PROJET SPECTRE

Spécification et programmation des systèmes communicants et temps réel

Localisation : *Grenoble*

Mots-clés : concurrence, génération de code, génie logiciel, interprétation abstraite, langage synchrone, logique temporelle, parallélisme, programmation parallèle, protocole de communication, sémantique, spécification formelle, système hybride, temps réel, vérification de programme.

1 Composition de l'équipe

Responsable scientifique et permanent

Joseph Sifakis, DR, CNRS

Secrétariat

Chantal Costes, AAR, CNRS (jusqu'au 1-6-1996)
Claude Amieux, AARP, CNRS (depuis le 1-1-1996)
Christine Servonnet, SAR contractuelle

Personnel Inria

Hubert Garavel, CR
Carlos Rodriguez, Ingénieur expert

Personnel des établissements partenaires (UMR VERIMAG)

Saddek Bensalem, MC, UJF, en année sabbatique du 1-10-95 au 30-9-96 [1]
Ahmed Bouajjani, MC, UJF, détaché au CNRS du 1-10-1994 au 30-9-96
Paul Caspi, DR, CNRS
Jean-Claude Fernandez, MC, UJF, détaché à l'Inria du 1-10-1994 au 30-9-96
Susanne Graf, CR, CNRS
Nicolas Halbwachs, DR, CNRS
Fabienne Lagnier, MC, UJF
Oded Maler, CR, CNRS
Florence Maraninchi, MC, UJF
Laurent Mounier, MC, UJF
Xavier Nicollin, MC, INPG [2]

[1] Université Joseph Fourier

[2] Institut National Polytechnique de Grenoble

Pascal Raymond, CR, CNRS
Ghislaine Thuau, MC, INPG
Sergio Yovine, CR, CNRS

Chercheurs post-doctorants

Eric Conquet, Matra Marconi Space (jusqu'au 1-3-1996)
Alain Kerbrat, Inria
Catherine Parent, Inria, jusqu'au 30-9-96
Florence Pagani, ATER, INPG à partir du 1-9-96

Chercheurs doctorants

Sébastien Bornot, boursier normalien
Marius Bozga, boursier MENESR
Conrado Daws, boursier MENESR
Thomas Goust, boursier Cifre, SGS-Thomson
Peter Habermehl, boursier MENESR
David Lesens, boursier MENESR
Radu Mateescu, boursier MENESR
Yann-Erik Proy, boursier Cifre, Schneider Electric
Patrick Roumanoff, boursier DRET
Hassen Saïdi, boursier du gouvernement algérien
Mihaela Sighireanu, boursier MENESR
Stavros Tripakis, boursier MENESR
Daniel Weber, thésard CNAM

2 Présentation du projet

Le projet SPECTRE est un projet commun entre le CNRS, l'INPG, l'UJF et l'Inria, localisé dans les locaux de VÉRIMAG.

L'évolution de l'informatique a conduit au développement de systèmes complexes, constitués de composants qui coopèrent. Les méthodes de conception de tels systèmes sont loin d'être maîtrisées et restent encore peu explorées. Ceci est dû au fait que la mise en œuvre des systèmes parallèles et distribués soulève de nombreux problèmes concernant le choix et l'utilisation des langages de description et des méthodes de programmation et de validation associées.

Répondre aux besoins des concepteurs des systèmes reviendrait à fournir des environnements qui supportent des langages de spécification et de programmation adéquats et qui intègrent des outils d'analyse et de génération de code. À l'heure actuelle, cet objectif paraît trop ambitieux si l'on veut l'atteindre dans sa généralité.

L'objectif du projet est d'aider le concepteur de certains types d'applications parallèles et temps réel, pour lesquelles le développement d'une méthode de conception rigoureuse semble à la fois nécessaire et réaliste à l'heure actuelle. Il s'agit de systèmes informatiques critiques dont le rôle est d'interagir de manière permanente avec un environnement. Des exemples typiques de tels systèmes sont les protocoles de communication et les systèmes de commande temps réel.

Dans ce domaine le projet s'attache à fournir des méthodes et des outils d'aide pour les trois tâches principales de la conception : la programmation, la spécification et la validation.

Programmation : par programmation on entend l'écriture d'algorithmes qui décrivent le fonctionnement d'un système. Nous nous intéressons à la définition, l'expérimentation et l'implantation de langages de haut niveau qui permettent une expression directe du parallélisme et des contraintes temps réel. Dans ce cadre, nous étudions la sémantique des langages et les techniques de compilation associées.

Concernant les langages de programmation, les recherches entreprises dans le cadre du projet sont menées selon deux axes :

- L'étude formelle et la compilation de langages existants, comme ESTELLE, LOTOS ou SDL, dans le but d'admettre ces langages en entrée de nos outils de validation.
- La conception et la compilation de nouveaux langages, mieux adaptés, à notre avis, à certains types d'applications. C'est le cas des langages synchrones LUSTRE et ARGOS.

Ces travaux sont menés d'une part sur le plan formel, par une étude approfondie de la sémantique des langages, considérée comme un pré-requis de la compilation et de la validation, et d'autre part en vue de la réalisation d'outils (compilateurs), dont les performances sont un critère prépondérant.

Spécification : la spécification est la description du comportement d'un système vis-à-vis de l'ensemble de ses utilisateurs. De nombreux langages ont été proposés pour la spécification des systèmes parallèles. Selon leur type, on peut distinguer deux approches :

- La première approche consiste à spécifier un système en donnant une description dont la sémantique est fondée sur les systèmes de transitions (sémantique opérationnelle). Des spécifications de ce type permettent de décrire le comportement d'un système comme la composition de comportements élémentaires. Les réseaux de Petri, les graphes d'états, les algèbres de processus et des langages tels que ESTELLE, LOTOS ou SDL, sont quelques exemples de formalismes utilisés.
- La deuxième approche consiste à exprimer les spécifications par un ensemble de propriétés, une propriété représentant une classe de systèmes. Le langage de spécification utilisé dans ce cas est un langage de type déclaratif, le plus souvent le langage des formules d'une logique. Les logiques de programmes, notamment les logiques temporelles, sont des exemples de formalismes utilisés pour l'expression des propriétés.

Nous pensons que les formalismes existants sont essentiellement complémentaires et que leur combinaison peut s'avérer intéressante. En effet, les langages déclaratifs tels que les logiques sont mieux adaptés à l'expression de propriétés globales — par exemple, l'exclusion mutuelle, l'absence de blocage ou l'absence de famine — tandis que les formalismes à base de systèmes de transitions se prêtent mieux à l'expression des relations de causalité directe et du séquencement.

