les capacités en mémoire de la machine. C'est pourquoi nous fournissons des technologies logicielles (voir 4.1) permettant de manipuler ces graphes de deux manières :

- soit sous forme *explicite*, en gardant en mémoire l'ensemble des états et des transitions (vérification énumérative),
- soit sous forme *implicite*, en explorant dynamiquement les parties du graphe en fonction des besoins (vérification à la volée).

### 2.3 Technologie des langages – compilation

En ce qui concerne les langages, il nous semble essentiel de s'appuyer sur des langages possédant simultanément un *caractère exécutable* et une *sémantique formelle*, ceci pour plusieurs raisons :

- la modélisation de systèmes critiques ne saurait reposer sur des langages dont la sémantique ne serait pas rigoureusement définie ;
- les techniques de "model-checking" nécessitent de pouvoir exécuter efficacement les programmes à vérifier ;
- les techniques de preuve, indispensables pour assurer la correction totale d'un système (ce qui n'est, en général, pas garanti par les méthodes de model-checking, qui n'effectuent qu'une vérification partielle) ne peuvent s'appliquer qu'aux langages ayant une sémantique formelle.

C'est pourquoi nous nous intéressons au langage LOTOS, le seul langage de description de protocoles ayant le statut de norme internationale [5] et possédant les propriétés ci-dessus. Il s'agit d'un langage basé sur les concepts des algèbres de processus (notamment CCS [6] et CSP [4]) pour la description du contrôle et les types abstraits algébriques [2] pour la description des données. LOTOS autorise à la fois la description du parallélisme asynchrone (aspects liés à la répartition, la synchronisation et la communication entre tâches) et celle des structures de données complexes manipulées dans les protocoles et les systèmes distribués.

Toutefois, les besoins en méthodes formelles et vérification allant en augmentant, il est nécessaire de réfléchir à de nouveaux langages, plus simples, plus expressifs et mieux adaptés aux besoins industriels. Cette réflexion est également guidée par l'apparition de protocoles à contraintes temporelles fortes — protocoles utilisés dans les réseaux à haut débit — pour lesquels il devient nécessaire de prendre en compte les aspects temporels de manière quantitative, et non plus seulement qualitative. Nous travaillons dans cette direction, notamment dans le cadre de la refonte de la norme LOTOS actuellement entreprise à l'ISO.

### 2.4 Implémentation et expérimentation

Dans la mesure du possible, nous essayons de valider nos propositions par le développement d'outils et l'application de ces outils à des études de cas complexes (notamment industrielles, dans le cadre de notre coopération avec le le GIE BULL-INRIA DYADE. Cette confrontation systématique avec les problèmes d'implémentation et d'expérimentation est un aspect essentiel de notre approche.

## 3 Domaines d'applications

Les modèles théoriques que nous utilisons (automates, algèbres de processus, bisimulations, logiques temporelles) et les logiciels que nous développons sont suffisamment généraux pour ne pas dépendre trop étroitement d'un seul secteur applicatif.

Nos méthodes peuvent s'appliquer à tout système ou protocole composé d'agents distribués communiquant par messages. Ce cadre conceptuel trouve de nombreuses

instantiations dans le domaine du logiciel, du matériel et des télécommunications. Les études de cas récemment conduites avec la boîte à outils CADP illustrent bien cette diversité applicative :

- **architectures multiprocesseurs :** arbitrage de bus, cohérence de caches ;
- **bases de données :** protocoles transactionnels, bases de connaissances distribuées ;
- **électronique de consommation :** télécommandes audiovisuelles, vidéo à la demande, bus FIREWIRE ;
- **protocoles de sécurité :** commerce électronique, distribution de clés cryptographiques ;
- **systèmes embarqués :** communications entre avions et tours de contrôle ;
- **systèmes répartis :** mémoire virtuelle, systèmes de fichiers répartis, ingénierie concourrante, algorithmes d'élection ;
- **télécommunications :** réseaux à haut débit, administration de réseaux, interactions de services téléphoniques.

# 4 Logiciels

## 4.1 La boîte à outils CADP

En collaboration avec le projet SPECTRE/VERIMAG, nous développons la boîte à outils CADP (CÆSAR/ALDÉBARAN DEVELOPMENT PACKAGE) pour l'ingénierie des protocoles et des systèmes distribués. Au sein de cette boîte à outils, nous avons en charge les logiciels suivants :

- CÆSAR est un compilateur qui produit, à partir d'un programme LOTOS, du code exécutable ou des modèles sur lesquels différentes méthodes de vérification peuvent être appliquées. Le programme source LOTOS est traduit successivement en une algèbre de processus simplifiée, un réseau de Petri étendu avec des variables et des transitions atomiques, et finalement un système de transitions étiquetées obtenu par simulation exhaustive.
- CÆSAR.ADT est un compilateur qui traduit les définitions de types abstraits LOTOS vers des bibliothèques de types et de fonctions en langage C. La traduction met en œuvre un algorithme de compilation par filtrage et des techniques pour la reconnaissance des classes de types usuels (nombres entiers, énumérations, tuples, listes...) qui sont identifiées automatiquement et implémentées de manière optimale.
- BCG (*Binary Coded Graphs*) est un format qui utilise des techniques efficaces de compression permettant de stocker des graphes (sous forme explicite) sur disque de manière très compacte. Ce format est indépendant du langage source et des outils de vérification. En outre, il contient suffisamment d'informations pour que les outils qui l'exploitent puissent fournir à l'utilisateur des diagnostics précis dans les termes du programme source. Pour exploiter ce format, un environnement logiciel est disponible, qui se compose de bibliothèques C et de plusieurs outils, notamment : BCG_IO (qui effectue des conversions de format), BCG_OPEN (qui permet d'appliquer à des graphes BCG les outils de l'environnement OPEN/CÆSAR pour la vérification à la volée), BCG_DRAW (qui permet d'afficher en PostScript une

représentation 2D d'un graphe), et BCG_EDIT (qui permet de modifier interactivement la représentation graphique produite par BCG_DRAW).

- OPEN/CÆSAR est un environnement extensible permettant de développer des outils de simulation, de vérification et de génération de test sur des graphes sous forme implicite. Ces outils peuvent être réalisés de manière simple, modulaire et indépendante du langage utilisé pour décrire les systèmes à valider. L'environnement OPEN/CÆSAR comprend un ensemble de bibliothèques avec leurs interfaces de programmation, ainsi que divers outils pour la simulation pas à pas, l'exécution aléatoire, la recherche de blocages, la recherche de séquences satisfaisant un certain critère, etc.
- XTL (*eXecutable Temporal Language*) est un méta-langage adapté à l'expression des algorithmes d'évaluation et de diagnostic pour les formules de logiques temporelles telles que CTL [1], HML [3], ACTL [7], etc. D'inspiration fonctionnelle, ce méta-langage offre des primitives d'accès à toutes les informations contenues dans les graphes BCG : états, étiquettes des transitions, fonctions *successeurs* et *prédécesseurs*, ainsi qu'aux types et fonctions du programme source. Il permet la définition de fonctions récursives servant à calculer des prédicats de base et des modalités temporelles portant sur les ensembles d'états et de transitions.

