INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Action VASY-RA

Validation de Systèmes – Recherche et Applications

Rhône-Alpes

THÈME 1C

Rapport d'Activité

1998

Table des matières

1 Composition de l'équipe **3**

2 Présentation et objectifs généraux **4**
2.1 Introduction 4
2.2 Technologie des modèles – vérification 4
2.3 Technologie des langages – compilation 5
2.4 Implémentation et expérimentation 6

3 Domaines d'applications **7**

4 Logiciels **7**
4.1 La boîte à outils CADP 7
4.2 Le compilateur TRAIAN 9

5 Résultats nouveaux **10**
5.1 Technologie des modèles – vérification 10
5.1.1 Amélioration de l'environnement OPEN/CAESAR 10
5.1.2 Développement du simulateur OCIS 11
5.1.3 Amélioration de l'environnement BCG 14
5.1.4 Développement de l'outil SVL 15
5.1.5 Amélioration de l'évaluateur générique XTL 16
5.2 Technologie des langages – compilation 17
5.2.1 Compilation du langage LOTOS 17
5.2.2 Contribution à la définition d'E-LOTOS 18
5.2.3 Réalisation du compilateur TRAIAN 19
5.2.4 Amélioration du générateur de compilateurs FNC-2 20
5.3 Etudes de cas et applications pratiques 20
5.3.1 Protocole de cohérence de caches CC-NUMA "Polykid" 21
5.3.2 Protocole de cohérence de caches CC-NUMA "Fame" 22
5.3.3 Autres études de cas 23

6 Contrats industriels (nationaux, européens et internationaux) **24**
6.1 Action Vasy (Dyade) 24

7 Actions régionales, nationales et internationales **25**
7.1 Actions nationales 25
7.1.1 Groupes de travail nationaux 25
7.1.2 Relations bilatérales nationales 25
7.2 Actions internationales 26
7.2.1 Groupes de travail internationaux 26
7.2.2 Relations bilatérales internationales 26
7.3 Accueil de chercheurs étrangers 26

8 Diffusion de résultats **27**
8.1 Diffusion de logiciels 27
8.1.1 Diffusion de la boîte à outils CADP 27
8.1.2 Diffusion du compilateur TRAIAN 28
8.2 Animation de la communauté scientifique 28
8.3 Enseignement universitaire 29
8.4 Participation à des colloques, séminaires, invitations 29

9 Bibliographie **29**

1 Composition de l'équipe

Responsable scientifique

Hubert Garavel [CR1 Inria]

Assistantes de projet

Béatrice Claudio

Joanna Payart

Personnel Inria

Radu Mateescu [CR2, depuis octobre 1998]

Ingénieurs experts

Mark Jorgensen [ingénieur Dyade, jusqu'à janvier 1998 [1]]

Christophe Discours [ingénieur Dyade, depuis avril 1998]

Chercheurs extérieurs

Ghassan Chehaibar [ingénieur Bull, jusqu'à février 1998 [2]]

Massimo Zendri [ingénieur Bull]

Chercheur post-doctorant

Charles Pecheur [bourse Inria Rhône-Alpes, jusqu'en mai 1998 [3]]

Chercheur doctorant

Mihaela Sighireanu [bourse MENRT, jusqu'en novembre 1998 [4]]

Stagiaires

Manuel Aguilar Cornejo [DEA, depuis octobre 1998]

Xavier Bouchoux [élève-ingénieur ENSIMAG, juillet-septembre 1998]

Bruno Hondelatte [élève-ingénieur ENSIMAG, janvier-juin 1998]

Pierre Kessler [élève-ingénieur ENSIMAG, janvier-juin 1998]

Aldo Mazzilli [élève-ingénieur CNAM, depuis octobre 1998]

1. Actuellement ingénieur responsable de l'intranet applicatif à Usinor/Sollac (Dunkerque).
2. Actuellement ingénieur à Bull, Les Clayes-sous-Bois.
3. Actuellement chercheur au *Research Institute for Advanced Computer Science* (RIACS) pour le compte de la NASA (*Ames Research Center*, Californie, USA).
4. Actuellement chercheur post-doctoral INRIA dans le cadre de l'action de recherche coopérative TOLERE regroupant les projets BIP et SOSSO.

2 Présentation et objectifs généraux

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 formelle, complétées par des outils informatiques adaptés, offrant des fonctionnalités de simulation, prototypage rapide, vérification et génération de tests.

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, etc.). Ces techniques, bien que moins générales que les approches par preuves (*theorem proving*), possèdent pourtant l'avantage de permettre une détection automatique, 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). Etant donné 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 équipes 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 *μ-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 μ-calcul modal 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 μ-calcul standard comme, par exemple, le fait qu'une variable donnée soit toujours croissante sur un chemin d'exécution. Nous travaillons aussi à la conception et à l'implémentation d'algorithmes d'évaluation efficaces pour cette extension du μ-calcul.

Bien que ces techniques soient efficaces et complètement automatisables, leur principale limitation réside dans 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 :

- Les techniques de *model-checking* nécessitent de pouvoir exécuter efficacement les programmes à vérifier ;
- La modélisation de systèmes critiques ne saurait reposer sur des langages dont la sémantique ne serait pas rigoureusement définie, car l'absence de sémantique formelle conduit bien souvent à des ambiguités et des divergences d'interprétation (notamment entre concepteurs et implémenteurs).
- De plus, 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 [ISO88] et possédant les propriétés ci-dessus. Il s'agit d'un langage basé sur les concepts des algèbres de processus (notamment CCS [Mil89] et CSP [Hoa85]) pour la description du contrôle et les types abstraits algébriques [EM85] 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 GIE BULL-INRIA DYADE). Cette confrontation systématique avec les problèmes d'implémentation et d'expérimentation est un aspect essentiel de notre approche.

[ISO88] ISO/IEC, «LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour», *International Standard n° 8807*, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, septembre 1988.

[Mil89] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1989.

[Hoa85] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.

[EM85] H. Ehrig, B. Mahr, *Fundamentals of Algebraic Specification 1 — Equations and Initial Semantics*, *EATCS Monographs on Theoretical Computer Science*, *6*, Springer Verlag, 1985.

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 incarnations 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, gestion de stocks ;
- **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 ;
- **interactions homme-machine :** interfaces graphiques, visualisation de données biomédicales, etc.

4 Logiciels

4.1 La boîte à outils CADP

Participants : Hubert Garavel [correspondant], Radu Mateescu.

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

En collaboration avec le laboratoire VERIMAG, nous développons la boîte à outils CADP (CÆSAR/ALDÉBARAN *Development Package*) pour l'ingénierie des protocoles et des systèmes distribués [2, 7, 3] (voir `http://www.inrialpes.fr/vasy/cadp`).

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 du réseau de Petri.

- 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, etc.) 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 (représentés 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 représentés 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 [CES86], HML [HM85], ACTL [NV90], 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.

[CES86] 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.

[HM85] M. Hennessy, R. Milner, « Algebraic Laws for Nondeterminism and Concurrency », *Journal of the ACM 32*, 1985, p. 137–161.

[NV90] 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.

A ces outils s'ajoutent ceux développés par le laboratoire VERIMAG, qui permettent la comparaison et la réduction de graphes modulo des relations d'équivalence et de préordre 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.

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.

4.2 Le compilateur TRAIAN

Participants : Mihaela Sighireanu [correspondant], Xavier Bouchoux.

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

Les algèbres de processus sont des formalismes particulièrement adaptés à la spécification des protocoles de télécommunication et des systèmes répartis. Cependant, en dépit d'une base mathématique rigoureuse, des efforts de normalisation (notamment ceux concernant le langage LOTOS) et d'un nombre croissant d'études de cas traitées avec succès, les algèbres de processus ne sont pas encore pleinement acceptées en milieu industriel. Elle se voient souvent supplantées par des langages dont l'apparence plus facile (syntaxe graphique ou proche des langages algorithmiques classiques) masque souvent l'absence de sémantique formelle, lacune assez inquiétante lorsqu'il s'agit de modéliser et de valider des systèmes critiques.

Le langage LOTOS NT (LOTOS *Nouvelle Technologie*) vise à regrouper les "meilleurs aspects des deux mondes", en définissant une technique de description formelle de "seconde génération" — par comparaison avec la première génération de langages normalisés : ESTELLE, LOTOS et SDL — qui combine les fondements théoriques rigoureux des algèbres de processus avec des facilités de description permettant d'assurer une meilleure diffusion industrielle du langage.

