INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Action VASY-RA

Validation de Systèmes – Recherche et Applications

Rhône-Alpes

THÈME 1C

Table des matières

1 Composition de l'équipe

Responsable scientifique

Hubert Garavel [CR1 INRIA]

Assistantes de projet

Joanna Payart

Béatrice Claudio

Personnel Inria

Radu Mateescu [CR2 INRIA]

Personnel Bull

Massimo Zendri [responsable d'action DYADE]

Ingénieurs experts

Christophe Discours [ingénieur DYADE, jusqu'au 31 janvier 1999]

Moëz Cherif [ingénieur Reutel (ALCATEL), depuis le 1er avril 1999]

Marc Herbert [ingénieur DYADE, depuis le 1er mai 1999]

Chercheur post-doctorant

Irina Smarandache [bourse INRIA Rhône-Alpes, depuis le 1er novembre 1999]

Chercheurs honoraires

Charles Pecheur [chercheur RIACS, NASA Ames Research Center, USA]

Mihaela Sighireanu [Maître de Conférences, LIAFA, Université Paris 7]

Stagiaires

Manuel Aguilar Cornejo [DEA, jusqu'au 30 juin 1999]

Claude Chaudet [élève-ingénieur IIE-CNAM, du 1er janvier au 30 juin 1999]

Aldo Mazzilli [élève-ingénieur CNAM, jusqu'au 30 septembre 1999]

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*, c'est-à-dire tout système dont on peut modéliser le comportement parallèle par une sémantique d'entrelacement des événements (*interleaving semantics*).

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 — à un certain niveau d'abstraction — d'un système avec ses *propriétés* qui décrivent le fonctionnement attendu du système (c'est-à-dire les services que le système doit fournir).

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 des descriptions de plus haut niveau que l'on traduit ensuite en 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 collaborons étroitement avec d'autres équipes qui développent des outils basés sur les relations d'équivalence et de préordre. En outre, depuis 1999, nous développons notre propre outil qui implémente diverses relations de bisimulation.

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 un ensemble de 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 cela conduit bien souvent à des ambiguïtés et des divergences d'interprétation (notamment entre concepteurs et implémenteurs).
- En général, les techniques de *model-checking* ne peuvent garantir la correction d'un système infini, puisqu'elles ne peuvent vérifier que des abstractions finies de ce système. C'est pourquoi on doit utiliser aussi des techniques de preuve, lesquelles ne s'appliquent qu'aux langages ayant une sémantique formelle.

Dans ce contexte, nos travaux actuels portent sur trois langages :

- Nous nous intéressons depuis longtemps 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.

  Nous utilisons LOTOS pour diverses études de cas industrielles et nous développons des outils logiciels pour ce langage dans le cadre de la boîte à outils CADP (voir § 4.1).
- Les algèbres de processus sont des formalismes particulièrement bien 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. Elles se voient souvent supplantées par des langages dont l'apparence plus facile (syntaxe graphique ou proche des langages algorithmiques classiques) masque souvent une absence

[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.

de sémantique formelle assez inquiétante lorsqu'il s'agit de modéliser et de valider des systèmes critiques.

Les besoins en méthodes formelles et vérification allant en croissant, il est nécessaire de réfléchir à de nouveaux langages qui combineraient les fondements théoriques rigoureux et l'expressivité des algèbres de processus avec une simplicité d'utilisation permettant d'assurer une meilleure diffusion industrielle.

Cette réflexion est également guidée par l'apparition de protocoles à contraintes temporelles fortes — protocoles utilisés dans les réseaux à haut débit — pour lesquels les aspects temporels doivent être pris en compte de manière quantitative, et non plus seulement qualitative.

Nous travaillons sur ces questions, notamment dans le cadre de la révision de la norme LOTOS entreprise à l'ISO depuis 1992 : cette révision devrait conduire à un nouveau langage nommé E-LOTOS (*Extended*-LOTOS) [Que98] qui vise à conjuguer une expressivité sémantique accrue (par exemple, avec l'introduction du temps quantifié) et une facilité d'apprentissage pour des non-experts. L'historique de nos contributions à l'ISO est disponible sur notre serveur Web, à l'adresse `http://www.inrialpes.fr/vasy/elotos`.

En parallèle, nous étudions une variante d'E-LOTOS, appelée LOTOS NT (LOTOS *Nouvelle Technologie*) [5, 13], dans laquelle nous avons introduit les concepts qui nous semblent pertinents (ce qui n'est pas toujours chose aisée dans une norme internationale).

Comme E-LOTOS, LOTOS NT se compose de trois parties : une *partie données*, qui permet une description naturelle des types de données et des fonctions tout en étant facilement analysable et implémentable, 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 une *partie modules*, qui autorise la structuration et la réutilisation des descriptions LOTOS NT.

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

Nous travaillons sur l'implémentation de LOTOS NT, pour lequel nous développons le compilateur TRAIAN (voir § 4.2).

– Enfin, dans le cadre d'une collaboration récente avec ALCATEL et le projet PAMPA (Rennes), nous nous intéressons à la notation UML pour la modélisation des systèmes informatiques. Issue des travaux de Booch, Jacobson et Rumbaugh, UML constitue le successeur commun de trois méthodes de développement à objets très utilisées, dans un souci d'unification destiné à faciliter le développement d'outils inter-opérables.

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

Dans ce contexte, nous cherchons à réutiliser au mieux les outils de CADP, en les connectant et en les adaptant à UML.

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.

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 conduites ces dernières années avec la boîte à outils CADP (voir notamment § 5.3.3) illustrent bien cette diversité applicative :

- **architectures matérielles :** arbitrage de bus, cohérence de caches, conception conjointe matériel-logiciel ;
- **bases de données :** protocoles transactionnels, bases de connaissances distribuées, gestion de stocks ;
- **électronique grand public :** télécommandes audiovisuelles, vidéo à la demande, bus FIREWIRE, réseaux locaux domestiques ;
- **protocoles de sécurité :** authentification, commerce électronique, distribution de clés cryptographiques ;
- **systèmes embarqués :** contrôle de trafic aérien ;
- **systèmes répartis :** mémoire virtuelle, systèmes de fichiers répartis, ingénierie concourante, algorithmes d'élection ;
- **télécommunications :** réseaux à haut débit, administration de réseaux, téléphonie mobile, 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], Moëz Cherif, Marc Herbert, Radu Mateescu, Aldo Mazzilli, Mihaela Sighireanu.

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, 8, 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.

- OPEN/CÆSAR [9] est un environnement logiciel extensible permettant de développer des outils de simulation, de vérification et de génération de tests 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. De ce point de vue, l'environnement OPEN/CÆSAR est l'un des constituants essentiels de la boîte à outils CADP, puisqu'il effectue la jonction entre les outils dédiés aux langages et les outils opérant sur les modèles. L'environnement OPEN/CÆSAR comprend un ensemble de bibliothèques avec leurs interfaces de programmation, ainsi que divers outils parmi lesquels :

  - EVALUATOR, qui évalue à la volée des formules de μ-calcul régulier sans alternance,
  - EXECUTOR, qui permet l'exécution aléatoire,
  - EXHIBITOR, qui recherche des séquences d'exécution caractérisées par une expression régulière,
  - GENERATOR et REDUCTOR, qui construisent le graphe des états accessibles,
  - SIMULATOR et XSIMULATOR, qui permettent la simulation interactive, et

  - TERMINATOR, qui recherche les états de blocage.

- 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 joue un rôle central dans la boîte à outils CADP. Il 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 le format BCG, nous développons un environnement logiciel qui se compose de bibliothèques avec leurs interfaces de programmation et de plusieurs outils, notamment :

  - BCG_DRAW, qui permet d'afficher en PostScript une représentation 2D d'un graphe,
  - BCG_EDIT, qui permet de modifier interactivement la représentation graphique produite par BCG_DRAW,
  - BCG_IO, qui effectue des conversions entre BCG et d'autres formats d'automates,
  - BCG_LABELS, qui permet de masquer et/ou de renommer par des expressions régulières les étiquettes d'un graphe BCG,
  - BCG_MIN, qui permet de minimiser des graphes BCG selon la bisimulation forte ou la bisimulation de branchement (éventuellement étendue au cas des systèmes probabilistes ou stochastiques), et
  - BCG_OPEN, qui permet d'appliquer à tout graphe BCG les outils disponibles dans l'environnement OPEN/CÆSAR.

- 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.

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.

[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.

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 masquant 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" devrait 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 [correspondante], Claude Chaudet, Hubert Garavel, Marc Herbert.

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.

Nous développons un compilateur, appelé TRAIAN, pour LOTOS NT, dont le but est de 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 actuelle de TRAIAN effectue l'analyse lexicale et syntaxique, la construction des arbres de syntaxe abstraite, les vérifications de sémantique statique et la génération de code C pour les définitions de types et de fonctions contenues dans les descriptions LOTOS NT.

Depuis 1998, le compilateur TRAIAN est diffusé sur Internet : il existe une page Web qui lui est consacrée (voir `http://www.inrialpes.fr/vasy/traian`) à partir de laquelle on peut télécharger librement le compilateur.

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 1999, nos travaux sur la vérification ont porté sur l'extension et l'amélioration d'outils existants, ainsi que sur le développement de nouveaux outils visant à faciliter l'utilisation des techniques formelles en milieu industriel.*

5.1.1 Développement du simulateur OCIS

Participants : Moëz Cherif, Hubert Garavel.

Nous avons poursuivi nos travaux relatifs au simulateur OCIS (OPEN/CÆSAR *Interactive Simulator*). Fonctionnant soit en mode textuel (lignes de commandes), soit en mode graphique, OCIS est destiné à remplacer les outils existants SIMULATOR et XSIMULATOR, par rapport auxquels il introduit des améliorations importantes. 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.

Le développement d'OCIS a été entrepris en 1998 à la demande de nos partenaires de BULL souhaitant disposer d'un outil pour faciliter la mise au point des descriptions formelles de protocoles en LOTOS. Toutefois, OCIS n'est pas lié à un langage source précis : par construction, il doit être possible de l'utiliser pour tout langage comportant du parallélisme et des tâches communicantes, pourvu que le compilateur de ce langage implémente l'interface OPEN/CÆSAR [9].