A ces outils s'ajoutent ceux développés par le projet SPECTRE/VERIMAG, qui permettent la comparaison et la réduction de graphes modulo des relations d'équivalences et de préordres appropriées, la génération compositionnelle de graphes par application progressive de réductions et d'abstractions, et la génération de tests à la volée.

Tous ces outils — ainsi que d'autres développés par les projets MEIJE (Sophia-Antipolis) et PAMPA (Rennes) et par les Universités de Liège et d'Ottawa — sont intégrés au sein de l'interface graphique EUCALYPTUS (développée en TCL/TK) qui offre un accès facile et uniforme aux différents outils, en cachant à l'utilisateur les conventions d'appel et les formats spécifiques à chaque outil.

## 4.2 Diffusion de logiciels

Depuis les années 80 et le début des années 90, les méthodes formelles ont connu un grand essor : de multiples langages, outils et méthodologies ont été proposés. Cette phase d'expansion semble sur le point de s'achever et une phase de sélection darwinienne s'apprête à lui succéder. Les langages inadaptés et les prototypes immatures seront délaissés, au profit d'outils qui auront fait leurs preuves sur des exemples industriels et pour lesquels l'existence d'une communauté importante d'utilisateurs permettra d'assurer les développements futurs.

La boîte à outils CADP est bien placée dans cette compétition. Elle s'appuie sur un langage normalisé, comporte des outils robustes (bien que perfectibles) et regroupe un nombre important d'utilisateurs.

En 1997, nous avons accompli un effort significatif en vue d'améliorer la visibilité de la boîte à outils CADP et d'accroître sa diffusion. Nous pouvons mentionner les faits marquants suivants (dont certains obtenus en collaboration avec le projet SPECTRE/VERIMAG, notamment Marius Bozga et Laurent Mounier) :

- Dans le cadre du travail de stage de P. Wendel, nous avons simplifié la procédure d'installation de CADP en développant l'outil INSTALLATOR qui automatise l'installation et la mise à jour de CADP grâce aux protocoles de l'INTERNET. Avec cet outil, nous avons cherché à atteindre l'ergonomie et la simplicité d'utilisation propres aux "assistants d'installation" du monde PC/WINDOWS.
- Début 1997, nous avons créé une page Web consacrée à la boîte à outils CADP (`http ://www.inrialpes.fr/vasy/cadp.html`). Tout au long de l'année, nous avons enrichi cette page, qui offre désormais les services suivants :
  - présentation des outils et de leurs fonctionnalités ;
  - documentations et publications associées ;
  - forum aux questions (FAQ, en anglais *Frequently Asked Questions*) ;
  - liste des études de cas réalisées avec CADP ;
  - démonstrations disponibles en ligne.
- En juin 1997, nous avons diffusé une nouvelle version de la boîte à outils : CADP version 97a "Twente", qui remplace la précédente version Z datant de décembre 1996.
- En décembre 1997, nous avons diffusé une nouvelle version : CADP version 97b "Liège", qui a été portée sous le système d'exploitation LINUX.
- Nous avons effectué plusieurs démonstrations publiques de la boîte à outils, notamment à l'occasion des colloques et conférences DYADE (Grenoble, avril 1997), COST-247 (Zagreb, juin 1997), ERCIM (Cesena, juillet 1997), CFIP'97 (Liège, octobre 1997) et ASE'97 (Incline Village, novembre 1997).
- Nous avons organisé le 24 juillet 1997 une journée de présentation des outils CADP à l'intention du monde académique et industriel, qui a réuni une quinzaine de participants.
- Nous avons également créé une lettre électronique (*The CADP Newsletter*) décrivant l'avancement des travaux et les nouveautés concernant CADP. Trois numéros ont été publiés en décembre 1996, juin 1997 et septembre 1997.

Ces efforts semblent avoir porté leurs fruits puisqu'en 1997, le nombre de licences sites pour CADP est passé de 127 à 148.

# 5 Résultats nouveaux

## 5.1 Amélioration des techniques de compilation et de vérification

*Mots-clés* : algèbre de processus, automate, bisimulation, compilation, émulation logicielle, génération de code, logique temporelle, parallélisme asynchrone, programmation parallèle, synchronisation, vérification de programme.

**Résumé :** *En 1997, nos travaux ont porté sur trois axes : (i) amélioration de nos compilateurs LOTOS (ajout de nouvelles fonctionnalités et accroissement des performances) ; (ii) définition et implémentation du langage XTL pour la vérification de propriétés exprimées en logique temporelle et $\mu$-calcul modal ; (iii) connexion des outils CADP avec les outils de vérification Fc2Tools développés par le projet Meije.*

### 5.1.1 Compilation du langage LOTOS

*Participants* : Hubert Garavel, Mark Jorgensen

Une partie importante de nos travaux est consacrée au traitement d'études de cas de complexité significative (notamment dans le cadre de notre collaboration avec BULL), pour lesquelles nous utilisons le langage LOTOS et la boîte à outils CADP. Il est donc important de maintenir et d'améliorer ces outils afin de répondre aux problèmes et besoins nouveaux constatés [12].

En 1997, outre diverses corrections d'anomalies et modifications ponctuelles, nous pouvons mentionner trois améliorations importantes (faites à la suggestion de nos partenaires de BULL) concernant les outils dédiés au langage LOTOS :

- Le compilateur CÆSAR.ADT, qui traduit des types abstraits algébriques LOTOS en langage C, permet d'importer et de manipuler, depuis un programme LOTOS des types et des fonctions externes, écrits directement en C. La nouvelle version de CÆSAR.ADT est désormais capable de produire un "squelette" des déclarations en langage C de ces types et fonctions externes, que l'utilisateur doit ensuite compléter manuellement.
- Initialement conçu pour la vérification exhaustive, puis adapté à la vérification à la volée (interface OPEN/CÆSAR), le compilateur CÆSAR a été étendu en 1997 pour permettre la génération de "code embarqué". Une nouvelle interface de programmation, nommée EXEC/CÆSAR, a été développée, qui permet d'interfacer le code C produit par les compilateurs CÆSAR et CÆSAR.ADT avec un environnement extérieur réel. Dans cette approche, chaque rendez-vous sur une porte visible du programme LOTOS se traduit par l'appel d'une fonction C externe (écrite à la main par l'utilisateur) avec un passage de paramètres correspondant aux entrées/sorties effectuées pendant le rendez-vous. Cette nouvelle fonctionnalité a été utilisée dans deux études de cas :
  - Dans le cas de la cellule de production (voir § 5.3.4), EXEC/CÆSAR a permis de piloter une interface graphique TCL/TK modélisant le fonctionnement réel d'une usine de production métallurgique.
  - Dans le cas de l'architecture multiprocesseur POLYKID de BULL (voir § 6.1), EXEC/CÆSAR a permis à G. Chehaibar de réaliser un prototype d'émulation logicielle d'un circuit contrôleur de cache distant. Le code C produit à partir de la description en LOTOS du circuit (3300 lignes environ) a pu être embarqué sur une carte à processeur POWERPC et interfacé avec l'extérieur (au moyen d'interruptions) afin de simuler le fonctionnement du circuit dans son environnement. Bien que fonctionnant correctement, cet émulateur logiciel n'avait pas une rapidité suffisante pour l'environnement de test industriel de BULL. C'est pourquoi, nous avons également cherché à améliorer les performances du code C engendré à partir de programmes LOTOS.
- Nous avons analysé finement les performances du code C produit par CÆSAR. Ceci nous a conduit à proposer 10 améliorations de ce code, qui ont été implémentées et ont eu pour effet de diviser par un facteur 9 en moyenne le temps nécessaire pour le franchissement des transitions. Par ailleurs, nous avons redimensionné les tables d'états utilisées, afin de tirer parti des capacités mémoires des ordinateurs actuels. Combinées, ces différentes modifications ont conduit à une réduction