LOTOS NT est composé d'une *partie données*, qui permet une description naturelle des types de données et des fonctions tout en étant facilement analysable et implémentable, d'une *partie contrôle*, qui étend l'algèbre de processus de LOTOS par des constructions plus expressives et la prise en compte du temps quantitatif, et d'une *partie modules*, qui autorise la structuration

et la réutilisation des descriptions LOTOS NT.

Nous avons commencé la conception de LOTOS NT en 1992, dans le cadre du groupe de travail ISO/IEC pour la révision de la norme LOTOS. Plusieurs de nos propositions d'amélioration de LOTOS sont d'ores et déjà intégrées dans la future norme, appelée E-LOTOS.

Nous développons également un compilateur, appelé TRAIAN, pour LOTOS NT, afin de pouvoir traduire automatiquement une description LOTOS NT vers un programme C pouvant ensuite être utilisé à des fins de simulation, de prototypage rapide, de vérification et de test.

La version courante de TRAIAN, disponible à l'adresse `http://www.inrialpes.fr/vasy/traian`, effectue l'analyse syntaxique et sémantique des descriptions LOTOS NT et traduit en C la partie données du langage.

Bien qu'il reprenne certains principes des compilateurs CÆSAR.ADT et CÆSAR dédiés à LOTOS, le compilateur TRAIAN préfigure une "nouvelle génération" d'outils pour l'ingénierie des protocoles, tant sur le plan algorithmique — nouvelles formes intermédiaires, nouveaux algorithmes de traduction — que sur le plan architectural — il est construit au moyen du générateur de compilateurs SYNTAX/FNC-2 (voir § 5.2.4).

5 Résultats nouveaux

5.1 Technologie des modèles – vérification

Mots clés : automate, bisimulation, compilation, concurrence, génération de code, génie logiciel, logique temporelle, méthodes formelles, modélisation, mu-calcul, parallélisme asynchrone, protocole de communication, spécification formelle, synchronisation, système distribué, vérification de programme.

Résumé : *En 1998, nos travaux sur la vérification ont porté sur l'extension et l'amélioration d'outils existants (OPEN/CÆSAR, BCG et XTL), ainsi que sur le développement de nouveaux outils (OCIS et SVL) visant à faciliter l'utilisation des techniques formelles en milieu industriel.*

5.1.1 Amélioration de l'environnement OPEN/CAESAR

Participants : Christophe Discours, Hubert Garavel.

L'environnement logiciel OPEN/CÆSAR (voir § 4.1) est l'un des constituants essentiels de la boîte à outils CADP. C'est lui qui assure la connexion entre les outils dédiés aux langages et les outils opérant sur les modèles.

Au-delà de CADP, il convient de noter que d'autres équipes de recherche ont aussi choisi d'interconnecter leurs propres outils à l'environnement OPEN/CÆSAR : on peut notamment citer le compilateur SIGNAL (développé par le projet EP-ATR) et l'outil KRONOS pour la vérification des systèmes temporisés (développé par le laboratoire VERIMAG).

En 1998, un article de synthèse décrivant les principes et l'architecture de l'environnement OPEN/CÆSAR a été publié [4].

Par ailleurs, cet environnement a été enrichi par de nouvelles bibliothèques logicielles :

- La bibliothèque CAESAR_HIDE permet de caractériser un ensemble d'étiquettes de transitions au moyen d'une liste d'expressions régulières contenue dans un fichier, ce qui trouve une application immédiate dans la mise en œuvre d'abstractions par masquage de transitions.

- La bibliothèque CAESAR_RENAME permet le renommage d'étiquettes de transitions au moyen d'une liste d'expressions régulières contenue dans un fichier, ces expressions étant étendues par des substitutions de manière semblable aux expressions régulières disponibles sous UNIX.

Ces bibliothèques ont permis d'enrichir les fonctionnalités de l'outil de génération de tests TGV (développé par le projet PAMPA et le laboratoire VERIMAG) et des outils de vérification ALDÉBARAN et PROJECTOR (développés par le laboratoire VERIMAG).

Enfin, nous avons entrepris en 1998 une réflexion visant à étendre l'interface de programmation OPEN/CÆSAR en y introduisant de nouvelles fonctionnalités : tâches, variables d'états, états locaux, orientation des communications, temps quantifié et informations de remontée dans le code source aux fins de débogage. Ce travail a débouché sur une nouvelle interface de programmation (appelée "OPEN/CÆSAR version 2") assurant une totale compatibilité ascendante avec l'interface actuelle (appelée "OPEN/CÆSAR version 1") qui en est un sous-ensemble.

5.1.2 Développement du simulateur OCIS

Participants : Hubert Garavel, Bruno Hondelatte, Pierre Kessler.

A la demande de nos partenaires de BULL, nous avons entrepris le développement d'un outil pour faciliter la mise au point des descriptions formelles de protocoles. Cet outil, baptisé OCIS (OPEN/CÆSAR *Interactive Simulator*), est un simulateur fonctionnant, soit en mode texte, soit en mode graphique (en utilisant les outils TCL/TK et TIX pour la construction d'interfaces graphiques).

Il est destiné à remplacer les outils existants SIMULATOR et XSIMULATOR, par rapport auxquels il introduit des améliorations importantes :

- Alors que les outils existants sont basés sur la version 1 de l'interface OPEN/CÆSAR, OCIS s'appuie sur la version 2 de l'interface OPEN/CÆSAR (voir § 5.1.1) qui fournit davantage d'information sur le protocole analysé. De ce fait, OCIS offre des possibilités accrues en matière d'inspection des états du système et de remontée dans le code source.

- OCIS permet de visualiser, de différentes manières, les communications (envois de messages, rendez-vous, diffusion, etc.) entre processus parallèles, notamment sous forme de diagrammes MSC (*Message Sequence Charts*, voir figure 1).

- OCIS permet de manipuler et de visualiser des scénarios de simulation arborescents, qui peuvent être stockés dans des fichiers (au format BCG) et ré-exécutés ultérieurement (voir figure 2).