En 1999, ce travail a pris un nouvel essor dans le cadre de notre participation au contrat REUTEL-2000 (voir § 6.3), puisqu'il s'agit de permettre l'utilisation d'OCIS sur des descriptions UML en l'interfaçant avec l'outil UMLAUT en cours de développement dans le projet PAMPA. Les principales améliorations apportées à OCIS en 1999 sont les suivantes :

- Plusieurs erreurs ont été corrigées et certaines parties du simulateur ont été réécrites afin de parvenir à un code plus simple et plus robuste.
- Les performances du simulateur ont été améliorées en supprimant certaines actions de visualisation inutiles, ainsi qu'en allégeant le protocole de communication entre le moteur de simulation (écrit en langage C) et l'interface graphique (développée avec les outils TCL/TK et TIX), ce qui a permis d'obtenir un temps de réponse quasi-instantané aux requêtes de l'utilisateur.
- Il est désormais permis de sauvegarder dans un fichier texte la séquence des commandes exécutées à partir de l'interface (textuelle ou graphique), puis de réexécuter le contenu de ce fichier (éventuellement au cours d'une session de simulation ultérieure).
- La possibilité de sauvegarder dans un fichier BCG le scénario de simulation — c'est-à-dire les parties du graphe explorées — a été affinée. Auparavant, on ne pouvait sauvegarder que la totalité du sous-graphe exploré. Désormais, on peut aussi sauvegarder le sous-arbre exploré à partir d'un état donné (éventuellement préfixé par le chemin conduisant à cet état à partir de l'état initial).
- La convivialité de l'interface utilisateur a été accrue, notamment en ce qui concerne la visualisation (grâce à un coloriage intuitif) des transitions franchies et de celles restant

à explorer, l'affichage du temps écoulé, l'historique des valeurs d'une variable à partir de l'état initial, etc.

L'outil OCIS sera intégré dans la prochaine version de la boîte à outils CADP.

5.1.2 Développement de l'outil BCG_MIN

Participants : Moëz Cherif, Hubert Garavel.

En étroite collaboration avec Holger Hermanns (Université de Twente, Pays-Bas), nous avons développé un nouvel outil de minimisation de graphes basé sur les bibliothèques logicielles de l'environnement BCG. Cet outil, appelé BCG_MIN, opère sur trois sortes de graphes (tous encodés dans le format BCG) :

- des systèmes de transitions "ordinaires", tels que ceux produits à partir de descriptions LOTOS,
- des systèmes de transitions "probabilistes" (connus aussi sous le nom de "processus de décision markoviens à temps discret"), qui comportent à la fois des transitions ordinaires et des transitions étiquetées par une probabilité $p \in [0, 1]$,
- des systèmes de transitions "stochastiques" (connus aussi sous le nom de "processus de décision markoviens à temps continu"), qui comportent à la fois des transitions ordinaires et des transitions étiquetées par un paramètre réel λ qui détermine une loi de la forme $prob(x > t) = e^{-\lambda t}$.

Pour les systèmes de transitions ordinaires, BCG_MIN implémente deux algorithmes de minimisation, l'un pour la bisimulation forte, basé sur l'algorithme de Kanellakis et Smolka [KS90], l'autre pour la bisimulation de branchement, basé sur l'algorithme de Groote et Vaandrager [GV90].

Pour les systèmes de transitions probabilistes et stochastiques, BCG_MIN implémente l'algorithme de Hermanns et Siegle [HS99].

Le développement de l'outil BCG_MIN s'est fait à partir d'un programme PASCAL de Jan-Friso Groote (CWI, Pays-Bas) qui implémentait l'algorithme de Groote et Vaandrager pour minimiser les systèmes de transitions ordinaires selon la bisimulation de branchement. Nous avions préalablement songé à nous baser sur d'autres outils de bisimulation tels qu'ALDÉBARAN ou FC2TOOLS, mais le code source de ces outils ne nous était pas accessible.

[KS90] P. C. KANELLAKIS, S. A. SMOLKA, «CCS expressions, finite state processes, and three problems of equivalence», *Information and Computation 86*, 1, mai 1990, p. 43–68.

[GV90] J. GROOTE, F. VAANDRAGER, «An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence», *in : Proceedings of the 17th ICALP (Warwick)*, M. S. Patterson (éditeur), *Lecture Notes in Computer Science, 443*, Springer Verlag, p. 626–638, 1990.

[HS99] H. HERMANNS, M. SIEGLE, «Bisimulation Algorithms for Stochastic Process Algebras and their BDD-based Implementation», *in : Proceedings of the 5th International AMAST Workshop ARTS'99 (Bamberg, Germany)*, J.-P. Katoen (éditeur), *Lecture Notes in Computer Science, 1601*, Springer Verlag, p. 244–265, mai 1999.

Nos premiers efforts ont porté sur la réécriture en langage C du programme de Jan-Friso Groote, sa connexion au format BCG, la prise en compte de la bisimulation forte, l'amélioration de certaines parties de code et la correction de quelques erreurs mineures. Holger Hermanns a ensuite ajouté le traitement des cas probabilistes et stochastiques.

L'outil BCG_MIN sera intégré dans la prochaine version de la boîte à outils CADP. Comparé à d'autres outils de vérification basés sur les bisimulations, tels qu'ALDÉBARAN et FC2TOOLS, les fonctionnalités de BCG_MIN sont plus limitées, puisque seule la minimisation est permise. Néanmoins, BCG_MIN constitue un complément utile aux outils existants, pour plusieurs raisons :

- Pour les systèmes de transitions ordinaires, BCG_MIN permet de minimiser efficacement des graphes de grande taille (plusieurs centaines de milliers d'états, plusieurs millions de transitions) que les autres outils ne parviennent pas à traiter par manque de mémoire. Par ailleurs, l'utilisation du format BCG en mode natif constitue un atout, car les fichiers BCG sont très compacts, ce qui entraîne des gains à la fois en place disque et en rapidité.

- BCG_MIN affiche les classes d'équivalence de manière plus précise, en montrant la relation exacte entre les numéros d'états du graphe à minimiser et ceux du graphe minimisé, ainsi qu'en affichant les τ-circuits dans le cas de la bisimulation de branchement.

- Enfin, BCG_MIN traite le cas des systèmes probabilistes et stochastiques.

5.1.3 Amélioration de l'évaluateur générique XTL

Participants : Radu Mateescu, Hubert Garavel.

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. Intégré dans CADP depuis septembre 1997, l'évaluateur XTL a continué d'être utilisé avec succès pour la validation d'applications industrielles : le protocole audio-vidéo HAVi de Philips [Rom99], une base de circuits numériques séquentiels [HT99b] et le système de mémoire virtuelle distribuée CFS [18].

En 1999, suite aux remarques des utilisateurs, nous avons corrigé plusieurs erreurs dans la version 1.1 et dans la documentation de l'outil XTL.

Par ailleurs, nous avons développé en XTL deux nouvelles bibliothèques implémentant des opérateurs qui s'avèrent nécessaires en pratique mais qui ne peuvent pas être exprimés au moyen des logiques temporelles classiques : un opérateur caractérisant les états non-déterministes d'un modèle et un opérateur calculant le plus court chemin allant de l'état initial jusqu'à un état

[Rom99] J. ROMIJN, *Analysing Industrial Protocols with Formal Methods*, thèse de doctorat, University of Twente, The Netherlands, septembre 1999.

[HT99b] J. HE, K. J. TURNER, «Specification and Verification of Synchronous Hardware using LOTOS», *in : Proceedings of the IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing, and Verification FORTE XII / PSTV XIX'99 (Beijing, China)*, J. Wu, S. T. Chanson, Q. Gao (éditeurs), IFIP, Kluwer Academic Publishers, p. 295–312, octobre 1999.

satisfaisant une propriété donnée. Ces deux primitives permettent d'obtenir des informations de diagnostic simples et concises sur les systèmes de transitions représentés en format BCG.

En collaboration avec Holger Hermanns de l'Université de Twente (Pays-Bas), nous envisageons le développement d'une bibliothèque XTL implémentant la logique temporelle probabiliste CSL [BKH99] permettant la vérification de chaînes de Markov. L'évaluation de cette logique nécessite des traitements numériques (calculs en virgule flottante, calculs vectoriels et matriciels, etc.). Or, le langage XTL offre un ensemble de types et de fonctions prédéfinis permettant de décrire aisément des opérateurs de logique temporelle, mais ne fournit pas de primitives pour effectuer des opérations vectorielles ou matricielles, celles-ci n'étant pas nécessaires pour l'évaluation des logiques temporelles "classiques".

C'est pourquoi nous avons entrepris en 1999 l'extension du langage et de l'outil XTL et de l'évaluateur. XTL 1.2 permet à l'utilisateur de déclarer dans une spécification XTL des types et des fonctions externes pour lesquels il doit fournir une implémentation en C ; celle-ci sera utilisée dans le code généré par l'outil pour évaluer les propriétés temporelles sur un modèle en format BCG. Il sera ainsi possible de réutiliser des bibliothèques mathématiques écrites en langage C pour implémenter la logique CSL. Ces nouveaux mécanismes ont nécessité des changements dans toutes les phases du compilateur XTL : analyse lexicale et syntaxique, analyse sémantique et génération de code.

5.1.4 Développement de l'outil EVALUATOR 3.0

Participants : Radu Mateescu, Mihaela Sighireanu.

Une manière de contourner le problème de l'explosion d'états est d'employer des techniques de vérification à la volée (voir § 2.2). Contrairement à l'approche énumérative, les techniques à la volée permettent de détecter des erreurs dans une spécification sans construire complètement le modèle sous-jacent, mais en le générant dynamiquement au fur et à mesure des besoins.

Un indicateur essentiel de l'efficacité d'un algorithme à la volée est sa complexité moyenne, qui doit être améliorée de façon à effectuer la vérification en générant une portion du modèle aussi petite que possible. Un autre aspect important est la possibilité de fournir un diagnostic (par exemple, une séquence de transitions dans le modèle) expliquant la valeur de vérité d'une propriété.

En 1999, nous avons entrepris la conception de plusieurs algorithmes de vérification à la volée réunissant les deux caractéristiques ci-dessus. Ces algorithmes sont basés sur la résolution de systèmes d'équations booléennes (SEBs), ce qui les rend suffisamment généraux pour traiter plusieurs problèmes de vérification traduisibles en termes de SEBs (vérification par bisimulations, préordres, logiques temporelles, etc.).