significative du temps nécessaire à la vérification exhaustive (selon les exemples traités, la rapidité du compilateur CÆSAR a été multipliée par un facteur variant entre 2 et 160).

### 5.1.2 Evaluateur générique XTL pour le $\mu$-calcul modal

*Participants* : Hubert Garavel, Radu Mateescu, Charles Pecheur

XTL (*eXecutable Temporal Language*, voir § 4.1) est à la fois un langage et un outil permettant de décrire et de valider des propriétés temporelles des systèmes. En 1997, le développement de la première version d'XTL s'est poursuivi dans le cadre du travail de thèse de R. Mateescu :

- L'outil XTL a été sensiblement amélioré et sa version 1.1 est désormais intégrée à la boîte à outils CADP.
- Les bibliothèques XTL permettant d'obtenir les logiques temporelles arborescentes classiques (HML, CTL, ACTL et LTAC) ont été finalisées et testées. Une nouvelle bibliothèque implémentant les opérateurs du $\mu$-calcul modal a été ajoutée.
- Cette première version d'XTL a été utilisée avec succès dans trois applications industrielles : la validation du protocole BRP de Philips [11], celle du bus série IEEE-1394 "Firewire" (voir § 5.3.2) et celle du système de mémoire virtuelle distribuée CFS (voir § 5.3.3).
- Pour les besoins de cette dernière application, Ch. Pecheur a défini une nouvelle gamme d'opérateurs XTL permettant de générer des traces de diagnostic expliquant la valeur de vérité des formules évaluées, à la manière de l'outil EXHIBITOR mais avec une expressivité accrue.

Par ailleurs, sur la base de l'expérience accumulée, la définition de la seconde version d'XTL a été achevée [Mat98] :

- Cette seconde version ajoute au langage XTL les opérateurs de point fixe et les modalités du $\mu$-calcul. Par rapport au $\mu$-calcul standard, les opérateurs de point fixe d'XTL sont paramétrés par des variables et des valeurs typées ; de même, les modalités sont doublement étendues par un mécanisme de filtrage (ou *pattern-matching*, afin de récupérer les valeurs échangées sur les canaux de communication pour les stocker dans des variables) et par un langage d'expressions régulières (afin de permettre l'expression aisée de contraintes de séquencement).
- La syntaxe et la sémantique statique du langage XTL version 2.0 ont été finalisées. Un analyseur lexico-syntaxique a été réalisé en utilisant le système SYNTAX développé à l'INRIA Rocquencourt (notamment par Pierre Boullier), qui a permis de valider les exemples de formules XTL figurant dans la thèse de R. Mateescu.
- La sémantique dynamique du langage XTL version 2.0 a été formellement définie (de manière dénotationnelle).
- Nous avons proposé divers algorithmes d'évaluation des formules XTL sur des modèles finis. Dans le cas particulier des formules XTL de degré d'alternance égal à 1 (c'est-à-dire ne contenant pas d'opérateurs de plus petit et de plus grand

[Mat98] Radu Mateescu. *Vérification des propriétés temporelles des programmes parallèles.* Thèse de doctorat, Institut National Polytechnique de Grenoble, 1998. à paraître.

point fixe mutuellement récursifs), nos algorithmes sont basés sur la traduction vers des systèmes d'équations booléennes paramétrées par des variables typées, qui sont ensuite résolus de manière soit globale, soit locale (à la volée). Dans le cas général où le degré d'alternance est quelconque, nos algorithmes généralisent les algorithmes globaux existants pour l'évaluation des formules de $\mu$-calcul standard au cas des formules XTL ($\mu$-calcul étendu avec valeurs).

Tout en conservant les fonctionnalités de la première version d'XTL (manipulation aisée des valeurs contenues dans les états et les actions des modèles, et définition concise de nouveaux opérateurs temporels non-standards), la seconde version d'XTL autorise l'écriture de formules avec les notations mathématiques usuelles de la logique modale et l'évaluation efficace des formules de $\mu$-calcul étendu à l'aide d'algorithmes spécialisés pouvant fonctionner à la volée.

Il convient de souligner qu'à notre connaissance, il n'existe actuellement aucun autre outil de vérification réunissant les fonctionnalités offertes par la seconde version d'XTL.

### 5.1.3 Intégration des outils du projet Meije

*Participants* : Hubert Garavel, David Jacquemin, Mark Jorgensen

Nous avons établi une collaboration active avec le projet MEIJE à Sophia-Antipolis (notamment avec Amar Bouali, Khalid Laksiouar et Robert de Simone) afin de permettre l'interconnexion des outils AUTOGRAPH et FC2TOOLS [BRRd96] développés par le projet MEIJE avec nos propres outils. Outre l'aspect positif du rapprochement entre deux équipes de l'INRIA travaillant dans le domaine de la vérification, cette collaboration répond à un double objectif pratique :

- Pour vérifier des systèmes *asynchrones* (spécifiés en LOTOS, par exemple), il est intéressant de pouvoir disposer des outils FC2TOOLS en sus de l'outil ALDEBARAN développé par le projet SPECTRE/VERIMAG. En effet, notre expérience montrant que les outils de vérification basés sur les bisimulations ne sont pas toujours exempts d'erreurs, il est souhaitable (notamment dans un contexte industriel) d'assurer une redondance en utilisant des outils d'origines différentes. Ceci donne de meilleures garanties de fiabilité sur les résultats de vérification obtenus et permet de comparer les performances respectives des différents outils.
- Pour vérifier des systèmes *synchrones* (spécifiés en ESTEREL, par exemple), les outils CADP apportent des fonctionnalités nouvelles (par exemple, l'évaluation de formules de logique temporelles) non disponibles dans les environnements dédiés aux langages synchrones. Ces fonctionnalités nouvelles pourraient notamment bénéficier à l'atelier logiciel ORCCAD pour le contrôle-commande de systèmes robotiques complexes (atelier développé par le projet BIP, le projet ICARE et le service robotique de l'INRIA Rhône-Alpes).