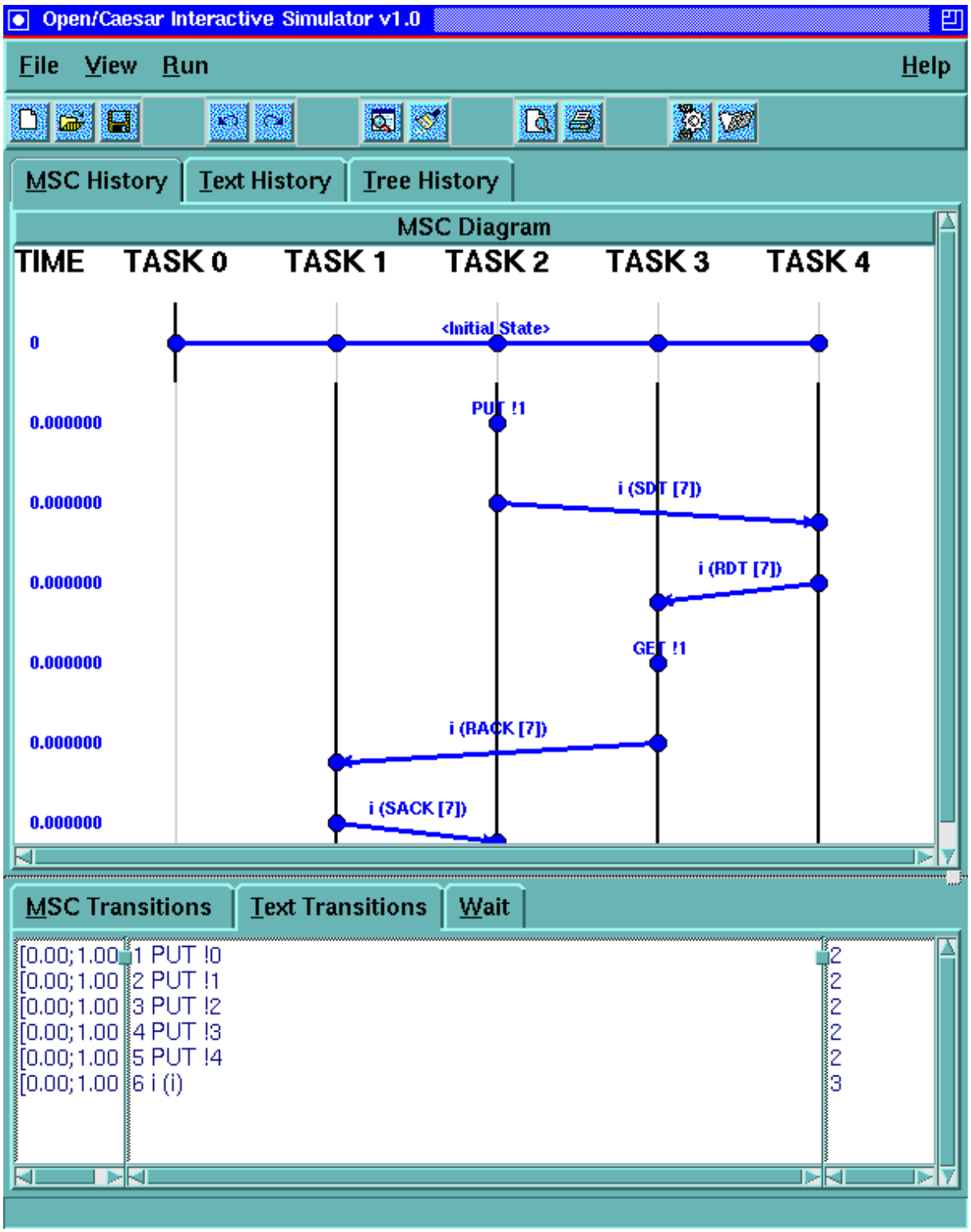

FIG. 1 – *Visualisation des communications avec* OCIS

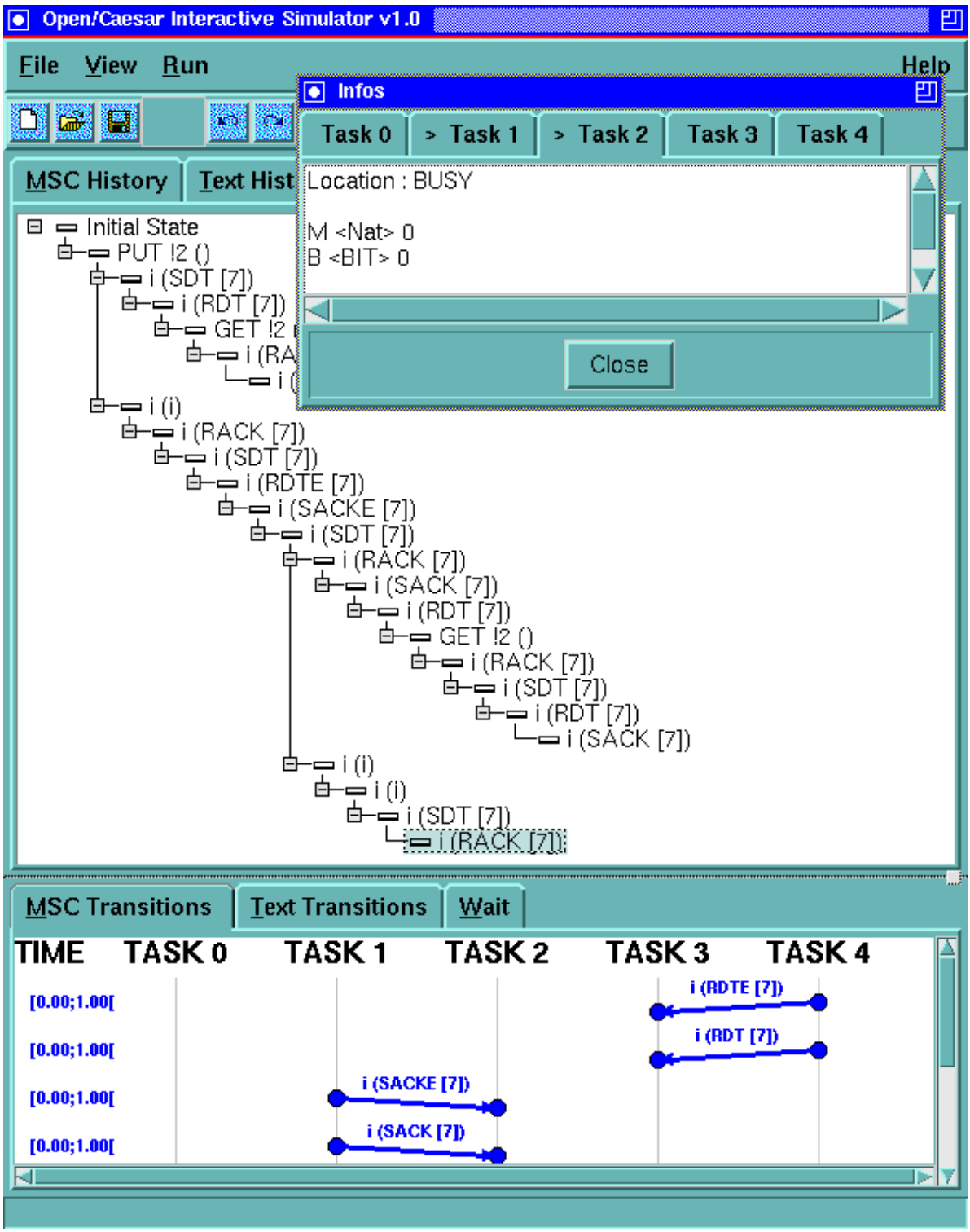

FIG. 2 – *Visualisation des scénarios avec* OCIS

- OCIS est adapté à la simulation des systèmes temporisés : à cet effet, il intègre les concepts de temps global quantitatif, d'attente sur un événement, d'action urgente, etc.
- Outre la simulation interactive, OCIS permet aussi la simulation guidée par un objectif : il est ainsi possible de rechercher automatiquement un chemin d'exécution (ou tous les chemins) défini(s) par une expression régulière. Ceci a été obtenu en combinant OCIS avec l'outil EXHIBITOR déjà disponible dans l'environnement OPEN/CÆSAR.
- Enfin, OCIS autorise l'utilisateur à recompiler le programme source sans avoir à quitter le simulateur.

A terme, l'objectif de ce travail est de disposer, pour le développement de protocoles et de systèmes distribués, d'un simulateur offrant des fonctionnalités comparables à celles présentes dans les débogueurs et environnements de développement existant pour les langages séquentiels. Un avantage essentiel d'OCIS est qu'il n'est pas lié à un langage source précis : à terme, nous envisageons de l'utiliser pour LOTOS, pour LOTOS NT, ainsi que pour tout langage dont le compilateur implémentera l'interface OPEN/CÆSAR version 2.

5.1.3 Amélioration de l'environnement BCG

Participants : Christophe Discours, Hubert Garavel, Radu Mateescu.

Le format BCG et l'environnement logiciel qui lui est associé (interfaces de programmation, bibliothèques logicielles et outils) jouent un rôle central dans la boîte à outils CADP.

En 1998, nous avons intensifié nos efforts visant à promouvoir l'utilisation du format BCG dans des contextes applicatifs multiples :

- Nous avons développé une interface de programmation (appelée BCG_READ) qui offre des primitives de haut niveau pour lire et manipuler aisément des fichiers au format BCG.
- En collaboration avec Laurent Mounier (laboratoire VERIMAG), nous avons systématisé l'emploi du format BCG pour tous les outils de CADP, notamment ALDÉBARAN, EXP.OPEN et PROJECTOR.
- Nous avons développé un outil nommé BCG_LABELS qui permet d'appliquer des abstractions et des renommages sur les étiquettes attachées aux transitions d'un graphe BCG ; cet outil tire parti des bibliothèques CAESAR_HIDE et CAESAR_RENAME récemment ajoutées à l'environnement OPEN/CÆSAR (voir § 5.1.1).
- Dans le cadre de notre collaboration avec BULL, nous avons développé deux outils nommés BCG_LOOP et BCG_SPLIT qui remédient à certaines limitations de l'outil de génération de tests TGV :
  - BCG_LOOP permet d'obtenir des tests cycliques, ce qui n'est actuellement pas possible avec l'outil TGV. Il prend en entrée un graphe BCG sans circuit produit par TGV et construit en sortie un graphe BCG obtenu en remplaçant certaines séquences d'actions conduisant à des verdicts de tests "non concluants" par des transitions

de "rebouclage". Cette introduction de circuits dans les tests est nécessaire pour modéliser correctement les cas où un même message doit être retransmis un nombre quelconque de fois.