Validation : les méthodes de validation sont nécessaires depuis les spécifications initiales jusqu'à l'implémentation. Elles permettent d'assurer que les choix effectués satisfont les besoins et éventuellement de réajuster ces choix. Elles concernent des activités diverses comme la vérification formelle, la simulation et le test. Elles doivent permettre d'intégrer, d'interpréter et de corriger les erreurs.

L'étude des méthodes formelles de validation et leur application effective aux systèmes parallèles ont connu un regain d'intérêt, pour deux raisons essentielles :

- La validation par des méthodes empiriques devient problématique : les systèmes réactifs, notamment temps réel, sont souvent des systèmes critiques, dont certaines propriétés doivent être absolument satisfaites. Par ailleurs, le non-déterminisme inhérent au parallélisme asynchrone rend l'expérimentation non réproductible et affaiblit considérablement la confiance que l'on peut accorder à une validation par test.
- D'autre part, on a constaté que les propriétés cruciales d'un système parallèle peuvent être étudiées sur des abstractions qui admettent souvent des modélisations finies. Alors les problèmes relatifs à la validation se simplifient, voire deviennent décidables.

Les méthodes de validation étudiées et mises en œuvre dans le cadre du projet SPECTRE sont fondées sur l'analyse d'un modèle sémantique des programmes.

Ce sont des méthodes automatiques, mais partielles, au sens où :

– soit elles ne s'appliquent qu'à certaines classes particulières de problèmes ; c'est le cas des méthodes fondées sur les modèles finis, qui s'appliquent aux programmes n'ayant qu'un nombre fini (et raisonnable) d'états ou à la vérification de propriétés analysables sur une abstraction finie du programme ;

– soit elles ne fournissent que des résultats partiels : sous cette rubrique, on peut ranger la génération de séquences de test (le test vise à prouver que le programme est faux, mais ne peut en général servir à prouver qu'il est correct), mais aussi les méthodes approchées fondées sur l'interprétation abstraite des programmes, qui peuvent fournir le résultat inverse : si la vérification d'une propriété réussit, alors la propriété est vraie, mais l'échec de la vérification est sans signification.

Trois facteurs interviennent dans la conception et la réalisation des outils actuels :

– l'utilisation de langages de programmation de haut niveau, tels que ESTELLE, LOTOS, SDL, LUSTRE ou ARGOS ;

– une recherche de performances pour aborder la vérification de systèmes réels ;

– un souci de réutilisabilité des composants logiciels développés afin de réduire l'effort de programmation.

Le projet consiste en un ensemble d'actions orientées vers des domaines d'applications particuliers — systèmes réactifs (voir section 3.1), systèmes distribués (voir section 3.2), systèmes temporisés (voir section 3.3) — et une action transversale fournissant la technologie de base pour la validation (voir sections 3.4, 3.5 et 3.6).

3 Actions de recherche

3.1 Langages synchrones et systèmes réactifs

Les langages synchrones constituent une famille de langages de haut niveau, dédiés à la programmation des systèmes réactifs. Principalement développés en France, ces langages ont donné lieu à une activité intense et à une collaboration étroite entre les équipes concernées : CMA/ENSM et Inria-Sophia — qui développe le langage ESTEREL—, Irisa/Inria-Rennes — qui développe le langage SIGNAL, et SPECTRE.

Deux langages synchrones sont étudiés dans le cadre du projet :

– le langage déclaratif LUSTRE adopte une approche "flot de données". Développé depuis 1984, ce langage fait l'objet d'une industrialisation par la société VÉRILOG.

– le langage ARGOS inspiré des STATECHARTS, est fondé sur une description en termes d'automates parallèles hiérarchisés.

Les recherches autour de ces langages portent sur leur sémantique, leur compilation — vers du code séquentiel, réparti, ou vers des circuits —, la spécification et la vérification de programmes synchrones, et diverses extensions : extension récursive de LUSTRE, extensions temporisées et hybrides d'ARGOS.

La collaboration visant à définir et développer les formats communs aux langages synchrones se poursuit, notamment dans le cadre du contrat EUREKA-SYNCHRON, qui se termine fin 96. Cette année, nous avons agi comme coordinateur de la proposition ESPRIT-LTR "SYRF", qui a été acceptée.

3.1.1 Langages et Compilation

Participants : Nicolas Halbwachs, Fabienne Lagnier, Florence Maraninchi, Pascal Raymond

Dans [635], nous avons étudié les problèmes que pose l'introduction du non-déterminisme dans les langages synchrones. Cette extension est utile lorsqu'on utilise ces langages non plus pour programmer, mais pour spécifier les systèmes, ou pour modéliser leur environnement. Le problème est que l'on veut permettre l'expression d'un non-déterminisme *explicite*, tout en interdisant l'apparition d'un non-déterminisme *implicite*, conséquence de la composition parallèle de processus. La solution consiste à contrôler le non-déterminisme à l'aide de signaux d'entrée fictifs, ou "oracles", à vérifier le déterminisme de la composition des processus ainsi déterminisés, puis à retirer les oracles. Cette construction n'est correcte que sous certaines conditions concernant le mode d'introduction des oracles.

Sur un plan plus pratique, nous avons développé un outil permettant de décrire un ensemble de comportements sous forme d'expressions régulières. Nommé "REGLO", cet outil compile les expressions de comportements en observateurs LUSTRE. Il offre ainsi un nouveau mode d'expression, particulièrement pratique pour exprimer les hypothèses sur l'environnement d'un programme. Par ailleurs, REGLO met en œuvre un nouvel algorithme, *linéaire*, de construction de reconnaisseur d'expressions régulières [638].

Une nouvelle version, solidifiée, du compilateur LUSTRE est en cours d'écriture par F. Lagnier, N. Halbwachs et P. Raymond. Cette version traitera complètement les tableaux de LUSTRE. Aussi, un outil d'interprétation de programmes LUSTRE et de simulation sous X11 a été développé.

La compilation d'ARGOS dans le format commun équationnel DC (pour "declarative code") a été étudiée [634]. En particulier, cette traduction tire parti des "conditions d'activations" de DC, pour exprimer l'activité sporadique des processus raffinant les états d'un automate ARGOS.

3.1.2 Répartition de code

Participant : Paul Caspi

Cette activité vise à produire des codes répartis à partir de programmes synchrones. En effet, les langages synchrones sont des langages hautement parallèles, et il était donc dommage de se restreindre aux seules compilations en programmes centralisés. Elle a consisté, par le passé, à construire et expérimenter un outil pratique de répartition de programmes synchrones, et à étudier les bases sémantiques des exécutions réparties de ces programmes.