[BRRd96] Amar Bouali, Annie Ressouche, Valérie Roy, and Robert de Simone. The Fc2Tools set : a Toolset for the Verification of Concurrent Systems. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Science*. Springer Verlag, August 1996.

Techniquement, la connexion des outils CADP et FC2TOOLS s'est effectuée en plusieurs étapes, par l'établissement de passerelles entre les formats BCG et EXP utilisés par les outils CADP et le format FC2 [MS93] utilisé par les outils FC2TOOLS :

- Nous avons étendu l'outil BCG_IO (outil de conversion de formats de graphes) pour qu'il accepte en entrée des automates séquentiels au format FC2 et nous l'avons adapté pour qu'il produise en sortie un format FC2 compatible avec la dernière version des outils FC2TOOLS.
- Nous avons développé un nouvel outil, nommé EXP2FC2, qui convertit des réseaux d'automates communicants du format EXP vers le format FC2, permettant ainsi d'appliquer les outils FC2TOOLS à ces réseaux.
- En collaboration avec le projet SPECTRE/VERIMAG, nous avons étendu le format EXP (et les outils manipulant ce format) pour accepter des automates communicants encodés sous divers formats (notamment BCG et FC2).
- En collaboration avec le projet MEIJE, nous avons aidé au développement d'un nouvel outil nommé FC2OPEN, qui permet d'appliquer les outils OPEN/CÆSAR à des automates séquentiels et des réseaux d'automates communicants au format FC2.
- Nous avons intégré ces outils au sein de l'interface graphique EUCALYPTUS de manière unifiée et ergonomique : l'utilisateur peut ainsi lancer n'importe quel outil sans se soucier du format, l'interface EUCALYPTUS se chargeant d'appeler les outils appropriés pour effectuer les conversions nécessaires.
- Notre évaluation des outils FC2TOOLS (version 1.1) nous a conduit à formuler un certain nombre de remarques et de suggestions que nos collègues du projet MEIJE ont pris en compte dans la nouvelle version 1.2 de leurs outils.

## 5.2 Amélioration des formalismes de spécification

*Mots-clés* : algèbre de processus, compilation, concurrence, génération de code, génie logiciel, modélisation, parallélisme asynchrone, programmation parallèle, spécification formelle, synchronisation, temps réel, vérification de programme.

**Résumé :** *En 1997, nous avons activement contribué au groupe de travail sur la normalisation du langage E-LOTOS (Extended-LOTOS). Nous avons également réalisé un prototype de compilateur E-LOTOS en utilisant les outils Syntax/Fnc-2 développés à l'INRIA Rocquencourt.*

### 5.2.1 Contribution à la définition d'E-LOTOS

*Participants* : Hubert Garavel, Mihaela Sighireanu

Une révision de la norme LOTOS est actuellement en cours à l'ISO : elle devrait conduire à un nouveau langage, nommé E-LOTOS (*Extended*-LOTOS) adapté aux nouvelles générations de protocoles et de systèmes distribués. Comparé à LOTOS, le langage E-LOTOS devrait être plus facile à apprendre par des non-experts et avoir une plus grande expressivité (par exemple, avec l'introduction du temps quantifié).

[MS93] Eric Madelaine and Robert de Simone. *FC2 : Reference Manual Version 1.1.* INRIA, Sophia-Antipolis (France), July 1993.

Nous participons à ces travaux en tant que délégués AFNOR (France) et RSI (Roumanie). L'historique de nos contributions est disponible sur notre serveur Web, à l'adresse `http ://www.inrialpes.fr/vasy/elotos`. En 1997, nos principales contributions ont été les suivantes :

- Nous avons organisé la réunion ISO consacrée à E-LOTOS qui s'est tenue à Grenoble en décembre 1996, dans les locaux de l'UR Rhône-Alpes.
- Suite à cette réunion, nous avons activement participé à la production du *Committee Draft*, première version officielle de la future norme E-LOTOS [ISOry]. Ce document s'est inspiré de certaines de nos propositions concernant les exceptions (opérateur "trap"), l'opérateur de parallélisme généralisé, le typage des portes, les types constructifs, et le système de modules [16]. Nous avons aussi fourni la syntaxe concrète du nouveau langage.
- Nous avons contribué à l'élaboration des positions française et roumaine au moment du vote concernant ce document, notamment en signalant des erreurs et en proposant des améliorations du langage E-LOTOS.
- Ces positions ont été discutées et défendues au cours de la réunion suivante tenue à Helsinki en juillet 1997. Un certain nombre d'entre elles ont été retenues, qui vont dans le sens d'un style "impératif contrôlé" pour E-LOTOS : passage de paramètres "in/out", itérations, etc.
- Par ailleurs, nous avons proposé l'introduction d'un mécanisme de coroutines [15, 17] pour généraliser la notion d'interruption proposée par d'autres membres du groupe de normalisation.
- Nous avons contribué à l'élaboration du *Final Committee Draft* issu de la réunion d'Helsinki.

En parallèle, dans le cadre du travail de thèse de M. Sighireanu, nous étudions une variante de E-LOTOS (appelée LOTOS NT) dans laquelle nous avons introduit les concepts qui nous semblent pertinents (ce qui n'est pas toujours chose aisée dans une norme internationale). La différence essentielle entre les deux langages réside dans le fait que LOTOS NT est un langage impératif alors que E-LOTOS s'inscrit dans un cadre fonctionnel. De plus, LOTOS NT se distingue d'E-LOTOS sur certains aspects (surcharge d'opérateurs, tableaux, typage statique) qui en font un langage plus facile à implémenter.

La syntaxe, la sémantique statique et la sémantique dynamique de LOTOS NT ont été formellement définies.

### 5.2.2 Réalisation du compilateur TRAIAN

*Participants* : Mihaela Sighireanu, Bruno Vivien

Afin de valider nos propositions, nous avons entrepris l'implémentation d'un compilateur (appelé TRAIAN) pour le langage LOTOS NT [18].

Compte-tenu du fait que la définition de LOTOS NT n'est pas complètement figée, nous avons choisi d'utiliser des méta-outils pour autoriser les évolutions futures.

[ISOry] ISO. Committee Draft on Enhancements to LOTOS. ISO/IEC JTC1/SC21/WG7 Project 1.21.20.2.3, 1997 January.

C'est pourquoi nous avons basé nos développements sur le système de génération de compilateurs SYNTAX/FNC-2 développé à l'INRIA Rocquencourt (notamment par Pierre Boullier, Philippe Deschamp, Martin Jourdan et Didier Parigot). En tant qu'utilisateurs de ce système, nous avons entretenu une collaboration intense avec Didier Parigot pour surmonter les problèmes rencontrés. Notre expérience avec SYNTAX/FNC-2 s'avère concluante, surtout depuis le portage de cet outil sur les systèmes SOLARIS et LINUX, portage auquel nous avons contribué.