- BCG_SPLIT permet de transformer les tests non déterministes produits par TGV en tests déterministes exécutables. Il prend en entrée un graphe BCG engendré par TGV et un ensemble d'étiquettes de transitions (définies par une liste d'expressions régulières) correspondant aux stimuli à fournir en entrée du système à tester. BCG_SPLIT fournit en sortie un ensemble de graphes BCG qui définissent des tests exécutables que l'on peut appliquer au système à tester.

- Nous avons développé un outil nommé BCG_REMOTE qui permet, à partir d'une machine donnée, de produire des graphes BCG stockés sur une autre machine, en utilisant un protocole de communication basé sur l'appel de procédures distantes.

Lors de son séjour post-doctoral au CWI (Amsterdam), R. Mateescu a interconnecté le compilateur μCRL du CWI avec l'environnement BCG, ce qui a permis de résoudre les problèmes d'espace disque relatifs à la taille des graphes engendrés et rendu possible l'utilisation de CADP pour la vérification de programmes μCRL.

Par ailleurs, d'autres chercheurs du CWI ont adopté les environnements BCG et OPEN/CÆSAR pour développer un outil de génération de tests de protocoles exploitant les propriétés des symétries [RS98].

5.1.4 Développement de l'outil SVL

Participants : Christophe Discours, Hubert Garavel, Mark Jorgensen.

Lorsque l'on essaie de vérifier des systèmes assez complexes, le problème de l'explosion d'états (voir § 2.2) survient fréquemment. Les techniques de vérification compositionnelle avec abstractions [GLS96,KM97] intégrées à la boîte à outils CADP visent à éviter l'explosion d'états et fournissent souvent une réponse appropriée à ce problème, y compris sur des études de cas industrielles significatives [1].

Toutefois, notre collaboration avec BULL a montré que ces techniques de vérification compositionnelles restaient délicates à mettre en œuvre et supposaient une certaine expertise, tant en ce qui concerne la façon d'appliquer les différents outils (ALDÉBARAN, CÆSAR, BCG,

[RS98] J. Romijn, J. Springintveld, «Exploiting Symmetry in Protocol Testing», *in : Proceedings of the IFIP Joint International Conference on Formal Description Techniques and Protocol Specification, Testing, and Verification FORTE/PSTV'98 (Paris, France)*, S. Budkowski, A. Cavalli, E. Najm (éditeurs), IFIP, Kluwer Academic Publishers, p. 337–352, novembre 1998. Full version available as Technical Report CSI-R9821, Computing Science Institute, University of Nijmegen, September 1998.

[GLS96] S. Graf, G. Lüttgen, B. Steffen, «Compositional Minimisation of Finite State Systems using Interface Specifications», *Formal Aspects of Computation 8*, septembre 1996.

[KM97] J.-P. Krimm, L. Mounier, «Compositional State Space Generation from LOTOS Programs», *in : Proceedings of TACAS'97 Tools and Algorithms for the Construction and Analysis of Systems (University of Twente, Enschede, The Netherlands)*, E. Brinksma (éditeur), *Lecture Notes in Computer Science, 1217*, Springer Verlag, Berlin, avril 1997. Extended version with proofs available as Research Report VERIMAG RR97-01.

EXP.OPEN, OPEN/CÆSAR, PROJECTOR, etc.) que pour l'écriture des nombreux fichiers intermédiaires (abstractions, renommages, synchronisations, *Makefiles*, etc.) nécessaires à la vérification compositionnelle.

C'est pourquoi nous avons conçu un langage appelé SVL (*System Validation Language*) destiné à simplifier et à automatiser la mise en œuvre de la vérification compositionnelle, afin que celle-ci devienne réellement utilisable dans un contexte industriel. Dans son principe, SVL se présente comme un langage de haut niveau pour l'écriture de scénarios de vérification :

- Il permet de décrire l'architecture du système à vérifier sous forme d'un système de processus communicants connectés par des opérateurs algébriques de composition parallèle ; le langage SVL unifie et remplace les deux formats de description d'architectures (".DES" et ".EXP") qui existaient précédemment dans CADP.

- SVL offre aussi des opérateurs additionnels permettant de spécifier facilement les différentes étapes (réductions, comparaisons, abstractions, etc.) de la vérification compositionnelle, ces opérations pouvant être effectuées soit par les outils de CADP, soit par les outils FC2TOOLS développés dans le projet MEIJE (INRIA Sophia-Antipolis), ceci de manière entièrement transparente pour l'utilisateur, qui n'a plus à se soucier de la syntaxe d'appel exacte des différents outils.

Nous avons également développé un compilateur pour le langage SVL. Réalisé à l'aide des outils SYNTAX/FNC-2 (voir § 5.2.4), ce compilateur produit, à partir d'un programme SVL, un *shell-script* UNIX contenant la liste des commandes à exécuter ainsi que les différents fichiers auxiliaires nécessaires à la vérification compositionnelle.

5.1.5 Amélioration de l'évaluateur générique XTL

Participants : Hubert Garavel, Radu Mateescu, Charles Pecheur.

XTL (*eXecutable Temporal Language*, voir § 4.1) est à la fois un méta-langage et un outil permettant la description et la vérification des propriétés temporelles des systèmes. L'évaluateur XTL version 1.1 est d'ores et déjà intégré à la boîte à outils CADP et a été utilisé avec succès pour la validation de trois applications industrielles : le protocole BRP de Philips [Mat97], le bus série IEEE-1394 "Firewire" [2] et le système de mémoire virtuelle distribuée CFS [11].

En 1998, nous avons continué les travaux autour de la version 1.1 d'XTL :

- Suite au retour d'expérience des utilisateurs, plusieurs erreurs et incohérences ont été corrigées dans l'outil et sa documentation. Les réponses aux questions des utilisateurs d'XTL ont été documentées dans le forum aux questions (FAQ) relatif à CADP.

- Un article [7] décrivant le langage et l'outil XTL version 1.1 a été publié. Les principes d'XTL ont été présentés lors d'un séminaire à l'Université de Twente et d'une réunion de l'action de recherche coopérative VERDON.

[Mat97] R. Mateescu, « Vérification de systèmes répartis : l'exemple du protocole BRP », *Technique et Science Informatiques 16*, 6, juin 1997, p. 725–751.

– Ch. Pecheur a défini en XTL une nouvelle version de la bibliothèque ACTL permettant de générer des séquences de diagnostic expliquant la valeur de vérité des formules [11].

Une nouvelle version 2.0 du langage XTL a été définie dans la thèse de R. Mateescu [1]. Cette seconde version complète le langage avec des opérateurs modaux et de point fixe du μ-calcul modal, étendus avec des variables typées et des expressions régulières.

Nous avons proposé des algorithmes efficaces pour l'évaluation des formules du langage XTL version 2.0 sur des modèles finis. A la différence des algorithmes d'évaluation utilisés pour XTL version 1.1, les nouveaux algorithmes permettent aussi d'évaluer les formules à la volée (sans construire préalablement le modèle). L'algorithme d'évaluation à la volée dédié au fragment du μ-calcul étendu sans alternance (c'est-à-dire les formules qui ne contiennent pas d'opérateurs de plus petit et de plus grand point fixe mutuellement récursifs) a fait l'objet d'une publication [8].

5.2 Technologie des langages – compilation

Mots clés : algèbre de processus, automate, compilation, concurrence, génération de code, génie logiciel, méthodes formelles, modélisation, parallélisme asynchrone, spécification formelle, synchronisation, système distribué, temps réel.

Résumé : *En 1998, nous avons travaillé sur l'amélioration des techniques de compilation pour le langage LOTOS. Nous avons activement participé au groupe de travail sur la normalisation du langage E-LOTOS (Extended-LOTOS) et poursuivi la réalisation du compilateur TRAIAN pour ce langage. Nous avons également contribué à l'amélioration du générateur de compilateurs Fnc-2 développé à l'INRIA Rocquencourt.*

5.2.1 Compilation du langage LOTOS

Participants : Hubert Garavel, Charles Pecheur.

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.

En 1998, nous pouvons mentionner plusieurs améliorations concernant notre compilateur CÆSAR dédié au langage LOTOS :

– Les activités de validation et de génération de tests pour les architectures multiprocesseur de BULL (voir § 5.3.1) ont permis de détecter et corriger deux bogues — jusqu'alors restées inaperçues — qui se manifestaient lorsque CÆSAR était utilisé sur des descriptions LOTOS de grande taille.