Cette activité est en sommeil depuis le départ d'Alain Girault. Nous avons cependant publié au cours de l'année, quelques résultats la concernant [616]. Elle devrait se poursuivre à l'avenir dans le cadre d'un projet de contrat Esprit appelé Crisys avec Schneider Electric.

3.1.3 Vérification de programmes

Participants : Saddek Bensalem, Paul Caspi, Nicolas Halbwachs, David Lesens, Catherine Parent, Pascal Raymond, Ghislaine Thuau

L'expérience montre que, dans le domaine des systèmes réactifs, les propriétés critiques d'un programme sont presque toujours des propriétés de "sûreté", exprimant que quelque chose de fâcheux ne se produit jamais au cours de l'exécution du programme. L'approche synchrone fournit une méthode originale pour spécifier ce type de propriétés, par l'écriture d'un programme "observateur" qui lit les entrées et les sorties du programme à vérifier et décide à chaque instant si la propriété est satisfaite. On doit alors vérifier que la composition parallèle du programme initial et de son observateur ne signale jamais de violation de la propriété. Nous avons développé des outils de vérification fondés sur l'examen de l'automate de contrôle du programme global, représenté soit en extension, soit symboliquement à l'aide de BDDs.

Dans le cas de LUSTRE, qui permet l'usage de variables numériques, cette vérification n'est qu'approchée, puisque l'automate n'est qu'une abstraction finie du programme. Nous étudions donc

aussi des extensions numériques de notre outil de vérification, fondées sur l'utilisation du module POLKA de manipulation de systèmes de contraintes linéaires (voir §3.5.2).

Une autre voie de recherche concerne la vérification de réseaux réguliers de taille variable. Il s'agit, par exemple, de prouver des propriétés d'une composition de n processus identiques, quel que soit le nombre n de processus. L'indécidabilité de ce type de problème est bien connue, même lorsque chacun des processus mis en jeu est d'états finis. Nous étudions [632] des procédés heuristiques, relevant de l'interprétation abstraite, pour découvrir des fonctionnements invariants de telles familles de réseaux, ces invariants permettant dans certains cas de prouver des propriétés. Ces travaux sont effectués dans le cadre de LUSTRE, mais sont généralisables à n'importe quel langage synchrone. Ce thème fait l'objet du travail de thèse de D. Lesens.

Enfin, une maquette de traducteur de LUSTRE vers PVS, l'éditeur de spécifications et de preuves du Stanford Research Institute, a été réalisée par C. Parent, S. Bensalem et P. Caspi. Des exemples ont été traités, qui ont montré l'intérêt de cet outil pour vérifier interactivement des programmes LUSTRE qui, parce qu'ils manipulent des données non finies, ne peuvent être traités par LÉSAR ou POLKA.

3.1.4 Extensions récursives de Lustre

Participants : Paul Caspi, Marc Pouzet

Ces travaux visent à étendre la notion de synchronisme à des constructions plus générales, pas forcément réactives. Nous avons montré, au cours de l'année écoulée, comment on peut étendre la sémantique opérationnelle synchrone et les vérifications statiques de synchronisme (calculs d'horloges) à une extension de LUSTRE, comportant un opérateur d'abstraction [637, 621]. Cela nous permet, par exemple, de donner un sens synchrone à des réseaux de flots de données dynamiques (Réseaux de Kahn synchrones). Ces travaux devraient se poursuivre par le développement d'un compilateur pour ce type de réseaux, en collaboration avec Marc Pouzet, Maître de Conférence à Paris 6.

3.1.5 L'environnement SYNCHRONE

Participants : Paul Caspi, Nicolas Halbwachs, Fabienne Lagnier, Pascal Raymond

Nous participons activement au projet EUREKA SYNCHRON, dont le but est de définir et de normaliser un ensemble de formats d'échange, communs aux langages synchrones, afin de constituer un socle commun à partir duquel des outils compatibles sont développés.

Notre équipe a été l'acteur principal de la conception du nouveau format DC (pour "declarative code"), dont la première version est maintenant stable. Nous travaillons activement à la connexion de nos outils à ce format.

Cette année, nous avons coordonné la proposition ESPRIT-LTR "SYRF", qui a été acceptée : Le projet devrait démarrer début 97, et réunit les trois projets Inria travaillant sur les langages synchrones, la GMD (A. Poigné, Allemagne), l'université de Linköping (S. Nadjm-Tehrani, Suède), et les sociétés Logikkonsult (Suède), SAAB-MA (Suède), Schneider-Electric et EDF.

3.2 Protocoles et systèmes distribués

La spécification des protocoles et des systèmes répartis nécessite l'utilisation de langages de haut niveau permettant une description aisée de la dispersion du contrôle et des problèmes liés à la synchronisation et à la communication entre entités séparées. Comparés aux langages synchrones, ces langages doivent permettre la description de comportements asynchrones et ils doivent fournir les moyens pour une description de données complexes et variées. Toutefois, avec l'apparition de protocoles à contraintes temporelles fortes — protocoles utilisés dans les réseaux hauts-débits — le besoin apparaît pour des langages qui combinent les caractéristiques des langages synchrones et asynchrones.

Depuis 1986, nous nous intéressons au langage LOTOS, une norme ISO pour la spécification des protocoles et des systèmes répartis. Il s'agit d'un langage de spécification algébrique de systèmes de processus communicants qui intègre les calculs de processus CCS et CSP pour la description du contrôle et les types abstraits algébriques pour la description des données.

Nous avons développé deux compilateurs efficaces (CÆSAR et CÆSAR.ADT) pour le langage LOTOS qui, avec l'outil de vérification ALDÉBARAN, constituent le socle de notre boîte à outils pour l'ingénierie des protocoles :

- CÆSAR est un compilateur qui produit, à partir d'un programme LOTOS, du code exécutable ou des modèles sur lesquels différentes méthodes de vérification peuvent être appliquées. Le programme source LOTOS est traduit successivement en une algèbre de processus simplifiée, un réseau de Petri étendu avec des variables et des transitions atomiques, et finalement un système de transitions étiquetées obtenu par simulation exhaustive.
- CÆSAR.ADT est un compilateur qui traduit les définitions de types abstraits LOTOS vers des bibliothèques de types et de fonctions en langage C. La traduction met en œuvre un algorithme de compilation par filtrage et des techniques pour la reconnaissance des classes de types usuels (nombres entiers, énumérations, tuples, listes...) qui sont identifiées automatiquement et implémentées de manière optimale.