Le compilateur TRAIAN effectue l'analyse lexicale et syntaxique pour LOTOS NT, la construction d'arbre abstrait et les vérifications de sémantique statique. Sur cette souche, nous avons entrepris le développement d'un traducteur de LOTOS NT vers LOTOS permettant de réutiliser pour LOTOS NT les outils existants pour LOTOS. Actuellement, ce traducteur prend en compte les définitions de types constructifs.

En utilisant l'outil PARADIS développé par Philippe Deschamp et Pierre Boullier, nous avons réalisé deux paragrapheurs ("pretty-printer"), l'un pour LOTOS (cet outil, nommé CÆSAR.INDENT a été intégré à la boîte à outils CADP) et l'autre pour LOTOS NT.

Enfin, nous avons étudié le problème de l'implémentation directe de LOTOS NT via une traduction vers un modèle "Réseaux de Petri temporisés".

## 5.3 Etudes de cas et applications pratiques

*Mots-clés* : application critique, algorithme réparti, application répartie, architecture multiprocesseur, architecture parallèle, atomicité, automate, base de connaissance, cohérence de caches, concurrence, génération de code, génération de test, logique temporelle, mémoire répartie, mémoire technique, parallélisme asynchrone, protocole de communication, spécification formelle, système distribué, temps réel, travail coopératif, vérification de programme.

**Résumé :** *Nous accordons une grande importance au traitement d'exemples réalistes qui nous permet de vérifier l'adéquation de nos méthodes et outils, et d'identifier de nouvelles orientations de recherche pour résoudre les problèmes rencontrés. En 1997, nous avons traité cinq études de cas dans des domaines très divers : architectures matérielles, protocoles de bus, systèmes distribués, production automatisée et bases de connaissances.*

### 5.3.1 Protocole de cohérence de caches CC-NUMA "Polykid"

*Participants* : Ghassan Chehaibar, Massimo Zendri

Dans le cadre du GIE DYADE, nous avons appliqué nos méthodes et outils au protocole de cohérence de caches de l'architecture CC-NUMA "POLYKID" actuellement développée par BULL. Ces travaux sont présentés au paragraphe 6.1.

### 5.3.2 Couche liaison du bus série IEEE-1394 "Firewire"

*Participants* : Radu Mateescu, Mihaela Sighireanu

Le bus FireWire (norme Ieee 1394) est un bus série à haut débit pour les ordinateurs personnels (Pc), auquel se sont ralliés les principaux constructeurs informatiques, éditeurs de logiciels et fabricants d'équipement audiovisuel. Dans le cadre du projet pan-européen Cost 247, le protocole de transmission asynchrone de la couche liaison du bus FireWire a été proposé par Jan-Friso Groote (Cwi, Amsterdam) comme un défi à relever pour les méthodes formelles et les outils de vérification automatique.

Partant d'une description des couches physique et liaison dans l'algèbre de processus $\mu$-Crl produite par Bas Luttik (Cwi)[Lut97], nous avons produit une description en E-Lotos (voir § 5.2.1) de ces deux couches, que nous avons complétée par une description de la couche transaction.

Nous avons ensuite traduit cette description E-Lotos en Lotos. En utilisant le compilateur Cæsar, nous avons produit plusieurs modèles du protocole correspondant à différents scénarios. Finalement, nous avons spécifié quatre propriétés de bon fonctionnement en logique temporelle Actl[NV90], que nous avons analysées sur les différents modèles grâce à l'outil Xtl (voir § 5.1.2).

Ceci nous a permis de découvrir et de corriger une erreur (réception non spécifiée de message) due à la sémantique ambiguë des machines d'états utilisées dans la norme Ieee, erreur susceptible de provoquer des blocages dans les implémentations du bus Firewire.

Ce travail a fait l'objet d'une publication [14]. L'erreur que nous avons détectée a été confirmée par Philips Research (Eindhoven, Pays Bas) et a été officiellement signalée au consortium "1394 Trade Association".

### 5.3.3 Système de mémoire virtuelle distribuée CFS

*Participants* : Charles Pecheur

Nous avons spécifié en Lotos et vérifié le système de fichiers réparti Cfs (*Cluster File System*), développé au sein du projet Sirac (action Mescaline du Gie Dyade). Ce système de fichiers est implémenté au-dessus d'Arias, un service de mémoire virtuelle répartie pour une grappe de machines homogènes reliées par un réseau local à haut débit.

Le travail de modélisation, accompli en collaboration étroite avec les développeurs de Cfs (notamment Thierry Jacquin de Bull) a permis de formaliser finement les aspects de synchronisation du système.

[Lut97] Bas Luttik. Description and Formal Specification of the Link Layer of P1394. In Ignac Lovrek, editor, *Proceedings of the 2nd COST 247 International Workshop on Applied Formal Methods in System Design (Zagreb, Croatia)*, June 1997.

[NV90] R. De Nicola and F. W. Vaandrager. *Action versus State based Logics for Transition Systems*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer Verlag, 1990.

En utilisant les outils CADP selon une approche compositionnelle, il a été possible d'engendrer un modèle complet pour un système comportant trois sites CFS. Sur ce modèle, l'utilisation de l'outil XTL a montré l'absence de blocage et a permis de formaliser et de vérifier les propriétés de cohérence de CFS comme des formules de logique temporelle ACTL.

### 5.3.4 Cellule de production métallurgique automatisée

*Participants* : Hubert Garavel, Mark Jorgensen

Proposée par Klaus Lewerentz et Thomas Lindner (FZI, Allemagne), la "cellule de production" constitue un exemple canonique pour l'application des méthodes formelles [LL95] : le problème consiste à modéliser le fonctionnement d'une usine métallurgique réelle, comportant de nombreux robots et automatismes.

Pour cet exemple, nous avons repris une description en LOTOS de la cellule de production élaborée en 1994 par H. Garavel. Celle-ci a été corrigée et adaptée afin de valider les nouvelles fonctionnalités de génération de code C récemment introduites dans CÆSAR (voir § 5.1.1).

Ce travail nous a permis d'obtenir un programme exécutable capable de piloter un simulateur graphique en TCL/TK représentant le fonctionnement des divers automatismes de l'usine. Outre la validation du générateur de code C, cet exemple constitue une démonstration attractive des possibilités offertes par la boîte à outils CADP.

### 5.3.5 Protocole de gestion de bases de connaissances distribuées Co4

*Participants* : Charles Pecheur

Nous avons spécifié en LOTOS et vérifié le protocole Co4 pour la gestion de bases de connaissances distribuées. Développé au sein du projet SHERPA, ce protocole permet la construction incrémentale et coopérative de bases de connaissance partagées entre plusieurs utilisateurs localisés sur des sites distants. Afin d'assurer l'intégrité des bases, ces utilisateurs communiquent suivant un protocole de décision consensuelle inspiré des procédures d'évaluation de publications scientifiques (*peer reviewing*)

En collaboration avec le projet SHERPA (Jérome Euzenat et Loïc Tricand de la Goutte), nous avons formellement spécifié en LOTOS le comportement du protocole Co4, jusque là défini de manière semi-formelle par un ensemble de règles d'inférences [Euz95]. Ce travail de modélisation a mis en évidence diverses omissions, imprécisions et erreurs ponctuelles dans la définition de Co4. Il a également permis d'obtenir un simulateur interactif correspondant au comportement du protocole.