Nous avons mis en œuvre ces algorithmes dans l'outil EVALUATOR version 3.0, que nous avons développé en utilisant l'environnement OPEN/CÆSAR [9]. EVALUATOR 3.0 permet de vérifier à la volée des formules du μ-calcul modal sans alternance étendu avec des expressions

[BKH99] C. BAIER, J.-P. KATOEN, H. HERMANNS, « Approximate Symbolic Model Checking of Continuous-Time Markov Chains », *in : Proceedings of CONCUR'99: Concurrency Theory*, J. Baeten, S. Mauw (éditeurs), *Lecture Notes in Computer Science, 1664*, Springer Verlag, p. 146–162, Berlin, août 1999.

régulières comme celles de la logique PDL-Δ [Str82]. Par rapport à la précédente version 2.0 (développée par Marius Bozga et Laurent Mounier du laboratoire VERIMAG), EVALUATOR 3.0 apporte les améliorations suivantes :

- Le langage d'entrée d'EVALUATOR 3.0 est plus concis que celui de la version 2.0. Les formules d'actions peuvent maintenant être spécifiées au moyen d'opérateurs booléens et de prédicats de base sur les étiquettes du modèle (pour lesquels on peut employer des expressions régulières sur chaînes de caractères). Des séquences régulières de transitions peuvent désormais être décrites plus succinctement, au moyen d'expressions régulières construites sur le vocabulaire des formules d'actions. Dans la version 3.0, il est également possible de définir de nouveaux opérateurs temporels comme des macros paramétrées par des formules, et de les grouper dans des bibliothèques réutilisables.

- Les algorithmes que nous avons mis en œuvre dans EVALUATOR 3.0 sont plus efficaces que ceux de la version 2.0, car ils explorent (beaucoup) moins d'états du modèle avant de déterminer la valeur de vérité d'une formule. Sur une large base d'exemples que nous avons traitée avec les deux versions de l'outil, le temps d'exécution d'EVALUATOR 3.0 est uniformément meilleur que celui de la version 2.0 (d'au moins 50% et, dans certains cas, de plusieurs ordres de grandeur). En outre, le fait que le nombre d'états explorés soit généralement plus faible permet d'augmenter la taille des applications traitées.

- La qualité des diagnostics produits par EVALUATOR 3.0 est nettement meilleure que celle de la version 2.0. En effet, les nouveaux algorithmes que nous avons développés permettent d'exhiber des diagnostics (portions du modèle) expliquant la valeur de vérité d'une formule, en fournissant un exemple lorsque la propriété est vraie et un contre-exemple lorsqu'elle est fausse. En outre, les diagnostics produits sont minimaux, dans le sens où ils ne contiennent pas d'information redondante. EVALUATOR 3.0 permet également d'obtenir des diagnostics complets pour des logiques temporelles particulières comme CTL [CES86] et ACTL [NV90] et de chercher dans un modèle des séquences de transitions régulières comme diagnostics de formules PDL-Δ.

L'outil EVALUATOR 3.0 a été utilisé pour vérifier des tâches robotiques spécifiées en ESTEREL dans l'environnement ORCCAD développé au sein du projet BIP. Ce travail a aussi permis de constituer un catalogue de schémas de propriétés temporelles génériques (exprimées en ACTL) destinées à faciliter l'utilisation d'EVALUATOR 3.0 conjointement avec ORCCAD.

5.1.5 Parallélisation d'algorithmes de vérification

Participants : Hubert Garavel, Radu Mateescu, Irina Smarandache.

[Str82] R. STREETT, « Propositional Dynamic Logic of Looping and Converse », *Information and Control*, 54, 1982, p. 121–141.

[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.

[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.

En 1999, nous avons entrepris des travaux sur l'utilisation de machines parallèles pour améliorer les performances des algorithmes de vérification énumérative (travail post-doctoral d'I. Smarandache).

En effet, ces algorithmes — qui nécessitent l'exploration et le stockage de graphes de dimensions importantes (plusieurs millions d'états) — sont souvent limités par la puissance de calcul et l'espace mémoire disponibles sur les machines séquentielles actuelles.

Les travaux effectués en 1999 ont porté sur des recherches bibliographiques et sur l'étude d'une machine massivement parallèle (grappe de PCs interconnectés par un bus SCI) en cours de développement dans les projets APACHE et SIRAC.

5.1.6 Génération de tests

Participants : Hubert Garavel, Marc Herbert, Massimo Zendri.

TGV (*Test Generation with Verification technology*) est un outil de génération automatique de tests de conformité à partir de spécifications formelles. Développé par le projet PAMPA de l'INRIA Rennes et le laboratoire VERIMAG, et reposant sur les interfaces de programmation de l'environnement OPEN/CÆSAR [9], TGV a été utilisé avec succès pour plusieurs études de cas.

Dans le cadre du GIE DYADE (voir § 6.1), nous utilisons TGV depuis plusieurs années pour produire des tests à partir de descriptions en LOTOS d'architectures multiprocesseurs développées par BULL.

De plus, le contrat REUTEL-2000 auquel nous participons (voir § 6.3) prévoit le développement d'une version de TGV adaptée à la notation UML.

C'est pourquoi, d'un commun accord avec les auteurs de TGV, nous avons décidé en 1998 d'intégrer et de diffuser, au sein de la boîte à outils CADP, une version "publique" de TGV permettant la génération de tests à la volée et capable de fonctionner pour divers langages (LOTOS, UML, etc.).

Nous avons eu de nombreux échanges techniques à ce sujet avec les membres du projet PAMPA qui développent les versions récentes de TGV (Thierry Jéron, Pierre Morel et Séverine Simon). En 1999, ceux-ci ont apporté d'importantes améliorations à leur logiciel. C'est ainsi que la toute dernière version de TGV intègre plusieurs de nos suggestions, notamment :

- Les possibilités de masquage et de renommage d'actions ont été enrichies par l'ajout d'expressions régulières, à l'aide des bibliothèques CAESAR_HIDE et CAESAR_RENAME que nous avions développées en 1998.

- La distinction entre entrées et sorties, précédemment effectuée selon un critère syntaxique rudimentaire sur le nom des actions, est désormais paramétrable grâce aux expressions régulières de la bibliothèque CAESAR_HIDE.

- Les objectifs de test qui, dans les versions antérieures, devaient être fournis à TGV dans un format non standard, sont désormais des automates "ordinaires" (encodés soit dans le format textuel ALDÉBARAN, soit dans le format binaire BCG), ce qui les rend compatibles avec les autres outils de CADP.

- L'expressivité des objectifs de test a été accrue grâce à l'emploi des expressions régulières de la bibliothèque CAESAR_HIDE, qui permettent de caractériser un ensemble d'actions.
- Les cas de test produits par TGV peuvent désormais être encodés dans le format BCG.
- Les algorithmes de génération de tests implémentés dans la nouvelle version de TGV traitent divers problèmes que nous avions signalés en 1998 : la prise en compte des boucles permet de générer moins de verdicts "inconclusifs" (un problème pour lequel nous avions développé l'outil BCG_LOOP en 1998) ; la production de tests non déterministes — que l'on transforme ensuite en tests déterministes exécutables grâce à l'outil BCG_SPLIT développé par VASY en 1998 — permet de produire à moindre effort des tests plus généraux.

Enfin, nous avons fourni au projet PAMPA nos outils de développement multi-plateformes — ce qui a permis le portage de TGV vers les systèmes d'exploitation SUNOS et LINUX — et contribué à la mise en forme de la documentation en ligne de TGV.

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 : *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. En 1999, nous avons donc maintenu nos compilateurs LOTOS et amélioré leurs performances. Nous avons également poursuivi le développement du compilateur TRAIAN pour le langage LOTOS NT, ce qui ouvre de nouvelles perspectives concernant la génération de compilateurs.*

5.2.1 Compilation des types du langage LOTOS

Participants : Hubert Garavel, Radu Mateescu.

Le code C généré par le compilateur CÆSAR.ADT pour les types de données complexes (tuples, listes, arbres, ensembles, etc.) a été amélioré de manière significative. Auparavant, tous ces types étaient implémentés comme pointeurs vers des structures ou des unions avec discriminants, ce qui entraînait un surcoût en mémoire de 4 octets par valeur mémorisée.

La nouvelle version de CÆSAR.ADT implémente tous les types LOTOS non-récursifs directement comme des structures ou des unions avec discriminants, ce qui permet d'économiser l'espace mémoire précédemment consommé par les pointeurs. Dans le cas des types LOTOS directement ou mutuellement récursifs (listes, arbres, etc.), la nouvelle version de CÆSAR.ADT met en œuvre des heuristiques visant à implémenter comme pointeurs un nombre minimal de types, de façon à "casser" toutes les dépendances cycliques entre les types. L'utilisateur

a la possibilité d'imposer l'implémentation des types en imposant que certains types soient implémentés avec ou sans pointeurs.

Ces améliorations, qui ont nécessité des changements importants dans le compilateur CÆSAR.ADT (ajout de contrôles sémantiques et algorithmes d'ordonnancement des définitions de types en langage C), ont permis d'obtenir des réductions parfois spectaculaires de l'espace mémoire pour les types de données couramment utilisés dans les protocoles de communication (structures, paquets, unités de transfert de données, etc.). Le gain est très tangible en vérification énumérative, où il est souvent nécessaire de mémoriser simultanément un nombre important de valeurs (plusieurs millions) représentant le contenu de tous les états du système. Ainsi, dans le cas de la spécification du bus SCSI-2 étudiée dans l'action de recherche coopérative VERDON, la consommation mémoire a chuté de 955 à 7 Méga-octets seulement.

5.2.2 Compilation des processus du langage LOTOS

Participant : Hubert Garavel.

En 1999, nous avons travaillé à généraliser le modèle réseau utilisé comme forme intermédiaire par le compilateur CÆSAR pour traduire les définitions de processus présentes dans une description LOTOS. La définition formelle de ce modèle réseau est donnée dans [7, 4] : il s'agit d'un réseau de Petri étendu par des variables d'état typées, dans lequel chaque transition est étiquetée par divers attributs construits à partir des informations présentes dans la description LOTOS. Ces attributs comprennent notamment une *porte* correspondant à un point de communication, une *offre* qui est une liste d'expressions typées correspondant aux valeurs émises ou reçues sur la porte, et une *action* qui est un bloc d'instructions séquentielles pouvant comporter des *affectations* (qui modifient la valeur d'une variable d'état), des *itérations* (qui font prendre à une variable toutes les valeurs possibles dans un domaine fini) et des *conditions* (gardes booléennes pouvant empêcher le franchissement de la transition). En 1992, afin de réduire le nombre d'états des graphes produits en vérification exhaustive, ce modèle avait été enrichi en attachant à chaque transition une liste de variables devant être réinitialisées (par défaut, à zéro) lorsque la transition est franchie.