3.2.1 Compilation des algèbres de processus

Participant : Hubert Garavel

En 1996, les travaux sur CÆSAR et CÆSAR.ADT se sont, pour l'essentiel, limités à la maintenance évolutive de ces deux outils.

Il faut toutefois mentionner le travail sur la génération compositionnelle des réseaux de Petri, entrepris les années précédentes, qui a été achevé et intégré dans CÆSAR. L'application de techniques de génération/minimisation alternées permet, sur certains "gros" exemples, une réduction importante de la taille des réseaux de Petri produits.

Dans le cadre du GIE DYADE, nous étudions diverses adaptations de CÆSAR qui pourraient permettre de minimiser davantage la taille des systèmes de transitions engendrés.

3.2.2 Etudes de cas

Participants : Hubert Garavel, Radu Mateescu, Laurent Mounier, Mihaela Sighireanu

En 1996, une part importante de nos travaux a été consacrée au traitement d'applications complexes qui ont été modélisées en LOTOS et analysées grâce à la boîte à outils CÆSAR/ALDÉBARAN :

- Dans le cadre l'action VASY (*VAlidation de SYstèmes*) du GIE BULL-Inria, nous avons collaboré avec les ingénieurs de Bull chargés de modéliser en LOTOS et de vérifier l'arbitre de bus de l'architecture matérielle POWERSCALE. Ce travail a donné lieu à publication [622].
- R. Mateescu a décrit analysé le protocole BRP (*Bounded Retransmission Protocol*) utilisé par Philips dans l'un de ses produits audiovisuels. Ce travail a donné lieu à publication [636].
- H. Garavel et L. Mounier ont décrit et analysé divers algorithmes distribués d'élection sur un réseau en anneau unidirectionnel avec perte de messages et pannes de station. Ce travail a donné lieu à publication [614].
- Nous avons utilisé nos outils pour analyser la description formelle en LOTOS du protocole OSI-TP (*Distributed Transaction Processing*), dans laquelle nous avons découvert plusieurs erreurs qui ont été rapportées à l'AFNOR [651].

D'autres équipes ont également utilisé nos outils pour diverses études de cas. Pour ne citer que les travaux publiés, on peut mentionner l'analyse de configurations ferroviaires (SICS, Suède), la caractérisation d'une erreur dans le protocole TCP (GMD-FOKUS, Allemagne), l'étude des interaction entre services téléphoniques (CWI, Pays-Bas), ainsi que la mise en défaut du protocole de sécurité EQUICRYPT basé sur le principe des "tiers de confiance" et destiné à contrôler l'accès à des services multimédia (Université de Liège, Belgique).

3.2.3 Contribution à l'action de normalisation E-LOTOS

Participants : Bruno Vivien, Hubert Garavel, Radu Mateescu, Mihaela Sighireanu

Une révision de la norme LOTOS est actuellement en cours à l'ISO : elle devrait conduire à un nouveau langage, baptisé E-LOTOS (*Extended*-LOTOS) adapté aux nouvelles générations de protocoles et de systèmes distribués de type ODP (*Open Distributed Processing*).

Comme délégués AFNOR (France) et RSI (Roumanie), nous avons participé aux réunions ISO qui se sont tenues à Liège et Kansas City et nous organisons la prochaine réunion ISO qui se tiendra à Grenoble en décembre 1996. Comme coordinateur du groupe de travail no 1 du projet COST-247, nous avons participé à l'organisation de trois réunions à Madrid, Maribor et Antalya. Nos contributions ont porté sur les axes suivants :

- Concernant la partie "données" du futur langage E-LOTOS, nous avons montré que certaines propositions existantes, strictement basées sur ML, étaient mal adaptées aux protocoles [646]. C'est pourquoi nous avons proposé une solution alternative [652] dont nous avons étudié l'application à la description de différentes normes OSI et ODP [650].

- Concernant la partie "contrôle", nous avons proposé diverses extensions à l'algèbre de processus LOTOS [648], parmi lesquelles une proposition pour introduire un mécanisme d'exceptions, qui a fait l'objet d'une publication [628].

- Concernant les modules, nous avons proposé d'enrichir le système de modules par diverses classes d'assertions (équations algébriques, formules de logique temporelles, relations de bisimulations et de préordres) afin de simplifier la vérification et de permettre l'expression des exigences (*requirements*) dès les premières phases de la conception d'un système [645].

- Au sein du comité de normalisation d'E-LOTOS, nous avons en charge la définition des aspects "externes" du langage qui comprennent notamment : la syntaxe et la sémantique statique du langage, ainsi que la traduction vers le modèle sémantique "interne" [647, 653].

Enfin, nous avons étudié le système de génération de compilateurs SYNTAX/FNC-2 développé à l'Inria-Rocquencourt. Suite aux conclusions de cette étude [654], nous avons retenu ce système pour servir de base de développement d'un futur compilateur E-LOTOS.

3.3 Systèmes temporisés et hybrides

Les travaux sur les systèmes ayant une base de temps continu ont connu un essor spectaculaire pendant les dernières années, et ceci essentiellement pour deux raisons : d'une part, la découverte de modèles sémantiques fondés sur les systèmes de transitions a permis d'étendre aux systèmes temporisés les méthodes classiques d'analyse des programmes ; d'autre part, de récents résultats ont montré que l'analyse de certaines classes de ces systèmes n'est pas plus difficile que l'analyse des classes correspondantes ayant une base de temps discret.

Du point de vue pratique, les résultats sur les systèmes temporisés permettent d'élargir le champ d'application de nos méthodes aux systèmes hybrides, systèmes contenant des composantes discrètes et continues. Ils permettent également une analyse plus fine des systèmes temps réel qui tient compte de leur environnement, souvent modélisé avec des variables continues.

3.3.1 Modèles des systèmes temporisés et hybrides

Participants : Oded Maler, Ahmed Bouajjani, Xavier Nicollin, Joseph Sifakis, Sergio Yovine

Nos travaux sur les systèmes ayant une base de temps continu nous ont conduit à la définition d'un modèle général pour systèmes hybrides. Ce modèle peut être considéré comme un automate étendu avec un ensemble de variables changeant de manière continue avec le temps. L'état d'un système dans ce modèle est déterminé par un état de l'automate (état de contrôle) et une valuation des variables. L'état peut changer soit de manière discrète en exécutant une transition qui le fait changer d'état de contrôle, soit de façon continue en laissant progresser le temps à partir du même état de contrôle. Les changements d'état sont spécifiés dans le modèle en étiquetant le graphe de l'automate :

- Les *transitions* sont étiquetées par des commandes gardées simples, des paires (garde, affectation). Une transition est franchissable si sa garde (contrainte sur les variables) est satisfaite. Son exécution entraîne la modification des variables selon l'affectation et le changement de l'état de contrôle correspondant.
- Les *états* sont étiquetées par un ensemble de contraintes portant sur les valeurs des variables et leurs dérivées. Les contraintes sur les dérivées définissent les lois d'évolution des variables en fonction du temps dans chaque état de contrôle ; les contraintes sur les valeurs, appelées *invariant*, définissent la condition de séjour dans chaque état de contrôle.