[LL95] Claus Lewerentz and Thomas Lindner, editors. *Formal Development of Reactive Systems – Case Study Production Cell*, volume 891 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, January 1995.

[Euz95] Jérôme Euzenat. Building Consensual Knowledge Bases : Context and Architecture. In *Proceedings of the 2nd International Conference on Building and Sharing Very Large-Scale Knowledge Bases (KBKS), Enschede the Netherlands*, pages 143–155, 1995.

Ensuite, l'utilisation des outils CADP a permis d'analyser le fonctionnement du protocole dans différents scénarios. Nous avons ainsi détecté diverses erreurs (violation de la structure hiérarchique des bases, réceptions de messages non prévues, etc.) et confirmé certaines limitations (possibilité d'introduire des connaissances contradictoires dans une même base). Ce travail a conduit à une définition révisée du protocole Co4[Euz97] et à une publication [13].

### 5.3.6 Autres études de cas

Nos travaux antérieurs sur les algorithmes d'élection dans un réseau en anneau ont fait l'objet d'une publication [10]. D'autres équipes ont également utilisé nos outils pour diverses études de cas. Pour ne citer que les travaux publiés, on peut mentionner :

- la modélisation et la validation du protocole MPI-BIP (*Message Passing Interface – Basic Interface for Parallelism*) de la plate-forme réseau MYRINET (projet REMAP et Laboratoire pour les Hautes Performances en Calcul, Lyon, France) [Her97] ;
- la détection de conflits (*feature interactions*) dans les services téléphoniques (CWI, Amsterdam, Pays-Bas) [Kor97] ;
- la mise en défaut du protocole de sécurité EQUICRYPT basé sur le principe des "tiers de confiance" et destiné à contrôler l'accès à des services multimédia (Université de Liège, Belgique) [GL97a,GL97b] ;
- la mise en défaut d'un protocole de dialogue entre un avion et la tour de contrôle pour l'automatisation du traffic aérien (Université Libre de Bruxelles, Belgique, en coopération avec l'institut EUROCONTROL) [dJMH97] ;
- la conception d'une plate-forme d'inter-opérabilité pour l'administration de réseaux hétérogènes (UFSC Florianopolis, Brésil) [NRL+97].

---


[Euz97] Jérôme Euzenat. A Protocol for Building Consensual and Consistent Repositories. Research Report RR-3260, INRIA, September 1997.

[Her97] Marc Herbert. Evaluation de performances et spécification formelle sur un réseau de stations haut débit. mastère de l'institut national des télécommunications, Laboratoire pour les hautes performances en calcul, Lyon, December 1997.

[Kor97] Henri Korver. Detecting Feature Interactions with CÆSAR/ALDEBARAN. *Science of Computer Programming*, 29(1–2) :259–278, July 1997. Special issue on Industrially Relevant Applications of Formal Analysis Techniques.

[GL97a] François Germeau and Guy Leduc. A Computer Aided Design of a Secure Registration Protocol. In Teruo Higashino and Atsushi Togashi, editors, *Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'97 (Ozaka, Japan)*. IFIP, Chapman & Hall, November 1997.

[GL97b] François Germeau and Guy Leduc. Model-based Design and Verification of Security Protocols using LOTOS. In Hilarie Orman and Catherine Meadows, editors, *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols (Rutgers University, New Jersey, USA)*, September 1997.

[dJMH97] Anthony de Jacquier, Thierry Massart, and Christian Hernalsteen. Vérification et correction d'un protocole de contrôle aérien. TR 363, Université Libre de Bruxelles, May 1997.

[NRL+97] M.S.M.A. Notare, B.G. Riso, P.S. Lorena, M.C.O Penna, and C.B. Westphall. Formal Design of a Platform for Telecommunication Heterogeneous Network Management.

# 6 Actions industrielles

## 6.1 Action Vasy (Dyade)

*Mots-clés* : activité de conception, algorithme réparti, application répartie, architecture multiprocesseur, architecture parallèle, automate, cohérence de caches, compilation, génération de code, génération de test, mémoire répartie, modélisation, parallélisme asynchrone, programmation parallèle, protocole de communication, spécification formelle, synchronisation, système distribué, vérification de programme.

*Participants* : Ghassan Chehaibar, Hubert Garavel, Mark Jorgensen, Massimo Zendri

Cette action a pour objectif l'utilisation des méthodes formelles (langage LOTOS et outils associés) pour la validation et le test des architectures multiprocesseurs développées par BULL. Cette action, à laquelle participe également le projet PAMPA, est coordonnée par G. Chehaibar.

Après avoir vérifié la correction de l'arbitre de bus de l'architecture POWERSCALE mise en œuvre dans les stations de travail et serveurs de la gamme ESCALA [CGM+96], l'action se consacre maintenant à la vérification du protocole de cohérence de caches de POLYKID, une architecture multi-processeurs CC-NUMA (*Cache Coherent Non Uniform Memory Access*) en cours de conception par BULL.

Cette collaboration s'est trouvée renforcée par le regroupement de l'équipe dans les locaux de l'Unité de Recherche Rhône-Alpes en décembre 1996, ainsi que par l'arrivée de M. Jorgensen en janvier 1997, puis de M. Zendri en juillet 1997. Les travaux effectués en 1997 ont comporté trois volets :

**Emulation logicielle :** G. Chehaibar a entièrement modélisé en LOTOS un élément essentiel de l'architecture POLYKID, le contrôleur de cache distant RCC (*Remote Cache Controller*). Cette description LOTOS a ensuite été compilée vers du code C en utilisant les compilateurs CÆSAR et CÆSAR.ADT. Pour des raisons de performances, la partie comportementale de cette description LOTOS a été ensuite réécrite en C, la partie données demeurant inchangée. Le programme ainsi obtenu a été inséré dans un environnement matériel à la place du circuit ASIC implémentant le contrôleur RCC.

Pour BULL, cette technique visant à tester l'architecture POLYKID avant même que le circuit RCC ne soit disponible, a permis d'identifier très tôt 3 défauts

In A. Seneviratne and V. Varadarajan, editors, *Proceedings of the 8th IFIP/IEEE International Workshop on Distributed Systems Operations and Management DSOM'97 (Sydney, Australia)*, October 1997.

[CGM+96] Ghassan Chehaibar, Hubert Garavel, Laurent Mounier, Nadia Tawbi, and Ferruccio Zulian. Specification and Verification of the PowerScale Bus Arbitration Protocol : An Industrial Experiment with LOTOS. In Reinhard Gotzhein and Jan Bredereke, editors, *Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification FORTE/PSTV'96 (Kaiserslautern, Germany)*, pages 435–450. IFIP, Chapman & Hall, October 1996. Full version available as INRIA Research Report RR-2958.

de conception, qui ont été corrigés dans l'implémentation matérielle du RCC. Ce travail s'est vu décerner le Prix *Bull R&D Award* destiné à récompenser l'innovation technique.