– Un nouvel algorithme d'analyse du flot de contrôle a été implémenté dans CÆSAR.

– Les réseaux de Petri étendus utilisés comme formalisme intermédiaire par CÆSAR [5, 4] comportent une classe de transitions particulières, dites *epsilon-transitions*. Le franchissement de ces transitions obéit à des règles complexes qui constituent le cœur de la phase d'exploration d'états. L'algorithme qui implémente les règles de franchissement des epsilon-transitions joue donc un rôle critique pour la rapidité de CÆSAR.

  En 1998, nous avons réactivé des travaux antérieurs [Gar94] visant à remplacer les règles existantes pour le franchissement des epsilon-transitions par de nouvelles règles plus générales, susceptibles de conduire à un algorithme plus efficace. Une analyse rigoureuse a permis de justifier la correction de ces nouvelles règles ainsi que celle de l'algorithme correspondant.

  De plus, sur base des nouvelles règles, trois optimisations complémentaires ont été définies et justifiées. Ces optimisations, qui font notamment appel à la théorie des ordres partiels, devraient à terme améliorer les performances du compilateur CÆSAR.

  Toutefois, ni la nouvelle sémantique des epsilon-transitions, ni les optimisations associées n'ont pu encore être incorporées dans CÆSAR du fait que, en 1998, les besoins industriels de BULL nous ont conduits à privilégier les travaux relatifs à la génération de tests par rapport aux activités de validation de protocoles.

5.2.2 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).

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 1998, nos principales contributions ont été les suivantes :

– Nous avons participé à l'élaboration du document *Final Committee Draft* [Que98], qui intègre les décisions prises lors de la réunion internationale d'Helsinki en juillet 1997. Nous avons analysé les versions préliminaires de ce document et formulé des remarques détaillées dont beaucoup ont été prises en compte par l'éditeur du document.

– Nous avons participé au vote international par lequel ce document a été approuvé en octobre 1998.

[Gar94] H. Garavel, «Sur la sémantique des epsilon-transitions», Publication interne, INRIA projet SPECTRE, mai 1994.

[Que98] J. Quemada, editor, «Committee Draft on Enhancements to LOTOS (E-LOTOS)», ISO/IEC FCD 15437, avril 1998.

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 (style de programmation impératif, surcharge d'opérateurs, tableaux, typage statique) qui en font un langage plus facile à utiliser et plus simple à implémenter.

En 1998, nos principales avancées sont les suivantes :

- La syntaxe, la sémantique statique et la sémantique dynamique de LOTOS NT ont été complètement formalisées.
- Les principes de LOTOS NT ont été présentés dans un article invité [3], ainsi qu'en diverses occasions, notamment lors de deux séminaires de l'action de recherche coopérative VERDON.
- Le langage LOTOS NT a été utilisé dans deux études de cas — qui ont donné lieu à plusieurs publications relatives à la modélisation de la couche liaison du bus IEEE 1394 "FIREWIRE" [2] et d'un système de gestion de stocks [9, 12].

5.2.3 Réalisation du compilateur TRAIAN

Participants : Xavier Bouchoux, Mihaela Sighireanu.

Afin de valider nos propositions concernant LOTOS NT, nous avons entrepris en 1997 l'implémentation d'un compilateur (appelé TRAIAN) pour ce langage. Ce travail a conduit à une souche de compilateur effectuant l'analyse lexicale et syntaxique pour LOTOS NT, la construction d'arbre abstrait et les vérifications de sémantique statique.

En 1998, nous avons poursuivi les développements liés à TRAIAN :

- Nous avons amélioré la partie amont (*front end*) du compilateur en corrigeant diverses bogues et en implémentant les traitements sémantiques correspondant à la partie modules de LOTOS NT.
- Nous avons entrepris l'implémentation de la partie aval (*back end*) du compilateur en implémentant un générateur de code qui traduit en langage C les définitions de types et de fonctions contenues dans une description LOTOS NT.
- Enfin, nous avons finalisé une technique d'implémentation de la partie contrôle de LOTOS NT, en proposant un modèle d'exécution basé sur les réseaux de Petri temporisés et les algorithmes correspondants [Sig99].

[Sig99] M. Sighireanu, *Contribution à la définition et à l'implémentation de la norme "Extended LOTOS"*, Thèse de doctorat, Université Joseph Fourier (Grenoble), janvier 1999, A paraître.

Ces travaux ont abouti aux premières versions "diffusables" de TRAIAN (voir § 8.1.2).

Nous avons établi une collaboration avec Pierre Wodey et Fabrice Baray (laboratoire ISIMA/LIMOS, Clermond-Ferrand) qui ont choisi d'utiliser LOTOS NT et TRAIAN comme base pour leurs travaux relatifs à la conception conjointe matériel-logiciel (*codesign*). Dans ce cadre, ils ont développé deux outils directement connectés en aval de TRAIAN : un transformateur de descriptions LOTOS NT procédant par raffinements successifs et un traducteur de LOTOS NT vers du code VHDL synthétisable (c'est-à-dire du code VHDL au niveau transfert de registres).

5.2.4 Amélioration du générateur de compilateurs FNC-2

Participants : Xavier Bouchoux, Christophe Discours, Hubert Garavel, Mark Jorgensen, Mihaela Sighireanu.

Nous avons basé le développement de nos compilateurs SVL (voir § 5.1.4) et TRAIAN (voir § 4.2 et 5.2.3) 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 SYNTAX/FNC-2, nous avons collaboré avec Didier Parigot afin d'améliorer notablement ce système. En 1998, notre contribution a porté sur les points suivants :

- Nous avons œuvré pour que la distribution officielle de SYNTAX/FNC-2 devienne "multi-architectures", c'est-à-dire qu'elle soit organisée de manière à permettre son utilisation simultanée sur des machines et des systèmes d'exploitation différents.

- Nous avons créé un site Web contenant un forum aux questions (FAQ) relatif à FNC-2 afin de documenter les problèmes les plus couramment rencontrés dans l'utilisation de ce système.

- Sur ce site Web, nous avons aussi dressé la liste complète des bogues de FNC-2 découvertes par notre équipe. En décembre 1998, cette liste répertoriait 45 bogues, dont 21 ont été corrigées.

- Nous avons mis en place pour FNC-2 une base de tests de non-régression indispensable à la stabilité de nos développements.

5.3 Etudes de cas et applications pratiques

Mots clés : algorithme réparti, application critique, application répartie, architecture multiprocesseur, architecture parallèle, atomicité, automate, cohérence de caches, génération de code, génération de test, génie logiciel, logique temporelle, mémoire répartie, modélisation, parallélisme asynchrone, protocole de communication, spécification formelle, synchronisation, 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 1998, nous avons traité principalement deux études de cas relatives aux architectures multiprocesseurs dans le cadre de la coopération BULL/INRIA, mais aussi d'autres applications dans des domaines très divers.

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

Participants : Ghassan Chehaibar, Christophe Discours, Hubert Garavel, Massimo Zendri.

Nous avons poursuivi les travaux entrepris en 1997 concernant le protocole de cohérence de caches de l'architecture CC-NUMA (*Cache Coherent Non Uniform Memory Access*) "POLYKID" développée par le centre BULL de Pregnana (Italie). En 1998, les travaux relatifs à POLYKID se sont orientés autour de deux axes principaux :

Spécification et vérification formelle : La description LOTOS modélisant le fonctionnement de l'architecture POLYKID, élaborée en 1997, a été complétée afin de prendre en compte les transferts de données. Ceci a permis de détecter une nouvelle erreur de cohérence des données dans le protocole.

Ensuite, la description LOTOS du protocole de cohérence de caches de POLYKID a été enrichie, de manière à intégrer les informations nécessaires à la génération de tests.

Cet effort de modélisation a mis en évidence la nécessité d'un outil de mise au point plus évolué, qui permettrait notamment une simulation guidée et un retour d'information sur le texte source des descriptions analysées. Ceci a motivé le développement du nouvel outil de simulation OCIS (voir 5.1.2).

Génération de tests : Nous avons continué l'expérimentation de l'outil de génération automatique de tests TGV (développé par le projet PAMPA et le laboratoire VERIMAG) qui permet de produire des tests définis par un objectif de test.

Afin de remédier à certaines limitations actuelles de TGV, nous avons développé deux nouveaux outils BCG_SPLIT et BCG_LOOP (voir § 5.1.3) qui permettent l'exploitation des tests générés par TGV.