Dans le modèle réseau de CÆSAR, l'action d'une transition est exécutée *avant* que les expressions de l'offre de la transition ne soient évaluées. Afin d'obtenir des réseaux plus compacts, il est apparu souhaitable que certaines actions (affectations ou réinitialisations de variables) puissent aussi être effectuées *après* l'évaluation de l'offre. Nous avons introduit cette extension dans le modèle réseau de CÆSAR et généralisé en conséquence les diverses optimisations opérant sur ce modèle réseau.

Les premiers résultats de ce travail sont encourageants :

- Sur de nombreux exemples LOTOS, ces modifications ont permis de réduire le nombre de places, de transitions et de variables des réseaux produits par CÆSAR. Dans de nombreux cas, ces réductions sont significatives. Ainsi, sur un exemple fourni par Elie Najm (Ecole Nationale Supérieure des Télécommunications), le nombre de places passe de 17 à 4, le nombre de transitions passe de 60 à 47 et le nombre de variables passe de 16 à 1.

- La réduction de la taille des réseaux entraîne fréquemment une réduction de la taille

des graphes correspondants. Souvent, le facteur de réduction est d'un ordre de grandeur, mais il peut être bien supérieur dans certains cas, comme sur l'exemple d'Elie Najm pour lequel la taille du graphe passe de 304 305 états et 1 434 070 transitions à 83 états et 364 transitions.

– La vitesse de génération des graphes a aussi été augmentée : sur des exemples LOTOS étudiés par Fabrice Baray (laboratoire ISIMA/LIMOS), la nouvelle version de CÆSAR est, en moyenne, 150 fois plus rapide que la précédente (la taille des graphes produits restant sensiblement la même).

Toutefois, l'amélioration n'est pas systématique car, sur quelques exemples, l'un des paramètres du réseau (par exemple, le nombre de variables) peut augmenter alors que les autres paramètres (par exemple, le nombre de places et de transitions) diminuent ; sur d'autres exemples, le nombre d'états du graphe peut diminuer tandis que le nombre de transitions augmente. L'analyse de cette situation reste à approfondir.

5.2.3 Compilation des types du langage LOTOS NT

Participants : Claude Chaudet, Hubert Garavel, Marc Herbert, Mihaela Sighireanu.

Cette année, les travaux relatifs au langage LOTOS NT et au compilateur TRAIAN (voir § 4.2) ont donné lieu à la soutenance de thèse de M. Sighireanu [13] en janvier 1999.

Ces travaux se sont poursuivis par l'utilisation du compilateur TRAIAN sur différentes études de cas :

– Modélisation en LOTOS NT du protocole SCSI-2 que nous avons étudié dans le cadre de l'action de recherche coopérative VERDON (voir § 7.1.1) ;

– Modélisation en LOTOS NT d'un *benchmark* — transmis par Jan-Friso Groote (CWI, Amsterdam) — servant à tester les performances de diverses implémentations de langages fonctionnels ;

– Modélisation en LOTOS NT d'un protocole de reconfiguration dynamique d'agents distribués [19] ;

– Utilisation de LOTOS NT pour la génération de compilateurs (voir § 5.2.4).

Nous pouvons également mentionner l'utilisation de TRAIAN et de CADP pour la conception conjointe matériel-logiciel [WB99] par Pierre Wodey et Fabrice Baray (laboratoire ISIMA/LIMOS, Clermont-Ferrand).

Ces études de cas, conjuguées avec la création d'une base de tests extensive pour TRAIAN, nous ont permis de découvrir et de corriger 44 bogues. Nous avons également apporté des améliorations importantes au compilateur, notamment :

– la refonte de l'algorithme d'implémentation des types généraux,

[WB99] P. WODEY, F. BARAY, «Linking Codesign and Verification by means of E-LOTOS FDT», *in : Proceedings of the Euromicro Workshop on Digital System Design: Architectures, Methods and Tools (Milano, Italy)*, L. Józwiak (éditeur), IEEE, septembre 1999.

- l'écriture d'une bibliothèque complète d'opérateurs usuels pour les types prédéfinis,
- la génération de code pour le mécanisme d'exceptions utilisé dans la partie données de LOTOS NT, et
- le portage de TRAIAN vers les systèmes d'exploitation LINUX et WINDOWS.

Ainsi, la dernière version de TRAIAN — qui sera distribuée au début de l'an 2000 — compile la quasi-totalité de la partie données du langage LOTOS NT. Par ailleurs, nous avons étudié d'autres évolutions de TRAIAN [20] :

- la suppression des pointeurs dans l'implémentation des types LOTOS NT non récursifs,
- l'optimisation de l'implémentation des types de données particuliers (types singletons, types énumérés "en cascade", entiers modulaires), et
- la factorisation des sous-termes identiques au moyen de tables de hachage, afin de réduire la quantité de mémoire allouée.

Ces évolutions, si elles ne sont pas encore intégrées dans la version officielle de TRAIAN, ont fait l'objet d'une implémentation prototype et, pour certaines, ont été intégrées au compilateur CÆSAR.ADT (voir § 5.2.1).

5.2.4 Outils de génération de compilateurs

Participants : Claude Chaudet, Christophe Discours, Hubert Garavel, Marc Herbert, Mihaela Sighireanu.

Le développement de compilateurs tient une place importante dans les travaux de l'équipe VASY. La boîte à outils CADP comporte plusieurs compilateurs qui, pour la partie analyse lexicale et syntaxique, sont tous réalisés à l'aide du système SYNTAX développé à l'INRIA Rocquencourt (notamment par Pierre Boullier et Philippe Deschamp). En revanche, ces compilateurs diffèrent en ce qui concerne la technologie utilisée pour décrire, parcourir et transformer les arbres de syntaxe abstraite. Selon cette technologie, on peut distinguer trois générations successives :

- Dans la première génération (compilateur CÆSAR), ces arbres étaient décrits en langage C, ce qui a entraîné des problèmes de mise au point liés à la manipulation de pointeurs et à l'allocation dynamique de mémoire.
- Dans la seconde génération (compilateurs CÆSAR.ADT et XTL), une solution mixte combinant les langages C et LOTOS a été utilisée : les arbres étaient décrits en types abstraits algébriques de LOTOS (lesquels étaient ensuite traduits en C par le compilateur CÆSAR.ADT) tandis que certains traitements à caractère impératif étaient programmés directement en C. Cette approche réduit les inconvénients liés à l'utilisation exclusive du langage C, mais se heurte à certaines limitations des types abstraits algébriques (absence de composition séquentielle, de variables locales, etc.).

– Dans la troisième génération (compilateurs TRAIAN, SVL et MCL_EXPAND pour EVALUATOR 3.0), le système de génération de compilateurs FNC-2 développé à l'INRIA Rocquencourt (notamment par Martin Jourdan et Didier Parigot) a été utilisé.

En tant qu'utilisateurs de FNC-2, nous avons collaboré avec Didier Parigot afin d'améliorer ce système. En 1999, notre contribution a porté sur la préparation, le test et l'amélioration de la version 1.18 de FNC2 :

– Nous avons conçu une version étendue des scripts FNC2 et PREFNC2 pour permettre leur utilisation en mode non-interactif tout en conservant la possibilité du mode interactif qui existait précédemment. Un manuel d'utilisation du script FNC2 a été rédigé et intégré à la version 1.18 de FNC2.

– Nous avons apporté les changements nécessaires pour que FNC2 ne soit plus dépendant des outils GNU (ces changements portent sur la détermination de l'architecture de la machine hôte et le calcul des dépendances entre fichiers). En particulier, ceci permet d'utiliser FNC2 sur une machine fonctionnant sous le système d'exploitation SOLARIS sans qu'il soit besoin d'installer préalablement les outils GCC et GMAKE de GNU.

– Nous avons réussi à contourner une erreur qui nous empêchait d'utiliser FNC2 sous le système d'exploitation SUNOS 4.1 et, par conséquent, de construire des versions de nos compilateurs pour ce système d'exploitation.

– Nous avons adapté FNC-2 pour qu'il puisse produire, par compilation croisée, des programmes exécutables sous le système d'exploitation WINDOWS de MICROSOFT.

– Nous avons mis à jour le forum aux questions (FAQ) que nous avions créé en 1998 afin de documenter les problèmes les plus couramment rencontrés dans l'utilisation du système FNC2 (voir `http://www.inrialpes.fr/vasy/fnc2`).

Néanmoins, en dépit de la participation active de l'équipe VASY dans l'amélioration de FNC2 et en dépit de l'expérience acquise avec ce système depuis 1996, nos difficultés d'utilisation n'ont cessé de s'amplifier :

– Si la version 1.18 de FNC2 corrigeait de nombreuses erreurs signalées par VASY les années précédentes, elle en introduisait de nouvelles. Ainsi, en 1999, nous avons découvert 9 nouvelles erreurs. Du fait de la complexité interne du système FNC2, la mise en évidence de ces erreurs et leur contournement a souvent été lente et difficile (nécessitant parfois plusieurs jours de travail).

– Dans le même temps, suite à l'arrêt de l'avant-projet OSCAR, le travail de développement de FNC2 a pratiquement cessé en 1999. Faute de maintenance, les problèmes qui étaient excusables pour un prototype de recherche susceptible d'améliorations sont devenus rédhibitoires pour un logiciel figé.

Dans ces conditions, nous avons dû nous résoudre, vers le milieu de l'année 1999, à abandonner l'utilisation du système FNC2 pour nos futurs développements et à rechercher une technologie alternative pour la construction de nos compilateurs.

En l'absence de solution bien établie pour ce problème, nous avons choisi d'expérimenter, en remplacement de FNC2, l'utilisation du langage LOTOS NT pour la description et la manipulation des arbres de syntaxe abstraite, les traitements sémantiques et la génération de code. Il s'agit de concevoir une quatrième génération de compilateurs, construits en utilisant le système SYNTAX, ainsi que les langages LOTOS NT et C. Dans cette approche, le compilateur TRAIAN est utilisé pour traduire en C les programmes LOTOS NT.