Différentes sous-classes du modèle général ont été étudiées.

- Les *automates hybrides linéaires* sont des systèmes hybrides où toutes les conditions portant sur les variables ou leurs dérivées sont des contraintes linéaires et les termes des affectations sont des expressions linéaires.
- Les *automates temporisés* sont des automates hybrides linéaires dont les variables sont des *horloges* (leurs dérivées sont égales à 1 dans tout état de contrôle). Les invariants et les gardes sont des comparaisons d'horloges à des constantes, et les affectations sont des remises à zéro.
- Les *automates à intégrateurs* se situant entre les deux classes mentionnées ci-dessus. Ils diffèrent des automates temporisés seulement par le fait que les dérivées des horloges peuvent prendre la valeur 1 ou 0. La valeur 0 correspond au blocage de la progression d'une horloge dans un état de contrôle et permet de mesurer et de comparer des *durées*.
- Les *systèmes dynamiques avec dérivées constantes* (PCD) sont des cas particuliers de systèmes hybrides linéaires où la dynamique discrète est simplifiée en interdisant les affectations des variables et le non-déterminisme. Ces systèmes sont proches des systèmes dynamiques classiques.

Cette année, nous avons défini une extension temps réel des expressions régulières pour décrire des signaux (fonctions de réels vers un alphabet) et nous avons montré pour ce formalisme les théorèmes correspondant aux théorèmes de Kleene et de Büchi pour les automates. Autrement dit, nous avons prouvé que le pouvoir expressif de notre formalisme est celui des automates temporisés [642].

3.3.2 Spécification compositionnelle des systèmes temporisés et hybrides

Participants : Sébastien Bornot, Joseph Sifakis, Sergio Yovine

Les modèles utilisés pour la description des systèmes hybrides sont des extensions des modèles discrets et non temporisés par addition de constructions qui permettent d'introduire le temps implicitement ou explicitement. Un point subtil dans la définition de leur sémantique est la façon dont les changements d'état continus interagissent avec les changements d'état discrets. Ceci est fait en pratique par des contraintes, appelées *invariants*, restreignant le domaine des variables continues : le temps peut avancer tant que les états atteints ne violent pas les invariants associés. L'utilisation d'invariants permet

d'exprimer l'urgence des actions. Lorsque lors d'une évolution continue un état est atteint au delà duquel un invariant n'est plus vérifié, l'avancement du temps s'arrête et seulement des actions peuvent être exécutées. Ce mécanisme est essentiel pour modéliser des situations où des actions ont lieu avant des dates limites.

L'utilisation d'invariants ou d'autres mécanismes équivalents peut être source d'inconsistances dans les spécifications parallèles. Par exemple, lorsque deux actions, une entrée et une sortie, soumises à des contraintes de date limite locales, participent à un rendez-vous, un blocage peut se produire lors de leur composition.

L'objectif de ce travail est l'étude de conditions et de méthodes de composition de systèmes hybrides qui préservent l'absence de blocage ; si les composants sont sans blocage, le système obtenu par composition n'a pas de blocages dus à la composition de contraintes temporelles. Les premiers résultats de notre étude publiés dans [640] montrent que — contrairement aux systèmes non temporisés où la synchronisation entre deux actions s'effectue par conjonction de leurs gardes respectives — pour les systèmes temporisés on peut définir des règles de composition de nature disjonctive. Un des avantages de ces dernières est qu'elles n'introduisent pas de blocages car elles tolèrent l'attente même indéfinie. Ce point de vue est illustré en montrant que des règles de ce type peuvent être utilisées pour obtenir les réseaux de Petri temporisés comme la composition d'automates temporisés. Nous travaillons actuellement pour le développement d'une théorie générale pour la spécification compositionnelle des systèmes temporisés.

3.3.3 Spécification logique des systèmes temporisés et hybrides

Participants : Ahmed Bouajjani, Yassine Lakhnech, Sergio Yovine

Nous exprimons les propriétés des systèmes hybrides dans des logiques temporelles dites "temps réel" — extensions de logiques du temps qualitatif — qui permettent d'exprimer des contraintes où intervient le temps quantitatif défini sur les réels. Un exemple de telles logiques est TCTL une extension de la logique CTL dont les modalités "possible" et "inévitable" sont paramétrées par des contraintes sur le temps où l'argument de ces opérateurs devient vrai. Ainsi, on peut exprimer dans cette logique des propriétés de la forme, *il est inévitable que* ou *il est possible que* dans un certain temps un événement se produise. Nous avons déjà étudié une méthode de vérification symbolique de propriétés temps réel exprimées en TCTL pour des automates temporisés, implémentée dans le cadre de l'outil KRONOS.

La logique TCTL est une logique à sémantique arborescente. L'année précédente, nous avons étudié des logiques ayant une sémantique linéaire permettant de décrire des classes de propriétés différentes. Nous avons défini une nouvelle logique temporisée appelée TATL qui est une extension de la logique temporelle propositionnelle par des contraintes utilisant des automates temporisés. Ces contraintes permettent de dire qu'une séquence d'exécution finie depuis un certain point donné est reconnue par un automate temporisé donné. Bien que la vérification soit indécidable pour la logique TATL toute entière, nous en identifions un fragment expressif où ce problème est décidable pour des automates temporisés. Nous avons aussi étudié les liens entre la logique CoD (*calcul des durées*) et les automates hybrides. Le problème de la vérification posé dans le cadre du CoD consiste à prouver une implication entre deux formules, l'une décrivant le système et l'autre ses spécifications. Ces formules sont exprimées dans des fragments identifiés par la communauté des utilisateurs du CoD (PROCOS) pour la description des systèmes et des spécifications. Nous avons montré qu'il est possible de réduire le problème de la vérification dans le CoD à un problème d'accessibilité dans une classe d'automates hybrides où ce problème est décidable.