Nous avons pu ainsi produire automatiquement 200 tests environ qui couvrent la totalité du plan de test prévu pour POLYKID. Ce résultat montre que notre approche de génération automatique de tests constitue une alternative crédible par rapport à la méthodologie traditionnelle, qui combine l'écriture manuelle de tests et la génération aléatoire de tests.

Notre travail a ensuite porté sur la production automatique de *tests généralisés* capables de traiter des situations plus complexes (chaque test généralisé représentant plusieurs centaines, voire plusieurs milliers de tests simples). Pour POLYKID, nous avons produit une dizaine de tests généralisés qui améliorent la couverture du plan de tests.

Nous avons établi une connexion entre les outils CADP et TGV, d'une part, et l'environnement de test SYNOPSYS utilisé par les développeurs de BULL, d'autre part. Elle permet au noyau de simulation de l'outil VSS (VHDL SYNOPSYS SIMULATOR) d'exécuter

en temps-réel les tests produits automatiquement. Ce travail intègre des modules développés à Rennes par le projet PAMPA (environ 10 000 lignes de code C), des modules développés à Pregnana et à Grenoble (environ 8 000 lignes de code C) et des modules de code VHDL développés à Pregnana, l'intégration des différents modules du système ayant été faite à Grenoble par M. Zendri.

L'environnement de test ainsi obtenu a été expérimenté sur l'implémentation en VHDL du protocole de cohérence de caches POLYKID développée à Pregnana — version 2.0 du circuit RCC (*Remote Cache Controller*). Ceci a permis de détecter cinq incohérences sérieuses (concernant la mise à jour des caches) entre la description LOTOS du protocole et l'implémentation réelle du circuit RCC.

Ce travail a donné lieu à deux publications, l'une parue en 1998 [6], l'autre à paraître en 1999 [VKZ99].

Notre travail sur l'architecture POLYKID s'est terminé en septembre 1998. Même si la direction de BULL n'a pas transformé le projet POLYKID en un produit commercial, les résultats obtenus par l'action VASY de DYADE (spécification formelle, émulation, vérification et génération de tests) sont considérés comme positifs et ont justifié la poursuite de la collaboration entre BULL et l'INRIA.

5.3.2 Protocole de cohérence de caches CC-NUMA "Fame"

Participants : Hubert Garavel, Massimo Zendri.

Le travail sur l'architecture POLYKID achevé, nous avons cherché à mettre à profit l'expérience acquise avec POLYKID en travaillant sur une autre architecture multi-processeurs, l'architecture FAME en cours de conception par le centre BULL des Clayes-sous-Bois (France). Ce travail reprend, dans un contexte différent, plusieurs des axes de recherche précédemment explorés avec succès pour l'architecture POLYKID :

Spécification formelle : Nous avons produit une description formelle du protocole de caches de FAME destinée à être utilisée pour la génération de tests. Contrairement au cas de POLYKID, pour lequel le point de départ de l'élaboration de la description formelle LOTOS était un document informel rédigé en langage naturel, nous nous sommes basés sur une description de FAME fournie sous la forme d'un programme écrit dans le langage d'entrée de l'outil MURPHI développé à l'Université Stanford.

Nous avons défini une méthode de traduction systématique permettant de convertir un programme MURPHI en une description LOTOS complétée par des fichiers écrits en C. Appliquée manuellement, cette méthode a permis d'obtenir rapidement une description du protocole de FAME pouvant être acceptée en entrée par les outils CADP et TGV.

[VKZ99] C. Viho, H. Kahlouche, M. Zendri, «Hardware-Testing using a Communication Protocol Conformance Testing Tool», *in : Proceedings of the Fith International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'99 (Amsterdam, The Netherlands)*, R. Cleaveland (éditeur), mars 1999. A paraître.

Génération de tests : Ensuite, nous avons appliqué à cette description les outils TGV et BCG_SPLIT afin de produire des jeux de tests utilisables pour le protocole de FAME. Les résultats obtenus ont été jugés positivement et ce travail devrait être poursuivi l'année prochaine.

5.3.3 Autres études de cas

Participants : Hubert Garavel, Radu Mateescu, Charles Pecheur, Mihaela Sighireanu, Massimo Zendri.

Nous avons utilisé les outils CADP et TRAIAN pour traiter plusieurs autres applications :

- Notre travail antérieur sur le système de mémoire virtuelle distribuée CFS (*Cluster File System*) développé au sein de l'action MESCALINE du GIE DYADE a fait l'objet d'un rapport de recherche [11].
- Nous avons modélisé en LOTOS NT et vérifié le comportement d'un système de gestion de stocks [9, 12].
- Nous avons modélisé en LOTOS et vérifié le comportement du protocole SCSI-2. Cet exemple servira d'exemple commun pour les équipes participant à l'action de recherche coopérative VERDON (voir § 7.1.1).
- Nous avons modélisé en LOTOS et en MURPHI le comportement d'un protocole de cache simplifié, mais réaliste, qui nous permet d'effectuer une comparaison entre les possibilités offertes par les différents outils de vérification.

D'autres équipes ont également utilisé la boîte à outils CADP pour diverses études de cas ou pour le développement d'outils au moyen des environnements OPEN/CÆSAR et BCG. Pour ne citer que les travaux publiés, on peut mentionner :

- la spécification et la vérification d'un outil de conception coopérative pour l'ingénierie des besoins (Université de Glasgow, Ecosse) [SJ98] ;
- l'analyse de performance d'un système téléphonique POTS (*Plain Old Telephony System*) à l'aide de chaînes de Markov (Université d'Erlangen, Allemagne) [HK99] ;
- la modélisation et la vérification de circuits asynchrones (réseaux combinatoires de portes XOR, compteur modulo 3) (Technion, Israël) [YG98,Yoe98] ;

[SJ98] M. Sage, C. Johnson, « Pragmatic Formal Design: A Case Study in Integrating Formal Methods into the HCI Development Cycle », *in : Proceedings of the 5th International Eurographics Workshop on Design, Specification and Verification of Interactive Systems DSV-IS '98 (Abingdon, UK)*, D. Duce, P. Johnson, P. Markopoulos (éditeurs), p. 134–154, juin 1998.

[HK99] H. Hermanns, J.-P. Katoen, « Automated Compositional Markov Chain Generation for a Plain-Old Telephone System», *Science of Computer Programming*, 1999, A paraître.

[YG98] M. Yoeli, A. Ginzburg, « LOTOS-Based Verification of Asynchronous Circuits », *Technical Report n° TR CS0951*, Technion, Computer Science Department, Haifa, Israel, janvier 1998.

[Yoe98] M. Yoeli, « Modulo-3 Transition Counter: A Case Study in LOTOS-Based Verification », *Technical Report n° TR CS0950*, Technion, Computer Science Department, Haifa, Israel, février 1998.

- la spécification et la vérification d'interfaces homme-machine (Queen Mary and Westfield College, Université de Londres, Royaume-Uni) [Mar97,MJR98] ;
- la génération de tests pour un protocole d'élection sur un anneau (Universités de Cottbus et de Magdebourg, Allemagne) [Ulr97,UK97] ;
- la spécification et la vérification d'un protocole de communication utilisé dans un système de surveillance du trafic routier (Université d'Eindhoven, Pays-Bas) [Wil98] ;
- la détection d'interactions entre services téléphoniques (Université d'Ottawa, Ontario, Canada) ; en octobre 1998, ce travail a obtenu le 2e prix au *Feature Interaction Contest* organisé dans le cadre du *5th International Workshop on Feature Interactions* (Lund, Suède).

6 Contrats industriels (nationaux, européens et internationaux)

6.1 Action Vasy (Dyade)

Participants : Ghassan Chehaibar, Christophe Discours, Hubert Garavel, Mark Jorgensen, Massimo Zendri.

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.

L'action Vasy du Gie Bull-Inria Dyade a pour objectif l'utilisation des méthodes formelles pour la validation et le test des architectures multiprocesseurs développées par Bull. Cette action, à laquelle participe également le projet Pampa (Rennes), est coordonnée par un ingénieur Bull installé dans les locaux de l'Unité de Recherche Inria Rhône-Alpes (Gh. Chehaibar, puis M. Zendri à partir de mars 1998). Entre 1996 et 1998, les travaux de Vasy ont

[Mar97] P. Markopoulos, *A Compositional Model for the Formal Specification of User Interfaces*, Phd thesis, Queen Mary and Westfield College, University of London, mars 1997.