Les deux approches diffèrent fortement quant à la programmation des traitements sémantiques :

- Le système FNC2 permet une approche déclarative : l'utilisateur définit les règles de calcul des attributs de manière locale à chaque nœud de l'arbre de syntaxe abstraite ; FNC2 calcule les dépendances entre attributs et en déduit automatiquement un ordonnancement pour l'évaluation des attributs.

- Au contraire, l'utilisation d'un langage fonctionnel/impératif tel que LOTOS NT requiert que l'utilisateur spécifie lui-même la séquence des traitements. En pratique, ceci ne semble guère être une contrainte, puisque l'utilisateur connaît en général l'ordre dans lequel les attributs seront évalués. En outre, le style impératif semble plus facile à relire, alors que le style déclaratif tend à obscurcir le flot des données.

Une première expérimentation a porté sur l'écriture d'un compilateur pour le langage SIMPROC, un petit langage algorithmique servant d'exemple pour l'utilisation du système FNC2 et distribué avec ce dernier. En utilisant SYNTAX, LOTOS NT et TRAIAN, nous avons réalisé un compilateur SIMPROC comportant un analyseur lexical et syntaxique, une table des symboles, un vérificateur de typage et un générateur de code objet.

Ce travail a permis de tester de nouveaux pans du langage LOTOS NT et d'améliorer le compilateur TRAIAN en corrigeant diverses erreurs et en complétant certaines parties manquantes.

Les conclusions de cette expérimentation sont encourageantes :

- L'utilisation de LOTOS NT permet une définition plus simple et plus concise du compilateur SIMPROC : 1280 lignes (dont 800 lignes de LOTOS NT) réparties en 7 fichiers, contre 1550 lignes réparties en 18 fichiers avec FNC2. La compacité du code LOTOS NT s'explique de plusieurs manières :

  - En FNC2, chaque étape sémantique du compilateur nécessite la définition et la construction d'un nouvel arbre — il n'est pas possible d'enrichir un arbre existant en lui ajoutant de nouveaux attributs — ce qui conduit à des recopies inutiles d'arbres, ainsi qu'à une pléthore d'identificateurs et de fichiers. De plus, cette redondance fait que toute modification, même mineure, entraîne des changements à de multiples endroits dans de multiples fichiers.

  - LOTOS NT possède plusieurs constructions fort utiles qui sont absentes de FNC2 : boucles permettant d'itérer sur tous les éléments d'une liste, fonctions renvoyant plusieurs résultats, possibilité de factoriser les traitements sémantiques communs à plusieurs règles, etc.

- La mise au point et la maintenance s'avèrent plus faciles avec LOTOS NT, à cause des contrôles stricts exercés par TRAIAN lors de la compilation de code LOTOS NT et de la possibilité offerte par TRAIAN d'imprimer automatiquement les structures de données complexes (notamment les arbres de syntaxe abstraite).

  Au contraire avec FNC2, la mise au point des calculs d'attributs doit se faire sur le code C généré par FNC2, ce qui est d'autant plus malaisé que ce code C est dispersé en de nombreux fichiers — au total, FNC2 ne crée pas moins de 120 fichiers pour construire le compilateur SIMPROC — et que FNC2 ne fournit aucun outil de visualisation des arbres de syntaxe abstraite.

- Du point de vue des performances, le temps de construction du compilateur SIMPROC, la taille du code binaire du compilateur obtenu et sa vitesse d'exécution sont très proches dans les deux approches.

  En revanche, la mesure des empreintes mémoire lors de la compilation d'un petit programme SIMPROC révèle que le compilateur construit avec FNC2 consomme 2,6 fois plus de mémoire que celui construit avec TRAIAN, et qu'en outre il donne lieu à des fuites de mémoire (mémoire allouée et non référencée).

Compte-tenu de ses perspectives prometteuses, l'expérience conduite avec le langage SIMPROC méritait d'être prolongée. C'est pourquoi nous avons entrepris, fin 1999, de réécrire en LOTOS NT le compilateur SVL actuellement développé avec FNC2.

5.2.5 Portage de logiciels vers le système d'exploitation Windows

Participants : Hubert Garavel, Marc Herbert, Aldo Mazzilli.

En 1999, nous avons travaillé sur le portage des outils CADP vers les systèmes d'exploitation WINDOWS 98 et WINDOWS NT de MICROSOFT. Nos résultats sont les suivants [21] :

- Pour le portage des programmes C, nous avons étendu nos outils de développement multi-plateformes afin d'y intégrer le compilateur croisé EGCS et la bibliothèque MINGW32 qui permettent la production de programmes exécutables WINDOWS depuis un environnement UNIX.

- Pour le portage des interfaces graphiques, nous avons dû modifier tous les programmes TCL/TK de CADP et effectuer une migration vers la version 8.2 de TCL/TK. Le portage de l'interface graphique EUCALYPTUS nous a permis de corriger des erreurs et d'apporter diverses améliorations. Le portage de l'assistant d'installation de CADP, INSTALLATOR, a nécessité des changements profonds puisque la nouvelle version d'INSTALLATOR n'utilise plus le logiciel EXPECTK (non disponible sous WINDOWS) mais l'interprète WISH

de TCL/TK avec un client FTP embarqué ; cette nouvelle version possède de meilleurs diagnostics d'erreurs et intègre de nouvelles fonctionnalités réclamées par les utilisateurs.

- Nous avons modifié tous les *shells-scripts* de CADP afin d'en éliminer les dépendances spécifiques à UNIX.

- Nous avons réécrit toute la documentation en ligne de CADP, initialement rédigée dans le format NROFF utilisé par la commande MAN d'UNIX, format qui n'existe pas sous WINDOWS. Ce travail a été complété par la mise en place d'une passerelle basée sur l'outil POLYGLOTMAN — ainsi qu'un ensemble de scripts et programmes C et LEX que nous avons développés — permettant une traduction automatique des pages NROFF vers les formats HTML (affichable sous WINDOWS) et POSTSCRIPT.

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 1999, nous avons traité plusieurs études de cas dans des domaines très divers, notamment dans le cadre de notre coopération avec BULL.*

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

Participant : Massimo Zendri.

En collaboration avec le projet PAMPA (Rennes), nous travaillons depuis octobre 1998 sur FAME, une nouvelle architecture multi-processeurs CC-NUMA développée au centre BULL des Clayes-sous-Bois (France) et basée sur des processeurs INTEL de nouvelle génération. Cette collaboration a été officialisée en 1999 par la création de l'action FORMALFAME du GIE BULL-INRIA DYADE (voir 6.1).

L'action FORMALFAME cherche à promouvoir l'utilisation de méthodes formelles pour la vérification et le test d'architectures multiprocesseurs. La méthodologie utilisée consiste à établir un modèle de référence de l'architecture visée en utilisant le langage de description formelle LOTOS. Les outils de simulation et de vérification CADP sont ensuite utilisés pour s'assurer de la correction du modèle de référence. Finalement, l'outil TGV est employé pour produire des jeux de tests "abstraits" déduits du modèle de référence ; ces tests abstraits sont ensuite traduits automatiquement en tests exécutables qui, appliqués dans l'environnement de test utilisé par BULL, serviront à valider l'implémentation du produit final.

En 1999, les travaux de FORMALFAME ont porté sur les points suivants :

- En janvier 1999, nous avons achevé les travaux entrepris en 1998 pour traduire en LOTOS 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. Bien que nous ayions montré la faisabilité d'une traduction systématique de MURPHI vers LOTOS, nous n'avons pas poursuivi cette approche car, à l'usage, la description de FAME en MURPHI s'est révélée difficile à faire évoluer et insuffisamment détaillée pour permettre la génération de tests pertinents.

- De février à septembre 1999, nous avons concentré nos efforts sur le circuit CCS (*Core Chip Set*) de FAME qui, pour un groupe de quatre processeurs, gère les communications (arbitrage de bus, gestion mémoire, *crossbar*, etc.) et assure la cohérence de caches. L'objectif de notre travail était le test fonctionnel de l'implantation du composant CCS — c'est-à-dire de son modèle VHDL au niveau RTL (*Register Transfer Level*) — en ciblant notamment les aspects relatifs à la correction du protocole de cohérence de caches.

  Nous avons élaboré une description LOTOS (1 200 lignes, 10 processus) constituant un modèle de référence du circuit CCS et de son environnement. Pour décrire cet environnement, nous avons modélisé le comportement des deux bus, du circuit réseau et de deux microprocesseurs INTEL accédant au bus.

  En appliquant l'outil TGV à cette description LOTOS, nous avons pu produire diverses suites de tests : 21 tests de base (temps de génération : 1 mn/test), 50 tests de collision (15 mn/test) et une suite de tests avec objectif de test généralisé (1 journée). En collaboration avec Solofo Ramangalahy (projet PAMPA), un traducteur écrit en LEX/YACC a été développé pour convertir les tests "abstraits" produits par TGV en tests directement exécutables dans l'environnement de test utilisé par BULL (c'est-à-dire en programmes C dans lesquels les émissions/réceptions de signaux sont effectuées par des appels à des fonctions définies dans des interfaces de programmation spécialisées).

- A partir d'octobre 1999, nous nous sommes intéressés au circuit NCS (*Network Chip Set*) de FAME qui assure les communications réseau. Nous avons élaboré une description LOTOS (1 200 lignes, 16 processus) modélisant trois circuits NCS placés dans un environnement comportant quatre bus et trois processeurs INTEL. A partir de cette description LOTOS, l'outil TGV a produit 50 tests de base (temps de génération : 30 secondes/test). Enfin, le traducteur de tests abstraits en tests exécutables développé pour le circuit CCS a été étendu pour permettre le test du modèle RTL du circuit NCS.

5.3.2 Applications embarquées sur cartes à puces

Participants : Hubert Garavel, Massimo Zendri.

En 1999, nous avons effectué une étude de faisabilité avec le centre de recherche et développement de BULL SC&T (*Smart Cards and Terminals*) situé à Louveciennes afin d'expérimenter l'utilisation des outils CADP et TGV pour la vérification et le test des cartes à puce.

En collaboration avec Sébastien Gelgon (BULL SC&T), un petit système d'exploitation embarqué sur une carte à puce a été choisi comme exemple. Ce système — qui assure l'interface entre la carte à puce au sens physique et les applications embarquées sur la carte — comporte un système de gestion de fichiers et de répertoires, avec des primitives de lecture-écriture sur fichiers, d'authentification, de remise à zéro de la carte, etc. Il s'agit d'un exemple significatif comprenant toutes les fonctionnalités nécessaires au fonctionnement de plusieurs applications embarquées (carte bancaire, porte-monnaie électronique, GSM, etc.).