Dans [620], nous unifions et nous simplifions les constructions et les méthodes de décisions obtenues l'année précédente. Nous considérons un cadre général en définissant une logique appelée HATL. Cette logique est une extension de la logique temporelle propositionnelle par des contraintes utilisant des automates hybrides. Ainsi, cette logique est une extension naturelle de TATL. Nous donnons une

traduction assez simple de fragments du CoD dans HATL. Enfin, nous montrons les liens entre cette logique et les automates hybrides et temporisés.

Dans [619], nous étudions des logiques à sémantique arborescente dans le but de définir des extensions de TCTL pour lesquelles le problème de la vérification est décidable. Nous proposons une extension de la logique temporelle arborescente CTL* par des modalités du passé et par des contraintes par automates temporisés (selon la même approche adoptée pour TATL). La considération d'opérateurs du passé conjointement avec les contraintes par automates permet d'exprimer les propriétés temporisées de manière simple, concise et naturelle. Nous illustrons ce fait en exprimant dans notre logique les spécifications d'un protocole de contrôle d'audio développé par Philips. Nous donnons ensuite un algorithme de vérification pour cette logique.

3.4 Méthodes de vérification énumératives

Les méthodes de vérification consistent à comparer une forme intermédiaire (*modèle*) d'un programme avec la propriété que l'on veut vérifier.

Les méthodes énumératives sont mises en œuvre dans la boîte à outils CADP (CÆSAR/ALDÉBARAN DEVELOPMENT PACKAGE). Ces méthodes rencontrent un problème crucial, celui de l'"explosion d'états" qui survient lorsque le nombre d'états du système à vérifier dépasse les capacités de la machine. En fonction de ce problème, on peut vérifier :

- soit à partir d'une représentation *explicite* du graphe en gardant en mémoire l'ensemble de ses états et transitions,
- soit à partir d'une représentation *implicite* consistant à garder en mémoire seulement un fragment "utile" du graphe.

Outre les compilateurs CÆSAR et CÆSAR.ADT déjà mentionnés, notre boîte à outils CADP comprend :

- l'outil de vérification ALDÉBARAN pour la comparaison et la réduction de graphes modulo une relation de d'équivalence appropriée,
- l'environnement BCG (*Binary Coded Graphs*) pour la manipulation des graphes représentés sous forme explicite,
- et l'environnement OPEN/CÆSAR permettant l'application de différentes méthodes d'analyse aux modèles engendrés, y compris la vérification à la volée et la simulation pas à pas.

3.4.1 Représentation explicite des graphes

Participants : Hubert Garavel, Radu Mateescu

Nous avons défini un format baptisé BCG qui met en œuvre des techniques efficaces de compression permettant de stocker les graphes sur disque de manière très compacte. Ce format est indépendant du langage source et des outils de vérification. En outre, il contient suffisamment d'informations pour que les outils qui l'exploitent puissent fournir à l'utilisateur des diagnostics précis dans les termes du programme source.

Depuis 1994, un environnement logiciel a été réalisé pour exploiter le format BCG ; il se compose de bibliothèques de fonctions C et de plusieurs outils, notamment : BCG_IO (qui effectue des conversions de format), BCG_OPEN (qui permet d'appliquer à des graphes BCG les outils de l'environnement OPEN/CÆSAR pour la vérification à la volée), BCG_DRAW (qui permet d'afficher en PostScript une représentation 2D d'un graphe), et BCG_EDIT (qui permet de modifier interactivement la représentation graphique produite par BCG_DRAW).

En 1996, les outils CÆSAR et ALDÉBARAN ont été adaptés pour produire et traiter directement des graphes au format BCG.

Le développement de l'outil BCG_XTL s'est poursuivi dans le cadre du travail de thèse de R. Mateescu. 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, ACTL, HML, 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.

En 1996, nous avons achevé la réalisation de la première version du compilateur XTL capable d'évaluer des formules temporelles sur des graphes BCG. Cet outil offre une interface de haut niveau au-dessus du format BCG et ajoute les logiques temporelles aux fonctionnalités de la boîte à outils CADP. Il a été utilisé avec succès dans l'enseignement (magistère UJF) ainsi que pour l'analyse du protocole BRP (cf. section 3.2.2).

Nous orientons à présent nos recherches vers l'extension du μ-calcul arborescent par des variables typées. De telles extensions permettent 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 étudions actuellement des algorithmes d'évaluation pour de telles extensions.

3.4.2 Vérification à la volée

Participant : Hubert Garavel

En 1996, l'environnement OPEN/CÆSAR a été amélioré. Les interfaces de programmation ont été simplifiées et rendues plus indépendantes du langage LOTOS. Les bibliothèques logicielles ont été corrigées et étendues. Enfin, la nouvelle version d'OPEN/CÆSAR permet d'utiliser des composants logiciels fournis soit sous forme de code source (en langage C), soit sous forme de code binaire.

L'outil de simulation interactive XSIMULATOR a été ré-écrit en TCL/TK afin de permettre sa portabilité vers d'autres machines que celles de SUN (travail de M. Jorgensen et H. Garavel).

L'outil de recherche de séquences EXHIBITOR a été entièrement ré-écrit (travail de X. Etchevers et H. Garavel). La nouvelle version de cet outil permet d'utiliser un langage plus expressif (incluant des expressions régulières et des opérateurs logiques) pour décrire les séquences à rechercher. Par ailleurs, elle implémente deux types d'algorithmes (parcours en largeur et en profondeur) et, de ce fait, permet de trouver la plus courte séquence satisfaisant les contraintes imposées. Cette nouvelle version, qui corrige les défauts de la précédente, a été utilisée avec succès par l'université de Liège pour la mise en défaut du protocole EQUICRYPT (cf. section 3.2.2).

3.4.3 Génération compositionnelle

Participants : Jean-Pierre Krimm, Laurent Mounier

Pour éviter le problème de "l'explosion d'états", qui survient lorsque l'on génère directement le modèle représentant un système de processus communicants, une solution consiste à adopter une approche incrémentale : on génère tout d'abord le modèle correspondant à un sous-ensemble du système (certains de ses processus), on réduit alors le modèle partiel ainsi obtenu (modulo une relation d'équivalence appropriée) avant de le composer avec le reste du programme (les processus non encore pris en compte). On obtient ainsi un modèle équivalent à celui correspondant au système initial, mais de taille beaucoup plus réduite.