[MJR98] P. Markopoulos, P. Johnson, J. Rowson, «Formal Architectural Abstractions for Interactive Software», *International Journal on Human-Computer Studies 49*, 5, décembre 1998, p. 675–715.

[Ulr97] A. Ulrich, «A description model to support test suite derivation for concurrent systems», *in : Kommunikation in Verteilten Systemen, GI/ITG-Fachtagung KiVS'97 (Braunschweig, Germany)*, M. Zitterbart (éditeur), Springer Verlag, p. 151–166, 1997.

[UK97] A. Ulrich, H. König, «Specification-based Testing of Concurrent Systems», *in : Proceedings of the IFIP Joint International Conference on Formal Description Techniques and Protocol Specification, Testing, and Verification FORTE/PSTV'97 (Ozaka, Japan)*, T. Higashino, A. Togashi (éditeurs), Chapman & Hall, novembre 1997.

[Wil98] T. Willemse, *The Specification and Validation of the OM/RR-Protocol*, Mémoire, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, juin 1998.

porté sur trois études de cas successives :

- l'arbitre de bus de l'architecture PowerScale mise en œuvre dans les stations de travail et serveurs de la gamme Escala,
- le protocole de cohérence de caches de Polykid, une architecture multi-processeurs Cc-Numa développée au centre Bull de Pregnana (voir § 5.3.1),
- le protocole de cohérence de caches Fame, une architecture multi-processeurs Cc-Numa développée au centre Bull des Clayes-sous-Bois (voir § 5.3.2).

L'expérience acquise durant ces études de cas a conduit à de notables progrès concernant les outils Cadp (voir § 5.1 pour les améliorations apportées en 1998).

7 Actions régionales, nationales et internationales

7.1 Actions nationales

7.1.1 Groupes de travail nationaux

Nous coordonnons l'action de recherche coopérative Verdon (Vérification et test de systèmes réactifs critiques comportant des données) à laquelle participent également les projets Meije et Pampa ainsi que le laboratoire Lsr-Imag.

7.1.2 Relations bilatérales nationales

En 1998, nous avons collaboré avec plusieurs projets Inria :

MEIJE (Sophia-Antipolis) : interconnexion des outils Cadp avec les outils de vérification Fc2Tools développés par Meije et action de recherche coopérative Verdon ;

OSCAR (Rocquencourt) : utilisation de l'outil Syntax/Fnc-2 développé par Oscar, suggestions pour l'amélioration de cet outil et contribution à son portage sous le système Linux ;

PAMPA (Rennes) : collaboration au sein du Gie Dyade portant sur l'utilisation et l'amélioration de l'outil Tgv et action de recherche coopérative Verdon ;

SIRAC (Rhône-Alpes) : application des outils Cadp à la validation du système de fichiers Cfs développé par Sirac et l'action Mescaline du Gie Dyade.

Nous entretenons également des relations scientifiques avec d'autres équipes :

Laboratoire ISIMA/LIMOS (Clermont-Ferrand) : méthodes de conception conjointe matériel-logiciel combinant Lotos NT et Vhdl (Pierre Wodey et Fabrice Baray) ;

Laboratoire LIP (Lyon) : comparaison des approches de preuve et de vérification par modèles (Pierre Lescanne, Eva et Kristoffer Rose) ;

Laboratoire LSR-IMAG (Grenoble) : application et comparaison de différentes méthodes formelles sur des exemples communs ; action de recherche coopérative VERDON (Didier Bert et Marie-Laure Potet) ;

Laboratoire VERIMAG (Grenoble) : coopération pour le développement conjoint des outils CADP (Laurent Mounier).

7.2 Actions internationales

7.2.1 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 pour effectuer un stage post-doctoral au CWI (Amsterdam) de décembre 1997 à septembre 1998.

- Nous participons à l'action de normalisation intitulée "*Enhancements to* LOTOS" dans le cadre du groupe de travail ISO/IEC JTC1/SC7/WG14. Pour ces travaux, H. Garavel est délégué AFNOR (*Association Française de Normalisation*) et M. Sighireanu est déléguée RSI (*Romanian Standard Institute*).

7.2.2 Relations bilatérales internationales

Nous entretenons des relations scientifiques avec plusieurs universités et centres de recherche internationaux, notamment :

En Europe : l'Université Libre de Bruxelles (Thierry Massart et Christian Hernalsteen), l'équipe SEN2 du CWI (Jan-Friso Groote et Judi Romijn), l'Université de Dortmund (Bernard Steffen et Volker Braun), l'Université de Stirling (Ken Turner et Ji He), et l'Université de Twente (Jan Tretmans et Axel Belinfante).

En Amérique du Nord : l'Université d'Etat de Caroline du Nord à Raleigh (Rance Cleaveland).

En Amérique du Sud : l'Université Fédérale de Sainte-Catherine à Florianopolis au Brésil (Mirela Sechi Moretti Annoni Notare).

7.3 Accueil de chercheurs étrangers

- Rance Cleaveland, professeur à l'Université d'Etat de Caroline du Nord à Raleigh (actuellement professeur à l'Université d'Etat de New York à Stony Brook), nous a rendu visite du 7 au 10 avril 1998. Il a donné un séminaire sur l'introduction de priorités dans l'algèbre de processus CCS.

- Jan-Friso Groote, qui dirige l'équipe SEN2 au CWI (Amsterdam), nous a rendu visite du 16 au 20 septembre 1998. Il a donné un séminaire sur la linéarisation des processus μCRL parallèles.

8 Diffusion de résultats

8.1 Diffusion de logiciels

8.1.1 Diffusion de la boîte à outils CADP

En 1998, nous avons accompli un effort significatif en vue d'améliorer la visibilité de la boîte à outils CADP et d'accroître sa diffusion :

- Nous avons finalisé le portage de CADP vers les machines de type Intel fonctionnant sous le système d'exploitation LINUX.
- Nous avons entrepris le portage de CADP vers les systèmes d'exploitation WINDOWS 98 et WINDOWS NT (travail d'Aldo Mazzilli).
- Nous avons mis à jour et enrichi la page Web consacrée à CADP (voir `http://www.inrialpes.fr/vasy/cadp`) que nous avions créée en 1997. Les améliorations portent sur les points suivants :
  - documentations techniques et publications associées ;
  - forum aux questions (FAQ) ;
  - liste des études de cas réalisées avec CADP ;
  - démonstrations disponibles en ligne.
- Tout au long de l'année 1998, nous avons réalisé et diffusé des mises à jour incrémentales de la boîte à outils CADP version 97b "Liège" diffusée en décembre 1997.
- Nous avons effectué plusieurs démonstrations publiques de la boîte à outils, notamment à l'occasion des colloques et conférences TACAS'98 (Lisbonne, mars 1998), FMICS'98 (Amsterdam, mai 1998), STTT'98 (Aalborg, juillet 1998) et au cours des rencontres INRIA-Industrie (Paris, novembre 1998).
- Nous avons collaboré avec l'Université de Dortmund pour intégrer les outils CADP dans la plate-forme logicielle ETI (*Electronic Tool Integration*) accessible en ligne par Internet [SMB97,BMW97,MBK97].

Ces efforts semblent avoir porté leurs fruits puisqu'en 1998, le nombre de licences site pour CADP est passé de 148 à 172.

[SMB97] B. Steffen, T. Margaria, V. Braun, «The Electronic Tool Integration Platform: Concepts and Design», *Springer International Journal on Software Tools for Technology Transfer (STTT) 1–2*, 1, décembre 1997, p. 9–30.

[BMW97] V. Braun, T. Margaria, C. Weise, «Integrating Tools in the ETI Platform», *Springer International Journal on Software Tools for Technology Transfer (STTT) 1–2*, 1, décembre 1997, p. 31–48.

[MBK97] T. Margaria, V. Braun, J. Kreileder, «Interacting with ETI: a User Session», *Springer International Journal on Software Tools for Technology Transfer (STTT) 1–2*, 1, décembre 1997, p. 49–63.

8.1.2 Diffusion du compilateur TRAIAN

En 1998, nous avons commencé à distribuer les premières versions du compilateur TRAIAN pour LOTOS NT :

- Nous avons créé une page Web spécialement consacrée à TRAIAN (voir `http://www.inrialpes.fr/vasy/traian`) à partir de laquelle on peut télécharger le compilateur en mode FTP anonyme.