Pour permettre, d'une part, l'utilisation des outils CADP et TGV et, d'autre part, la réutilisation de code C déjà existant à BULL SC&T, nous avons modélisé ce système d'exploitation à l'aide d'une technique originale consistant à combiner les langages C et LOTOS :

- La partie de code LOTOS (130 lignes) spécifie l'interface du système d'exploitation avec son environnement (c'est-à-dire les fonctions primitives offertes par le système et les résultats qu'elles renvoient) et le comportement global du système sous la forme d'un automate à commandes gardées.

- La partie de code C (1 700 lignes) décrit les types de données, les variables d'état, les conditions sous lesquelles les primitives système peuvent être exécutées et les modifications qu'elles apportent à l'état du système.

L'utilisation des outils CADP a permis d'analyser plusieurs propriétés de bon fonctionnement, certaines simples (par exemple, l'absence de blocage), d'autres plus complexes (par exemple, le fait qu'une carte ne peut pas s'authentifier sans tirer au moins un nombre aléatoire). L'outil TGV a été employé pour produire automatiquement une dizaine de tests de séquencement.

5.3.3 Autres études de cas

Participants : Manuel Aguilar Cornejo, Hubert Garavel, Radu Mateescu, Charles Pecheur, 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'une publication [18] ;

- Nous avons formalisé en logique temporelle et vérifié les propriétés de bon fonctionnement du protocole SCSI-2 qui sert d'exemple commun pour les équipes participant à l'action de recherche coopérative VERDON (voir § 7.1.1) ;

- Nous avons spécifié en LOTOS et LOTOS NT un protocole de reconfiguration dynamique d'agents développé au sein du projet SIRAC. Nous avons ensuite identifié et vérifié les propriétés essentielles au bon fonctionnement du protocole [19].

D'autres équipes ont également utilisé la boîte à outils CADP pour diverses études de cas. Pour ne citer que les travaux publiés, on peut mentionner :

- la vérification et l'optimisation du protocole TCAP pour la communication téléphonique [AvL99] ;
- la modélisation et la validation du protocole de transport haut débit XTP [BA98] ;
- la vérification de circuits séquentiels [HT99a] ;
- 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 vérification du protocole CHAP pour l'authentification des connexions réseau [Led99] ;
- la vérification du protocole EQUICRYPT pour la sécurité de l'accès aux services multimédia [LBL+99] ;
- la vérification du protocole INRES de transfert de données [LA98] ;
- la spécification et la vérification d'un système distribué de sécurité des télécommunications destiné à lutter contre les téléphones mobiles clonés [NdSCRW99] ;
- la modélisation et la vérification du protocole audio-vidéo HAVi de PHILIPS [Rom99] ;

[AvL99] T. ARTS, I. VAN LANGEVELDE, « How μCRL Supported a Smart Redesign of a Real-life Protocol», *in : Proceedings of the 4th International ERCIM Workshop on Formal Methods for Industrial Critical Systems (Trento, Italy)*, S. Gnesi, D. Latella (éditeurs), ERCIM, CNR, p. 31–53, juillet 1999. Also available as CWI Technical Report SEN-R9910.

[BA98] A. BENSLIMANE, A. ABOUAISSA, « XTP Specification and Validation with LOTOS », *in : Proceedings of the Western MultiConference WMC'98, Communication Networks and Distributed Systems Modeling and Simulation CNDS'98 (San Diego, California, USA)*, Society for Computer Simulation International, janvier 1998.

[HT99a] J. HE, K. J. TURNER, «Protocol-Inspired Hardware Testing», *in : Proceedings of the IFIP 12th International Workshop on Testing of Communicating Systems IWTCS'99 (Budapest, Hungary)*, G. Csopaki, S. Dibuz, K. Tarnay (éditeurs), Kluwer Academic, p. 131–147, septembre 1999.

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

[Led99] G. LEDUC, « Verification of two versions of the Challenge Handshake Authentication Protocol (CHAP) », *Annals of Telecommunications*, 1999, To appear.

[LBL+99] G. LEDUC, O. BONAVENTURE, L. LÉONARD, E. KOERNER, C. PECHEUR, « Model-Based Verification of a Security Protocol for the Conditional Access to Services», *Formal Methods in System Design 14*, 2, mars 1999, p. 171–191.

[LA98] M. LUUKKAINEN, A. AHTIAINEN, « Compositional Verification of large SDL systems », *in : Proceedings of the 1st Workshop of the SDL Forum Society on SDL and MSC SAM'98 (Berlin, Germany)*, juin 1998.

[NdSCRW99] M. S. M. A. NOTARE, F. A. DA SILVA CRUZ, B. G. RISO, C. B. WESTPHALL, «Wireless Communications: Security Management Against Cloned Cellular Phones », *in : Proceedings of the IEEE Wireless Communications and Networking Conference WCNC'99 (New Orleans, LA, USA)*, IEEE, p. 1412–1416, septembre 1999.

[Rom99] J. ROMIJN, *Analysing Industrial Protocols with Formal Methods*, thèse de doctorat, University of Twente, The Netherlands, septembre 1999.

- la validation de l'interface multi-utilisateur d'un système de contrôle du traffic aérien [SJ99] ;

- l'utilisation de LOTOS NT et de CADP pour la conception conjointe matériel-logiciel [WB99].

Par ailleurs, il convient de noter que d'autres équipes de recherche ont aussi adopté les composants logiciels offerts par les environnements BCG et OPEN/CÆSAR pour développer leurs propres outils. Outre le cas déjà mentionné de l'outil TGV (voir § 5.1.6), nous pouvons citer les réalisations suivantes :

- l'outil de génération de tests TORX [BFdV+99] développé à l'Université de Twente (Pays-Bas) ;

- deux environnements permettant la simulation et la vérification de programmes BDL [CJ99] et UML développés dans le projet PAMPA (voir § 6.3) ;

- un outil de génération de tests pour les circuits séquentiels [HT99b] développé à l'Université de Stirling (Ecosse) ;

- un outil de génération de tests de protocoles exploitant les propriétés des symétries [Rom99] développé au CWI (Pays-Bas) ;

- l'outil KRONOSOPEN pour la vérification à la volée de systèmes temporisés [Tri99] développé par le laboratoire VERIMAG (Grenoble).

[SJ99] M. SAGE, C. JOHNSON, « A Declarative Prototyping Environment for the Development of Multi-User Safety-Critical Systems », *in : Proceedings of the 17th International System Safety Conference ISSC'99 (Orlando, Florida, USA)*, System Safety Society, août 1999.

[WB99] P. WODEY, F. BARAY, « Linking Codesign and Verification by means of E-LOTOS FDT », *in : Proceedings of the Euromicro Workshop on Digital System Design: Architectures, Methods and Tools (Milano, Italy)*, L. Józwiak (éditeur), IEEE, septembre 1999.

[BFdV+99] A. BELINFANTE, J. FEENSTRA, R. G. DE VRIES, J. TRETMANS, N. GOGA, L. FEIJS, S. MAUW, L. HEERINK, « Formal Test Automation: A Simple Experiment », *in : Proceedings of the IFIP 12th International Workshop on Testing of Communicating Systems IWTCS'99 (Budapest, Hungary)*, G. Csopaki, S. Dibuz, K. Tarnay (éditeurs), Kluwer Academic, septembre 1999.

[CJ99] H. CANON, C. JARD, « Un modèle sémantique pour la validation des logiciels objets en télécommunication », *in : Actes du Colloque Francophone pour l'Ingénierie des Protocoles CFIP'99 (Nancy, France)*, A. Schaff, F. Lepage, J.-P. Thomesse (éditeurs), Hermès, p. 83–98, Paris, avril 1999.

[HT99b] J. HE, K. J. TURNER, « Specification and Verification of Synchronous Hardware using LOTOS », *in : Proceedings of the IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing, and Verification FORTE XII / PSTV XIX'99 (Beijing, China)*, J. Wu, S. T. Chanson, Q. Gao (éditeurs), IFIP, Kluwer Academic Publishers, p. 295–312, octobre 1999.

[Tri99] S. TRIPAKIS, « Extended Kronos/CADP tool: minimization, on-the-fly verification and compositionality », *Technical Report n° T226*, VERIMAG, Grenoble, France, avril 1999.

6 Contrats industriels (nationaux, européens et internationaux)

6.1 Action FormalFame (Dyade)

Participants : Hubert Garavel, Marc Herbert, Radu Mateescu, 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.

Depuis 1996, nous entretenons une collaboration de longue durée avec BULL dans le cadre du GIE BULL-INRIA DYADE. Cette collaboration, à laquelle participe également le projet PAMPA (Rennes), est coordonnée par un ingénieur BULL (M. Zendri) installé dans les locaux de l'Unité de Recherche INRIA Rhône-Alpes. Elle vise à démontrer que les méthodes formelles et les outils développés à l'INRIA pour la validation et le test des protocoles de télécommunications peuvent aussi être appliqués avec succès aux architectures multi-processeurs développées par BULL. L'objectif à long terme est d'offrir une chaîne complète et intégrée d'outils pour la spécification formelle, la simulation, le prototypage rapide, la vérification, la génération de tests et leur exécution.

Une première étape de cette collaboration a pris fin en 1998 avec l'achèvement de l'action VASY de DYADE, consacrée à deux études de cas successives : le protocole d'arbitrage de bus de l'architecture POWERSCALE [1] et le protocole de cohérence de caches de l'architecture multiprocesseurs POLYKID. Le résultat de ces expérimentations a été jugé positif : la faisabilité de l'approche proposée a été démontrée et BULL a manifesté son intérêt à poursuivre l'application de cette approche sur de nouvelles architectures.

Depuis octobre 1998, nos efforts sont consacrés à FAME, une architecture multi-processeurs CC-NUMA en cours de développement au centre BULL des Clayes-sous-Bois (France). D'abord informelle, cette collaboration a été officialisée en 1999 par le comité de pilotage et le comité de direction de DYADE sous la forme d'une nouvelle action intitulée FORMALFAME.

En 1999, les principaux résultats de FORMALFAME sont les suivants :

- modélisation formelle et génération de tests pour l'architecture FAME (voir § 5.3.1),
- amélioration des performances des compilateurs CÆSAR et CÆSAR.ADT (voir § 5.2.1 et § 5.2.2),
- intégration de l'outil TGV dans la boîte à outils CADP (voir § 4.1).