Pour l'INRIA, cette expérience a permis de développer et d'expérimenter le compilateur CÆSAR en mode "génération de code" (EXEC/CÆSAR) et de le confronter aux sévères contraintes de performances requises par l'émulation logicielle d'un composant matériel (nécessité de franchir 100 000 transitions par seconde) : les performances de CÆSAR ont pu être multipliées par un facteur important (entre 10 et 150 selon les exemples).

**Modélisation et vérification formelle :** G. Chehaibar a modélisé formellement en LOTOS un sous-ensemble de l'architecture POLYKID comportant deux processeurs. En elle-même, cette étape de modélisation a soulevé 20 problèmes dans la description informelle du protocole. Sur ces 20 erreurs, 8 correspondaient à de "vraies" erreurs comportementales, les autres problèmes étant essentiellement des ambiguïtés ou des spécifications incomplètes.

La phase de vérification formelle, basée sur les outils de CADP, a permis de reproduire certaines erreurs découvertes pendant la modélisation (blocages, perte de cohérence dans les caches) et de montrer que les diverses corrections apportées par BULL supprimaient ces mêmes erreurs. En particulier, cette vérification formelle a permis de tester les outils de vérification compositionnelle[KM97] récemment introduits par nos collègues du projet SPECTRE/VERIMAG.

Les résultats de cette étude sont considérés comme très positifs par BULL, car certaines des erreurs détectées auraient été très difficiles à identifier avec une approche plus "classique" basée sur la simulation et le test. Pour des systèmes de cette complexité, les méthodes formelles procurent une aide efficace pour la mise au point du protocole et l'amélioration de la qualité du produit final.

**Génération de tests :** M. Zendri a expérimenté l'outil TGV [FJJ+97] développé par les projets PAMPA et SPECTRE/VERIMAG. Conduite sur deux études de cas (bus SCSI-2 et POLYKID), cette expérimentation a mis en évidence l'intérêt de la génération automatique de tests, mais aussi la nécessité d'étendre les fonctionnalités de TGV et de l'adapter à l'environnement industriel de test utilisé par BULL.

---

[KM97] Jean-Pierre Krimm and Laurent Mounier. Compositional State Space Generation from Lotos Programs. In Ed Brinksma, editor, *Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands)*, volume 1217 of *Lecture Notes in Computer Science*, Berlin, April 1997. Springer Verlag.

[FJJ+97] Jean-Claude Fernandez, Claude Jard, Thierry Jéron, Laurence Nedelka, and César Viho. An Experiment in Automatic Generation of Test Suites for Protocols with Verification Technology. *Science of Computer Programming*, 29(1–2) :123–146, July 1997. Special issue on Industrially Relevant Applications of Formal Analysis Techniques.

### 6.2 Autres actions industrielles

*Participants* : Charles Pecheur

Grâce au travail post-doctoral de Ch. Pecheur, notre équipe a indirectement participé à deux autres actions industrielles, en contribuant à la validation et à l'amélioration de deux applications réparties :

**Protocole Co4 pour la gestion de bases de connaissances :** ce protocole conçu au sein du projet SHERPA devrait être utilisé dans le cadre du contrat Storia pour conserver la mémoire technique d'entreprise de SGS-THOMSON ;

**Système de fichiers répartis CFS :** conçu au sein du projet Sirac, ce système pourrait être industrialisé par BULL, dans le cadre de l'action MESCALINE du GIE DYADE.

## 7 Actions régionales, nationales et internationales

### 7.1 Actions nationales

Au sein de l'INRIA, nous collaborons avec plusieurs projets :

**MEIJE (Sophia-Antipolis) :** utilisation des outils de vérification FC2TOOLS développés par MEIJE, suggestions pour l'amélioration de ces outils, et interconnexion de ces outils avec CADP ;

**OSCAR (Rocquencourt) :** utilisation de l'outil FNC-2 développé par OSCAR, suggestions pour l'amélioration de cet outil et contribué à son portage sous les systèmes SOLARIS et LINUX ;

**PAMPA (Rennes) :** collaboration au sein du GIE DYADE portant sur l'utilisation et l'amélioration de l'outil TGV développé par PAMPA et SPECTRE/VERIMAG ;

**REMAP (Rhône-Alpes) :** application de nos méthodes et outils pour valider le protocole MPI-BIP de la plate-forme MYRINET développée par REMAP ;

**SHERPA (Rhône-Alpes) :** application de nos méthodes et outils pour valider le protocole CO4 développé par SHERPA ;

**SIRAC (Rhône-Alpes) :** application de nos méthodes et outils pour valider le système de fichiers CFS développé par SIRAC (action MESCALINE du GIE DYADE) ;

**SPECTRE (Rhône-Alpes) :** collaboration autour de la boîte à outils CADP développée conjointement avec SPECTRE/VERIMAG.

### 7.2 Actions financées par la Commission Européenne

Nous participons à l'action pan-européenne COST 247 "*Verification and Validation Methods for Formal Descriptions*". Dans cette action, qui rassemble 18 pays, H. Garavel coordonne le groupe de travail n° 1 consacré à LOTOS et E-LOTOS.

### 7.3 Réseaux et groupes de travail internationaux

- Nous sommes membres du groupe de travail ERCIM sur les méthodes formelles pour les systèmes industriels critiques. Dans ce cadre, R. Mateescu a obtenu une bourse post-doctorale ERCIM pour un projet de coopération entre l'INRIA et le CWI (Amsterdam) débutant au 1er décembre 1997.
- Nous participons à l'action de normalisation intitulée "*Enhancements to* LOTOS" dans le cadre du groupe de travail ISO/IEC SC21/WG7 consacré aux systèmes distribués ouverts (ODP). Pour ces travaux, H. Garavel est délégué AFNOR (*Association Française de Normalisation*) et M. Sighireanu est déléguée RSI (*Romanian Standard Institute*).

### 7.4 Relations bilatérales internationales

**Europe :** Nous avons eu des relations scientifiques avec l'Université Libre de Bruxelles (Pr. Thierry Massart), l'Université Polytechnique de Bucarest (Pr. Marian Dobre, qui nous a rendu visite le 4 juillet 1997), l'Université de Liège (Pr. Guy Leduc), l'Université de Madrid (Pr. Juan Quemada) et l'équipe SEN2 du CWI (Pr. Jan-Friso Groote)

**Amérique du Sud :** Nous avons collaboré avec l'Université Fédérale de Sainte-Catherine à Florianopolis au Brésil (Mirela Sechi Moretti Annoni Notare, qui nous a rendu visite le 7 juillet 1997).