- Nous avons préparé et distribué trois versions successives de TRAIAN : une version 0.1 (février 1998), une version 0.5 (septembre 1998) et une version 1.0 (décembre 1998).

Pour l'année 1998, le nombre de téléchargements de TRAIAN (versions 0.1 et 0.5 confondues) s'élève à 23, ce qui constitue un début prometteur.

8.2 Animation de la communauté scientifique

- H. Garavel est responsable de l'action de recherche coopérative VERDON (Vérification et test de systèmes réactifs critiques comportant des données) initiée par la Direction Scientifique de l'INRIA.

- 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 Guy Leduc (Université de Liège), il a contribué à la préparation d'un numéro thématique de cette revue consacré à l'ingénierie des protocoles (à paraître en 1999).

- H. Garavel est membre du comité de programme d'ARTS'99 (*5th International AMAST Workshop on Real-Time and Probabilistic Systems*, Bamberg, Allemagne, 26–28 mai 1999).

- H. Garavel est membre du comité de programme de CFIP'99 (7e Colloque Francophone sur l'Ingénierie des Protocoles, Nancy, France, 26–29 avril 1999).

- H. Garavel est membre du comité de programme de TACAS'99 (*5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Amsterdam, Pays-Bas, 22–26 mars 1999).

- H. Garavel a participé au jury de thèse de Christian Hernalsteen [Her98] à l'Université Libre de Bruxelles en juin 1998.

- H. Garavel est membre de la commission de spécialistes (CSE) de l'Institut National Polytechnique de Grenoble (sections 26 et 27).

[Her98] C. Hernalsteen, *Specification, Validation and Verification of Real-Time Systems using ET-LOTOS*, Thèse de doctorat, Université Libre de Bruxelles, juin 1998.

8.3 Enseignement universitaire

- H. Garavel a dispensé le cours "Temps Réel" destiné aux étudiants en 3e année de l'ENSIMAG (24 heures annuelles).
- H. Garavel a créé un cours "Spécification et vérification de protocoles" destiné aux étudiants du DEA d'informatique de l'Université de Savoie (12 heures annuelles).
- H. Garavel a organisé quatre séances de travaux pratiques consacrées à la spécification et la validation de protocoles à l'ENST Paris (12 heures annuelles).
- M. Sighireanu est monitrice à l'Université Joseph Fourier. Elle a participé à l'enseignement de langages de programmation à l'Ecole d'Informatique (niveau licence).
- H. Garavel a encadré les mémoires de probatoire d'Aldo Mazzilli et Jean Dina, élèves-ingénieurs CNAM (centre agréé de Grenoble).

8.4 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). En outre :

- H. Garavel et M. Sighireanu ont donné deux séminaires au cours de la première réunion de l'action VERDON à l'Ecole des Mines de Paris, le 11 février 1998.
- M. Sighireanu a présenté ses travaux de recherche à l'équipe PLUME de l'Ecole Normale Supérieure de Lyon le 31 mars 1998.
- H. Garavel et M. Sighireanu ont présenté un article invité [3] dans le cadre du colloque FMICS'98 (*3rd International Workshop on Formal Methods for Industrial Critical Systems*) à Amsterdam (Pays-Bas) le 26 mai 1998.
- R. Mateescu a donné un séminaire invité à l'Université de Twente (Pays-Bas) le 17 juillet 1998.
- H. Garavel, R. Mateescu et M. Sighireanu ont donné trois séminaires au cours de la deuxième réunion de l'action VERDON à l'INRIA Sophia-Antipolis le 15 septembre 1998.
- H. Garavel et R. Mateescu ont présenté les résultats de l'action VASY-RA au cours du séminaire d'évaluation du programme 1C de l'INRIA à Saint-Malo les 6 et 7 octobre 1998.

9 Bibliographie

[1] G. Chehaibar, H. Garavel, L. Mounier, N. Tawbi, F. Zulian, «Specification and Verification of the PowerScale Bus Arbitration Protocol: An Industrial Experiment with LOTOS», *in : 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), R. Gotzhein, J. Bredereke (éditeurs), IFIP, Chapman & Hall, p. 435–450, octobre 1996. Full version available as INRIA Research Report RR-2958.

[2] J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, M. Sighireanu, «CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox», *in : Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, R. Alur, T. A. Henzinger (éditeurs), *Lecture Notes in Computer Science*, *1102*, Springer Verlag, p. 437–440, août 1996.

[3] 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 (éditeur), juin 1997.

[4] H. Garavel, J. Sifakis, «Compilation and Verification of LOTOS Specifications», *in : Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*, L. Logrippo, R. L. Probert, H. Ural (éditeurs), IFIP, North-Holland, p. 379–394, juin 1990.

[5] H. Garavel, *Compilation et vérification de programmes LOTOS*, Thèse de doctorat, Université Joseph Fourier (Grenoble), novembre 1989.

[6] H. Garavel, «Compilation of LOTOS Abstract Data Types», *in : Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*, S. T. Vuong (éditeur), North-Holland, p. 147–162, décembre 1989.

[7] H. Garavel, «An Overview of the Eucalyptus Toolbox», *in : Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, Z. Brezočnik, T. Kapus (éditeurs), University of Maribor, Slovenia, p. 76–88, juin 1996.

Thèses et habilitations à diriger des recherches

[1] R. Mateescu, *Vérification des propriétés temporelles des programmes parallèles*, Thèse de doctorat, Institut National Polytechnique de Grenoble, avril 1998.

Articles et chapitres de livre

[2] M. Sighireanu, R. Mateescu, «Verification of the Link Layer Protocol of the IEEE-1394 Serial Bus (FireWire): an Experiment with E-LOTOS», *Springer International Journal on Software Tools for Technology Transfer (STTT) 2*, 1, décembre 1998, p. 68–88.

Communications à des congrès, colloques, etc.

[3] H. Garavel, M. Sighireanu, «Towards a Second Generation of Formal Description Techniques – Rationale for the Design of E-LOTOS», *in : Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)*, J.-F. Groote, B. Luttik, J. Wamel (réd.), CWI, p. 187–230, Amsterdam, mai 1998. Invited lecture.

[4] H. Garavel, «OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing», *in : Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*, B. Steffen (réd.), *Lecture Notes in Computer Science*, *1384*, Springer Verlag, p. 68–84, Berlin, mars 1998.

[5] W. Janssen, R. Mateescu, S. Mauw, J. Springintveld, «Verifying Business Processes using SPIN», *in : Proceedings of the 4th International SPIN Workshop (Paris, France)*, G. Holzmann, E. Najm, A. Serhrouchni (réd.), ENST, p. 21–36, Paris, novembre 1998.

[6] H. Kahlouche, C. Viho, M. Zendri, «An Industrial Experiment in Automatic Generation of Executable Test Suites for a Cache Coherency Protocol», *in : Proceedings of the IFIP 11th International Workshop on Testing of Communicating Systems IWTCS'98 (Tomsk, Russia)*, A. Petrenko, N. Yevtushenko (réd.), Chapman & Hall, septembre 1998.

[7] R. Mateescu, H. Garavel, «XTL: A Meta-Language and Tool for Temporal Logic Model-Checking», *in : Proceedings of the International Workshop on Software Tools for Technology Transfer STTT'98 (Aalborg, Denmark)*, T. Margaria (réd.), BRICS, University of Aarhus, p. 33–42, Aarhus, juillet 1998.

[8] R. Mateescu, «Local Model-Checking of an Alternation-Free Value-Based Modal Mu-Calculus», *in : Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation VMCAI'98 (Pisa, Italy)*, A. Bossi, A. Cortesi, F. Levi (réd.), University Ca' Foscari, Venice, septembre 1998.

[9] M. Sighireanu, «Model-Checking Validation of the LOTOS Descriptions of the Invoicing Case Study», *in : Proceedings of the International Workshop on Comparing System Specification Techniques (Nantes, France)*, H. Habrias (réd.), p. 1998, mars 1998.

Rapports de recherche et publications internes

[10] H. Garavel, «OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing», *Research Report n° RR-3352*, INRIA, Grenoble, mars 1998.

[11] C. Pecheur, «Advanced Modelling and Verification Techniques Applied to a Cluster File System», *Research Report n° RR-3416*, INRIA, Grenoble, mai 1998.

[12] M. Sighireanu, K. Turner, «Requirement Capture, Formal Description and Verification of an Invoicing System», *Research Report n° RR-3575*, INRIA, Grenoble, décembre 1998.