Cette approche, désignée sous le nom de *génération compositionnelle* avait déjà été expérimentée avec succès dans le passé sur plusieurs exemples de programmes LOTOS. Malheureusement, il arrive fréquemment que la taille des modèles partiels ainsi générés soit encore trop importante, du fait que certaines synchronisations imposées par leur environnement (les autres processus du programme) ne sont pas prises en compte. Une solution à ce problème, proposée par S. Graf et B. Steffen en 1990 pour

l'algèbre CSP, consiste à restreindre à l'aide d'un opérateur de *projection* le modèle associé à un sous-ensemble du programme en utilisant une *interface* exprimant l'ensemble des séquences d'exécutions autorisées par son environnement.

Nous avons adapté cette solution à la composition parallèle de LOTOS en définissant un nouvel opérateur de projection. Cet opérateur admet deux types d'interfaces, soit des interfaces proposées par l'utilisateur, soit dans certains cas calculées automatiquement sur le système initial (DEA et magistère de J.P. Krimm).

Nous avons alors intégré cette méthode de génération compositionnelle dans la boîte à outils CADP en lui ajoutant notamment les deux composants suivants :

- l'outil PROJECTOR permet de restreindre à la volée un système de transitions étiquetées modulo un ensemble de séquences d'exécutions ;
- l'outil DES2AUT permet, à partir d'une expression de composition étendue avec des opérateurs de projection, de générer un *shell-script* UNIX qui enchaîne les différents appels aux outils de CADP (CAESAR, ALDEBARAN et PROJECTOR) pour effectuer une génération compositionnelle.

3.4.4 Génération de séquences de test

Participants : Jean-Claude Fernandez, Alain Kerbrat

Nous avons poursuivi notre activité de recherche autour de la génération automatique de séquences de test. Le test de conformité a pour objectif de démontrer l'adéquation d'une implantation à une spécification (formelle ou non) du protocole. La génération automatique de séquences de test consiste à dériver les séquences de tests à partir de la spécification formelle et éventuellement des objectifs de tests formalisés.

L'outil TGV (*Test Generator based on Verification Technology*) [625] que nous avons implémenté dans l'environnement OPEN/CÆSAR génère, à partir d'une spécification et d'un objectif de test modélisant la propriété à tester (tous deux décrits par des systèmes de transitions étiquetées au format ALDÉBARAN), une séquence de test dont certaines transitions sont décorées par des verdicts et des temporisations.

Cette année, nous avons formalisé les concepts de verdict et de temporisation [627]. Nous avons également étudié des méthodes de génération de tests à la volée [613]. Actuellement, dans le cadre du GIE BULL-Inria, nous implémentons cette génération de tests à la volée pour des spécifications en LOTOS dans l'outil TGV et nous étudions la génération à la volée pour des spécifications en SDL.

Cette recherche est développée en étroite collaboration avec C. Jard, Th. Jéron et C. Viho du projet PAMPA de l'Irisa.

3.4.5 Intégration des outils et interface utilisateur

Participant : Hubert Garavel

L'intégration et la diffusion des différents outils au sein d'un environnement cohérent constitue un travail indispensable. En 1996, nous avons rédigé deux articles de présentation [626, 629] et nous avons ajouté plusieurs exemples de démonstration.

Dans le cadre du projet euro-canadien EUCALYPTUS-2, nous avons entièrement réécrit l'interface utilisateur qui intègre nos outils et ceux des universités de Liège, Montréal et Ottawa (ainsi que d'autres outils développés par le projet PAMPA et à l'Institut National des Télécommunications). Cette nouvelle version de l'interface EUCALYPTUS — développée dans l'environnement TCL/TK (travail de J.M. Frume et H. Garavel) — permet d'accéder à différents outils de manière simple et homogène (cf figure 1).

Enfin, une nouvelle passerelle (baptisée BRIDGE) a été développée, qui permet de ré-exécuter avec le simulateur XELUDO de l'Université d'Ottawa les séquences de diagnostic produites par les outils ALDÉBARAN et OPEN/CÆSAR (travail de Mark Jorgensen).

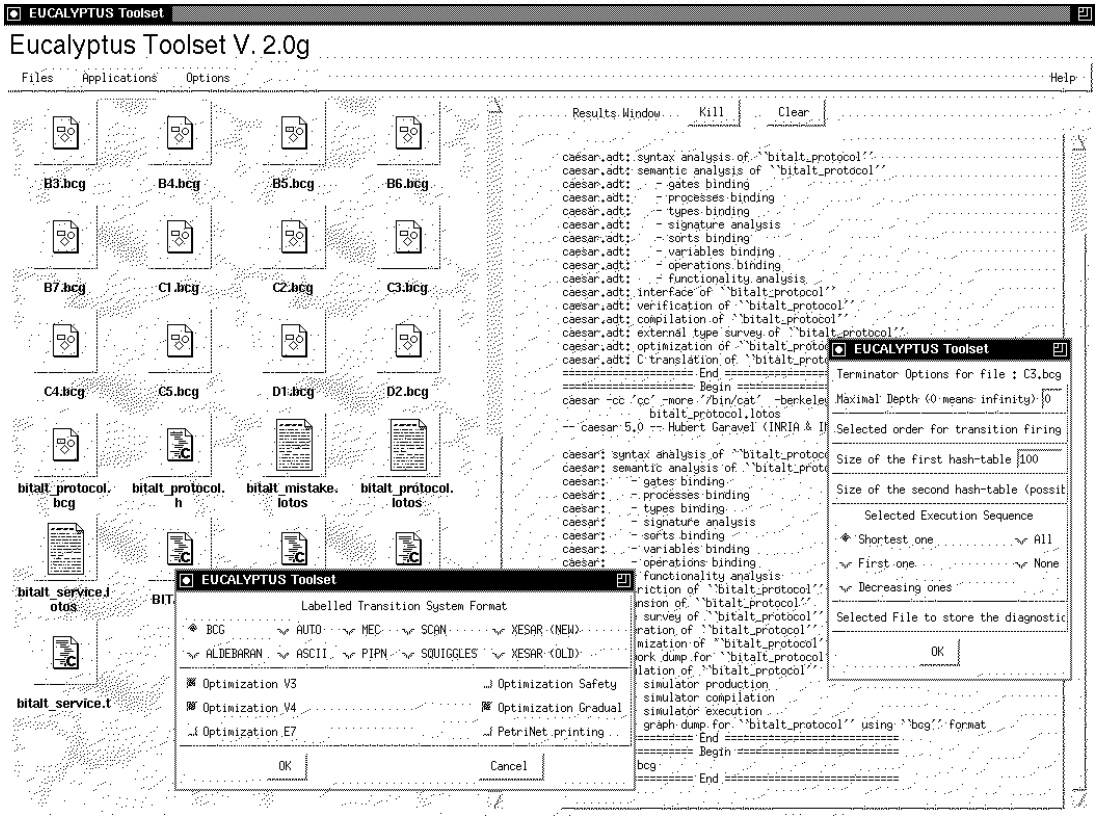

Figure 1: La nouvelle interface utilisateur EUCALYPTUS

3.5 Méthodes de vérification symboliques

Nous rangeons sous cette rubrique les méthodes visant à considérer des ensembles d'états implicitement — par des formules — par opposition aux méthodes énumératives. Les méthodes symboliques sont utilisées