### 7.5 Accueil de chercheurs étrangers

Ch. Pecheur, ingénieur et docteur de l'Université de Liège, est accueilli dans l'équipe depuis le 15 janvier 1997 pour une durée d'un an, sur une bourse post-doctorale de l'INRIA Rhône-Alpes.

## 8 Diffusion de résultats

### 8.1 Animation de la communauté scientifique

- Nous avons organisé la réunion de normalisation internationale ISO SC21/WG7 consacrée à la définition d'EXTENDED-LOTOS, qui a eu lieu du 9 au 12 décembre 1996 à l'INRIA Rhône-Alpes.
- Nous avons participé à la création du "Club des Méthodes Formelles" de l'INRIA Rhône-Alpes à un double titre : d'une part dans le cadre d'un séminaire donné par Ch. Pecheur et M. Sighireanu le 26 juin 1997, et d'autre part en organisant le 24 juillet 1997 une journée consacrée aux outils CADP à l'intention du monde académique et industriel.
- H. Garavel est membre du comité de rédaction de la revue TSI (*Technique et Science Informatiques*). En collaboration avec Roland Groz (France Telecom/CNET) et Claude Jard (IRISA), il a été responsable de deux numéros thématiques de cette revue consacrés au thème "Méthodes formelles : Validation de systèmes complexes" [8, 9].

### 8.2 Enseignement universitaire

- H. Garavel assure le cours "Temps Réel" destiné aux étudiants en 3e année de l'Ensimag (24 heures annuelles).
- H. Garavel a organisé plusieurs séances de travaux pratiques consacrées à la spécification et la validation de protocoles à l'Enst Paris (8 heures annuelles) et à l'Enserg (16 heures annuelles).
- M. Sighireanu est monitrice à l'Université Joseph Fourier (64 heures annuelles). Elle a participé à l'enseignement de langages de programmation à l'Ecole d'Informatique (niveau licence).

### 8.3 Participation à des colloques, séminaires, invitations

- Nous avons présenté des communications dans plusieurs conférences et colloques internationaux (voir à ce sujet la liste de nos publications).
- Ch. Pecheur a participé à l'Ecole Européenne "*Methods and Tools for the Verification of Infinite State Systems*" organisée à Grenoble du 23 au 25 mars 1997.
- H. Garavel et R. Mateescu ont participé au second colloque international Ercim "*Formal Methods in Industrial Critical Systems*" qui s'est tenu à Cesena (Italie) les 4 et 5 juillet 1997.
- H. Garavel a participé à la réunion de normalisation consacrée à E-Lotos, organisée à Helsinki (Finlande) du 14 au 18 juillet 1997.
- Ch. Pecheur a participé au Colloque Francophone sur l'Ingénierie des Protocoles Cfip'97 qui s'est tenu à Liège (Belgique) du 29 septembre au 2 octobre 1997.
- H. Garavel a participé au 11e colloque Cost 247 qui s'est tenu à Stirling (Ecosse) les 13 et 14 octobre 1997. A cette occasion, il a présenté les résultats obtenus par Ch. Pecheur dans l'étude du protocole Co4.

## 9 Bibliographie

### Ouvrages et articles de référence de l'équipe

[1] E. M. Clarke, E. A. Emerson, A. P. Sistla, « Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications », *ACM Transactions on Programming Languages and Systems 8*, 2, avril 1986, p. 244–263.

[2] H. Ehrig, B. Mahr, *Fundamentals of Algebraic Specification 1 — Equations and Initial Semantics*, *EATCS Monographs on Theoretical Computer Science*, *6*, Springer Verlag, 1985.

[3] M. Hennessy, R. Milner, « Algebraic Laws for Nondeterminism and Concurrency », *Journal of the ACM 32*, 1985, p. 137–161.

[4] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.

[5] ISO/IEC, « LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour », International Standard nNo 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, septembre 1988.

[6] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[7] R. D. Nicola, F. W. Vaandrager, *Action versus State based Logics for Transition Systems*, *Lecture Notes in Computer Science*, *469*, Springer Verlag, 1990, p. 407–419.

## Livres et monographies

[8] H. Garavel, R. Groz, C. Jard (réd.), *Méthodes formelles : validation de systèmes critiques (1)*, *16-6*, Technique et Science Informatiques. Hermès, Paris, juin 1997.

[9] H. Garavel, R. Groz, C. Jard (réd.), *Méthodes formelles : validation de systèmes critiques (2)*, *16-9*, Technique et Science Informatiques. Hermès, Paris, novembre 1997.

## Articles

[10] H. Garavel, L. Mounier, « Specification and Verification of Various Distributed Leader Election Algorithms for Unidirectional Ring Networks », *Science of Computer Programming 29*, 1–2, juillet 1997, p. 171–197, Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Full version available as INRIA Research Report RR-2986.

[11] R. Mateescu, « Vérification de systèmes répartis : l'exemple du protocole Brp », *Technique et Science Informatiques 16*, 6, juin 1997, p. 725–751.

## Communications à des manifestations scientifiques

[12] H. Garavel, M. Jorgensen, R. Mateescu, C. Pecheur, M. Sighireanu, B. Vivien, « CADP'97 – Status, Applications and Perspectives », *in : Proceedings of the 2nd COST 247 International Workshop on Applied Formal Methods in System Design (Zagreb, Croatia)*, I. Lovrek (réd.), juin 1997.

[13] C. Pecheur, « Specification and Verification of the CO4 Distributed Knowledge System Using LOTOS », *in : Proceedings of the 12th IEEE International Conference on Automated Software Engineering ASE-97 (Incline Village, Nevada, USA)*, M. Lowry, Y. Ledru (réd.), novembre 1997. Extended version available as INRIA Research Report RR-3259.

[14] M. Sighireanu, R. Mateescu, « Validation of the Link Layer Protocol of the IEEE-1394 Serial Bus ("FireWire") : an Experiment with E-LOTOS », *in : Proceedings of the 2nd COST 247 International Workshop on Applied Formal Methods in System Design (Zagreb, Croatia)*, I. Lovrek (réd.), juin 1997. Full version available as INRIA Research Report RR-3172.

## Rapports de recherche

[15] H. Garavel, M. Sighireanu, « A Proposal for Coroutines and Suspend-Resume in E-LOTOS », Vasy report, INRIA, décembre 1996, Input document [GR3] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Grenoble, France, December, 9–11, 1996.

[16] M. Sighireanu, H. Garavel, « On the Definition of Modular E-LOTOS », Vasy report, INRIA, décembre 1996, Input document [GR2] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Grenoble, France, December, 9–11, 1996.

[17] M. Sighireanu, H. Garavel, « A Proposal for Coroutines in E-LOTOS », Vasy report, INRIA, juillet 1997, Input document [HEL2] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Helsinki, Finland, July, 14–18, 1997.

[18] B. Vivien, « Etude et réalisation d'un compilateur E-LOTOS à l'aide du générateur de compilateurs SYNTAX/FNC-2 », mémoire d'ingénieur, CNAM, Grenoble, décembre 1997.