6.2 Action SmartTools (Dyade)

Participants : Claude Chaudet, Hubert Garavel, Mihaela Sighireanu.

Mots clés : compilateur, compilation, environnement de programmation, génération de code, génie logiciel.

Depuis 1999, nous participons à l'action SmartTools du Gie Bull-Inria Dyade. Dirigée par Isabelle Attali (avant-projet Oasis), cette action vise à construire un environnement de développement convivial et interactif pour JavaCard (architecture de programmation standard pour cartes à puces). Dans ce contexte, nous sommes particulièrement intéressés par la disponibilité annoncée d'outils génériques permettant de réaliser des environnements de programmation complets (éditeurs structurés, vérificateurs syntaxiques et sémantiques, interprètes, traducteurs, outils d'aide à la mise au point, etc.) pour différents langages informatiques.

6.3 Contrat Reutel-2000 (Alcatel)

Participants : Moëz Cherif, Hubert Garavel.

Mots clés : activité de conception, algorithme réparti, application répartie, compilation, concurrence, génération de code, génération de test, génie logiciel, ingénierie des protocoles, modélisation, langage à objets, langage d'interface, parallélisme asynchrone, parallélisme synchrone, protocole de communication, simulation, spécification formelle, synchronisation, système distribué, validation, vérification de programme.

Depuis avril 1999, nous participons à la deuxième phase du contrat Reutel-2000, un contrat de recherche entre Alcatel et l'Inria s'inscrivant dans le contexte de l'accord-cadre de collaboration Alcatel/Inria. Les équipes Adp, Compose, Ep-Atr et Pampa de l'Inria Rennes sont également engagées dans le contrat Reutel-2000, dont Claude Jard est le coordonnateur à l'Inria.

L'objectif de cette collaboration est la maîtrise du développement logiciel d'applications de télécommunications réutilisables, par la conception d'outils de manipulation formelle à l'intérieur d'une chaîne de développement définie par Alcatel, en combinant les approches à objets et les modèles de parallélisme synchrone et asynchrone.

L'industrie des télécommunications est soumise à de fortes contraintes visant à réduire les coûts et les délais de développement tout en améliorant la qualité du logiciel. Compte-tenu de la taille et de la complexité des applications mises en œuvre, les spécifications et les programmes doivent être conçus de manière suffisamment générique pour pouvoir fonctionner dans des configurations hétérogènes, ainsi que pour assurer une flexibilité et une réactivité élevées en regard des évolutions du marché et des technologies naissantes.

Le cadre de développement considéré par Alcatel prend en compte la norme Corba (*Common Object Request Broker Architecture*). Les spécifications sont élaborées selon une méthodologie de conception à objets compatible avec la notation Uml (*Unified Meta Language*). Le développement repose sur l'écriture de schémas de programmes dans des langages d'interface inspirés d'Idl (*Interface Definition Language*) mais étendus pour faire apparaître des informations comportementales. L'utilisation de tels langages d'interface offre une certaine indépendance vis-à-vis des différents langages utilisés pour la programmation des objets logiciels et des différentes plate-formes d'exécution. Dans cette approche, la mise en œuvre d'une application sur une plate-forme donnée se fait, lorsque cela est possible, par génération automatique de code (C, C+!! +, Java, etc.).

Dans ce contexte, l'Inria soutient l'utilisation de méthodes formelles et d'outils associés permettant la manipulation des schémas de programmes écrits dans les langages d'interface

(génération de code, analyse, transformation et optimisation de code, vérification, génération de tests) afin d'assister les concepteurs d'applications et améliorer la maîtrise du développement.

La seconde phase de REUTEL comporte quatre axes de recherche : nous intervenons principalement dans le thème intitulé "Outils de validation et de génération de code pour UML". L'objectif est la réalisation d'une plate-forme de démonstration pour UML offrant des fonctionnalités de simulation, de vérification et de génération de tests. En 1999, notre contribution a porté sur les points suivants :

- Nous avons poursuivi le développement du simulateur OCIS dans le but d'en faire un outil robuste utilisable pour UML (voir § 5.1.1).
- En collaboration avec Thierry Jéron, Pierre Morel et Séverine Simon de l'équipe PAMPA, nous avons participé à l'amélioration du générateur de tests TGV et à son intégration au sein de la boîte à outils CADP (voir § 5.1.6).
- En collaboration avec Alain Le Guennec de l'équipe PAMPA, nous avons joué un rôle de conseil pour la connexion de l'outil UMLAUT à l'environnement OPEN/CÆSAR. UMLAUT permet d'appliquer des transformations complexes à un modèle UML selon des règles définies par l'utilisateur. En particulier, le modèle UML d'une application répartie peut être automatiquement transformé en un système de transitions étiquetées, ce qui permet de lui appliquer tous les outils disponibles dans l'environnement OPEN/CÆSAR.
- En collaboration avec Hubert Canon de l'équipe PAMPA, nous avons joué un rôle de conseil pour la connexion du compilateur BDL à l'environnement OPEN/CÆSAR. Conçu par les équipes EP-ATR et PAMPA, BDL (*Behavioural Description Language*) est un langage d'interface qui permet d'enrichir les diagrammes de classes UML avec des concepts comportementaux. En effet, la notation UML comporte un grand nombre de diagrammes (diagrammes de séquences, StateCharts...) sans pour autant garantir la cohérence entre ces différentes vues comportementales. La langage BDL se veut un formalisme "pivot" entre les diverses vues. Lorsqu'il est interprété de manière asynchrone, le modèle sémantique sous-jacent est celui des systèmes de transitions étiquetées, ce qui autorise une connexion de BDL avec les outils CADP et TGV.

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) initiée en 1998 par la Direction Scientifique de l'INRIA.

D'une durée de deux ans (1998-1999), cette action regroupe les projets MEIJE, PAMPA et VASY de l'INRIA, ainsi que l'équipe SCOP du laboratoire LSR-IMAG. Elle est consacrée à une recherche fondamentale dans le domaine de la vérification et du test de systèmes réactifs comportant une partie contrôle et une partie données de complexité significative, notamment

en combinant les méthodes de vérification énumérative avec des techniques appropriées pour la représentation et le traitement des données (voir `http://www.inrialpes.fr/vasy/verdon`).

En 1999, nos travaux dans le cadre de VERDON ont porté sur la compilation efficace des types de données pour les langages LOTOS et LOTOS NT (voir § 5.2.1 et § 5.2.3), sur l'amélioration de la vérification énumérative pour LOTOS (voir § 5.2.2), sur la vérification à la volée de formules logiques et sur la génération de diagnostics.

7.1.2 Relations bilatérales nationales

En 1999, nous avons collaboré avec plusieurs projets INRIA :

APACHE (Rhône-Alpes) : utilisation de la plate-forme "grappe de PCs" développée par les projets APACHE et SIRAC pour l'expérimentation d'algorithmes de vérification massivement parallèles ;

BIP (Rhône-Alpes) : connexion des outils de vérification BCG, EVALUATOR 3.0 et XTL à l'environnement de développement pour le langage synchrone ESTEREL et à l'atelier ORCCAD pour la conception de contrôleurs de tâches robotiques ;

MEIJE (Sophia-Antipolis) : collaboration dans le cadre de l'action de recherche coopérative VERDON (voir § 7.1.1) ;

PAMPA (Rennes) : collaboration dans le cadre de l'action FORMALFAME du GIE DYADE (voir § 6.1), du contrat REUTEL-2000 (voir § 6.3) et de l'action de recherche coopérative VERDON (voir § 7.1.1) ;

SIRAC (Rhône-Alpes) : utilisation des outils CADP pour l'analyse de deux protocoles développés au sein du projet SIRAC pour la reconfiguration dynamique d'agents communicants ; utilisation de la plate-forme "grappe de PCs" développée par les projets APACHE et SIRAC.

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

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

Laboratoire LIAFA (Paris) : développement du compilateur TRAIAN (Mihaela Sighireanu) ;

Laboratoire LSR-IMAG (Grenoble) : collaboration dans le cadre de l'action de recherche coopérative VERDON (voir § 7.1.1) afin d'interconnecter l'atelier B et la boîte à outils CADP (Didier Bert, Francis Cave et Marie-Laure Potet) ;

Laboratoire VERIMAG (Grenoble) : coopération pour l'amélioration des outils CADP (Laurent Mounier).

7.2 Actions internationales

7.2.1 Groupes de travail internationaux

– Nous sommes membre de FMICS (*Formal Methods for Industrial Critical Systems*), l'un des onze groupes de travail d'ERCIM. Depuis juillet 1999, H. Garavel coordonne ce groupe qui comprend actuellement 66 chercheurs appartenant à 27 organisations.

  En 1999, H. Garavel et M. Herbert ont conçu et réalisé le nouveau serveur Web de FMICS (voir `http://www.inrialpes.fr/vasy/fmics`) et assuré la mise à jour de la liste de diffusion électronique de FMICS.

– H. Garavel est membre du comité technique (*ETItorial Board*) pour la plate-forme d'intégration logicielle ETI (*Electronic Tool Integration*) développée à l'Université de Dortmund et accessible en ligne par Internet [SMB97,BMW97,MBK97].

7.2.2 Relations bilatérales internationales

Nous entretenons des relations scientifiques avec plusieurs universités et centres de recherche internationaux. En 1999, nous avons notamment collaboré avec :

– l'équipe SEN2 du CWI (Jan-Friso Groote, Judi Romijn et Izak van Langevelde),

– l'Université de Dortmund (Bernard Steffen et Tiziana Margaria),

– l'Université de Stirling (Ken Turner et Ji He),

– et l'Université de Twente (Ed Brinksma, Axel Belinfante, Holger Hermanns et Jan Tretmans).

7.3 Accueil de chercheurs français et étrangers

– Gregor v. Bochmann, professeur à l'Université d'Ottawa (Canada) nous a rendu visite le 12 mai 1999.

– Valentin Cristea, professeur à l'Université Polytechnique de Bucarest (Roumanie) nous a rendu visite du 4 au 10 juin 1999.

– Alain Le Guennec, doctorant dans le projet PAMPA, nous a rendu visite du 22 au 26 février 1999. Il a donné un séminaire présentant une approche à objets pour la modélisation et la validation de systèmes distribués.

[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.

- Holger Hermanns, de l'Université de Twente (Pays-Bas), nous a rendu visite du 14 au 16 septembre 1999. Il a donné un séminaire sur la vérification par modèles des chaînes de Markov à temps continu.
- Le Duc Hoa, doctorant au Laboratoire d'Interaction Homme-Systèmes (LIHS) de l'Université Toulouse 1 nous a rendu visite les 28 et 29 juillet 1999. Il a donné un exposé sur la vérification de propriétés pour des réseaux de Petri étendus dans le cadre du projet ESPRIT MEFISTO.
- Solofo Ramangalahy, ingénieur-expert dans le projet PAMPA, nous a rendu visite du 3 au 6 août et du 4 au 5 novembre 1999.

8 Diffusion de résultats

8.1 Diffusion de logiciels

En 1999, nous avons travaillé à augmenter la diffusion de la boîte à outils CADP :

- En janvier 1999, nous avons finalisé une nouvelle version officielle de CADP (version 97b "Liège").
- A cette occasion, nous avons rédigé et publié le 4^{e} numéro de la lettre électronique "*The CADP Newsletter*" décrivant l'avancement des travaux et les nouveautés concernant CADP (voir `http://www.inrialpes.fr/vasy/cadp/news4.html`).
- Nous avons enrichi la page Web consacrée à CADP (voir `http://www.inrialpes.fr/vasy/cadp`). Les améliorations portent sur les points suivants :
  - Mise en ligne des manuels d'utilisation des outils CADP sous forme de pages HTML,
  - Mise à jour de la liste des études de cas réalisées avec CADP ayant donné lieu à des publications scientifiques ou des rapports de recherche.
- Nous avons effectué des démonstrations publiques de la boîte à outils CADP à l'occasion de la conférence CAV'99 (Trento, Italie, juillet 1999).
- Outre le portage de CADP vers les systèmes d'exploitation WINDOWS 98 et WINDOWS NT (voir § 5.2.5), nous avons traité divers problèmes de portage relatifs aux nouveaux systèmes d'exploitation SOLARIS 7 et REDHAT LINUX 6.

Ces efforts semblent avoir porté leurs fruits puisqu'en 1999, le nombre de licences site pour CADP est passé de 172 à 198.

Par ailleurs, la diffusion de la version 1.0 du compilateur TRAIAN entreprise en 1998 s'est poursuivie : en 1999, 34 sites différents ont téléchargé le compilateur. Les pages Web concernant E-LOTOS (`http://www.inrialpes.fr/vasy/elotos`) et TRAIAN (`http://www.inrialpes.fr/vasy/traian`) ont été mises à jour.

8.2 Animation de la communauté scientifique

- Depuis janvier 1998, H. Garavel est responsable de l'action de recherche coopérative VERDON (voir § 7.1.1).
- Depuis juillet 1999, H. Garavel est responsable du groupe de travail FMICS d'ERCIM (voir § 7.2).
- En collaboration avec Roland Groz (France Telecom/CNET) et Guy Leduc (Université de Liège), H. Garavel a coordonné la préparation d'un numéro thématique de la revue TSI (*Technique et Science Informatiques*) consacré aux réseaux et protocoles [12].
- H. Garavel a été 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 a été membre du comité de programme de CFIP'99 (7^{e} Colloque Francophone sur l'Ingénierie des Protocoles, Nancy, France, 26–29 avril 1999).
- H. Garavel a été membre du comité de programme de FMICS'99 (*4th International ERCIM Workshop on Formal Methods for Industrial Critical Systems*, Trento, Italie, 11–12 juillet 1999).
- H. Garavel a été 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).

8.3 Enseignement universitaire

Le projet VASY est équipe d'accueil pour :

- le DEA "Informatique : Système et Communications" commun à l'Institut National Polytechnique de Grenoble et à l'Université Joseph-Fourier, et
- le DEA "Informatique : communication et coopération dans les systèmes à agents" de l'Université de Savoie.

En 1999 :

- H. Garavel et R. Mateescu ont dispensé le cours "Temps Réel" destiné aux étudiants en 3^{e} année de l'ENSIMAG (21 heures annuelles).
- R. Mateescu a assuré, conjointement avec Flavio Oquendo, le cours "Méthodes formelles pour l'ingénierie des logiciels : spécification et vérification de protocoles" destiné aux étudiants du DEA d'informatique de l'Université de Savoie (24 heures annuelles).
- H. Garavel a donné le 12 janvier 1999 un séminaire d'introduction aux algèbres de processus à l'intention des étudiants de 2^{e} année du Magistère d'Informatique et Modélisation (MIM), formation co-habilitée entre l'Université Claude Bernard (Lyon), l'Université Joseph Fourier (Grenoble) et l'Ecole Normale Supérieure de Lyon.

- M. Cherif, H. Garavel et R. Mateescu ont encadré le mémoire de probatoire [22] de Yan Uschanoff, élève-ingénieur CNAM (centre agréé de Grenoble).

- H. Garavel a participé au jury de thèse de Judi Romijn [Rom99] à l'Université de Twente (Pays-Bas) en octobre 1999.

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

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 a participé aux colloques et conférences ARTS'99, CFIP'99, FMICS'99 et TACAS'99 en qualité de membre du comité de programme (voir § 8.2).

- R. Mateescu a également assisté au colloque FMICS'99.

- H. Garavel a participé à une *panel session* de TACAS'99 intitulée "*Software Engineering and the Verification Tool Builder*".

- Cl. Chaudet et R. Mateescu ont présenté deux exposés consacrés respectivement à la spécification en LOTOS NT du système SCSI-2 et à la vérification en XTL du système SCSI-2 au cours de la troisième réunion de l'action VERDON à l'INRIA Rhône-Alpes, le 3 mars 1999.

- H. Garavel a été invité par l'Université d'Erlangen (Allemagne) à donner, le 25 mai 1999, un séminaire d'une journée consacré aux logiciels BCG, OPEN/CÆSAR et XTL.

- H. Garavel a présenté un exposé sur le bilan de l'utilisation de SYNTAX/FNC2 par l'équipe VASY au cours de la réunion de lancement de l'action SMARTTOOLS de DYADE à l'INRIA Sophia-Antipolis, le 2 juin 1999.

- R. Mateescu a présenté un exposé sur la génération automatique de diagnostics pour l'évaluation à la volée de formules logiques au cours de la quatrième réunion de l'action VERDON à l'INRIA Rennes, le 3 juin 1999.

- H. Garavel a assisté à la conférence CAV'99 (*11th International Conference on Computer-Aided Verification*, Trento, Italie, 7–10 juillet 1999).

- H. Garavel et R. Mateescu ont présenté deux exposés consacrés respectivement à l'optimisation des types structurés dans le compilateur CÆSAR.ADT et à la résolution à la volée des systèmes d'équations booléennes paramétrées au cours de la cinquième réunion de l'action VERDON au laboratoire LSR-IMAG, le 16 décembre 1999.

[Rom99] J. ROMIJN, *Analysing Industrial Protocols with Formal Methods*, thèse de doctorat, University of Twente, The Netherlands, septembre 1999.

9 Bibliographie

Ouvrages et articles de référence de l'équipe

[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, 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 (éditeurs), CWI, p. 187–230, Amsterdam, mai 1998. Invited lecture.

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

[7] 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.

[8] 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.

[9] 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 (éditeur), *Lecture Notes in Computer Science, 1384*, Springer Verlag, p. 68–84, Berlin, mars 1998. Full version available as INRIA Research Report RR-3352.

[10] 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 (éditeur), BRICS, p. 33–42, juillet 1998.

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

Livres et monographies

[12] H. Garavel, R. Groz, G. Leduc (éditeurs), *Réseaux et protocoles, 18-6*, Technique et Science Informatiques. Hermès, Paris, juin 1999.

Thèses et habilitations à diriger des recherches

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

Communications à des congrès, colloques, etc.

[14] H. Garavel, M. Sighireanu, « A Graphical Parallel Composition Operator for Process Algebras », *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'99 (Beijing, China)*, J. Wu, Q. Gao, S. T. Chanson (éditeurs), IFIP, Kluwer Academic Publishers, p. 185–202, octobre 1999.

[15] J. Groote, R. Mateescu, « Verification of Temporal Properties of Processes in a Setting with Data », *in : Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology AMAST'98 (Amazonia, Brazil)*, A. M. Haeberer (éditeur), *Lecture Notes in Computer Science, 1548*, Springer Verlag, p. 74–90, janvier 1999. Full version available as CWI Technical Report SEN-R9804, Amsterdam, The Netherlands.

[16] W. Janssen, R. Mateescu, S. Mauw, P. Fennema, P. van der Stappen, «Model Checking for Managers», *in : Proceedings of the 5th and 6th International SPIN Workshops on Theoretical and Practical Aspects of SPIN Model Checking, Trento (Italy), Toulouse (France)*, D. Dams, R. Gerth, S. Leue, M. Massink (éditeurs), *Lecture Notes in Computer Science, 1680*, Springer Verlag, p. 92–107, septembre 1999.

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

[18] C. Pecheur, « Advanced Modelling and Verification Techniques Applied to a Cluster File System », *in : Proceedings of the 14th IEEE International Conference on Automated Software Engineering ASE-99 (Cocoa Beach, Florida, USA)*, R. J. Hall, E. Tyugu (éditeurs), IEEE Computer Society, octobre 1999. Extended version available as INRIA Research Report RR-3416.

Rapports de recherche et publications internes

[19] M. Aguilar Cornejo, «Etude comparative et application des techniques de description formelle LOTOS et E-LOTOS à des protocoles pour les systèmes répartis», *Mémoire de DEA*, juin 1999.

[20] C. Chaudet, «Compilation optimisée des types de données LOTOS, E-LOTOS et LOTOS NT», *Mémoire d'ingénieur*, Institut d'Informatique d'Entreprise - CNAM, Evry, juin 1999.

[21] A. Mazzilli, « Etude de la migration et de la portabilité des applications informatiques d'Unix vers Windows NT », *Mémoire d'ingénieur*, CNAM, Grenoble, décembre 1999.

[22] Y. Uschanoff, « Présentation du programme de vérification Verisoft », *Mémoire de probatoire en informatique*, CNAM, Grenoble, juin 1999.

Divers

[23] D. Latella, S. Gnesi, H. Garavel, « Towards Reliable Computer Systems», Ercim News 39, octobre 1999.