- soit pour améliorer les performances des méthodes de vérification décidables : c'est le cas des méthodes travaillant sur des ensembles d'états représentés par des formules booléennes — codées en BDDs — sur lesquelles nous avons beaucoup travaillé dans le passé (les méthodes exactes d'analyse des systèmes temporisés entrent aussi dans ce cadre),
- soit pour résoudre des problèmes pour lesquels il n'existe pas de méthode exacte. Deux types de méthodes sont explorées dans ce cas :
  - les méthodes approchées qui calculent, par exemple, des approximations supérieures ou inférieures de l'ensemble des états accessibles d'un système.
  - les méthodes assistées par l'utilisateur, et notamment l'utilisation de démonstrateurs de théorèmes.

3.5.1 L'outil de vérification KRONOS

Participants : Conrado Daws, Stavros Tripakis, Sergio Yovine, Joseph Sifakis.

KRONOS est un outil dont le but est d'aider les concepteurs de systèmes temporisés, tels que par exemple les protocoles de communication, les circuits et les systèmes multimédia, à vérifier leur fonctionnement correct de manière automatique.

Les systèmes temporisés sont décrits par des automates, dits automates temporisés, augmentés d'un ensemble d'horloges qui mesurent le temps écoulé. Les comportements de l'automate sont restreints à ceux qui satisfont les contraintes sur les horloges associées aux transitions.

Les critères de correction dont la validation nous intéresse sont les propriétés temporelles quantitatives telles que par exemple l'absence de pannes dans un laps de temps borné et l'offre périodique d'un service particulier. Ces propriétés s'expriment naturellement dans des logiques temporelles temps réel tel que TCTL.

Les algorithmes mis en œuvre dans l'outil KRONOS s'appuient sur une représentation symbolique des ensembles d'états par des contraintes linéaires sur les horloges du système.

Cette année, nous avons conçu une structure de données, appelée "Diagramme de Décision Numérique" pour représenter les polyèdres convexes particuliers qui apparaissent dans la vérification des automates temporisés [643].

Développements Les développements réalisés autour de KRONOS dans la dernière année sont les suivants :

- Dans le cadre du travail de thèse de St. Tripakis, nous avons développé et implanté un algorithme qui, étant donné un automate temporisé, génère son modèle quotient modulo une relation d'équivalence entre les états qui fait abstraction du temps exact écoulé. Cette relation est une bisimulation qui préserve une grande partie des propriétés temporelles que l'on veut vérifier, en particulier celles de sûreté.

  D'une part, les résultats expérimentaux obtenus avec des études de cas significatives montrent que cet algorithme est plus performant que d'autres proposés précédemment, notamment à l'université Stanford. D'autre part, nous avons créé un lien entre KRONOS et la boîte à outils CADP. En effet, étant donné que le modèle quotient généré peut être vu comme un système de transitions étiquetées non-temporisé, il est désormais possible d'utiliser les outils BCG pour le visualiser, ainsi que l'outil ALDÉBARAN pour vérifier des propriétés de sûreté. Ce travail a été présenté à CAV'96 [641].

- Dans le cadre du travail de thèse de C. Daws, nous avons développé et mis en œuvre un algorithme pour réduire le nombre d'horloges d'un automate temporisé. En effet, nous avons constaté que, dans la pratique, un système contient souvent de nombreuses horloges redondantes. Etant donné que la complexité du problème de la vérification de systèmes temporisés est exponentielle dans le nombre d'horloges, il est nécessaire de les éliminer.

  Les sources de redondance sont d'une part la génération automatique d'automates temporisés par les compilateurs de langages de haut niveau et d'autre part la modélisation par composition. En partant de ces faits, nous avons développé un algorithme qui combine deux stratégies complémentaires. La première consiste à détecter des horloges dont les valeurs ne sont pas testées par les contraintes associées aux transitions de l'automate. La deuxième consiste à détecter des paires d'horloges qui sont toujours égales.

  Les résultats expérimentaux obtenus avec des études de cas significatives montrent que la réduction achevée par l'algorithme conduit à des économies d'espace mémoire de presque 50% en moyenne, pouvant aller jusqu'à plus de 90% dans certains cas.

  Ce travail a été présenté à la conférence internationale *Real-Time Systems Symposium* RTSS'96 [624].

Etudes de cas Au cours de cette année, nous avons traité les études de cas suivantes :

- Dans le cadre d'une convention avec le CNET sur la spécification et la vérification de qualités de service de systèmes temporisés, nous avons analysé la spécification SDL du protocole *Fast Reservation Protocol with Delayed Transmission* FRP/DT fournie par les chercheurs du CNET.

– Nous avons expérimenté KRONOS dans le domaine de l'analyse de circuits asynchrones avec des retards. Les résultats obtenus lors de cette première expérience sont très intéressants et ils ont donné lieu à une publication dans la septième *Israeli Conference on Computer Systems and Software Engineering* ICCSSE'96 [633].

– D'autres études de cas ont été présentées dans le colloque international *Hybrid Systems III* [623].

Représentations symboliques Les algorithmes mis en œuvre dans l'outil KRONOS s'appuient sur une représentation symbolique des ensembles d'états par des contraintes linéaires sur les horloges du système.

Cette année nous avons conçu un codage de ces contraintes linéaires par des expressions booléennes. Cette représentation, appelée "Diagramme de Décision Numérique" [643] est en cours d'implantation au-dessus de notre package de BDD.

3.5.2 Analyse d'invariants linéaires

Participants : Nicolas Halbwachs, Yann-Erik Proy

Pour traiter certains programmes numériques simples, nous proposons une technique particulière de l'interprétation abstraite concernant la synthèse d'invariants linéaires. Cette technique consiste à calculer automatiquement, en chaque état du programme, une approximation supérieure de l'ensemble des valeurs possibles des variables numériques, sous forme d'un système de contraintes linéaires. Ces résultats sont utilisés pour démontrer des propriétés de sûreté (inaccessibilité de certains états), exclusion mutuelle entre événements ou même pour choisir les valeurs de certains paramètres numériques.

Cette technique a été implantée dans l'outil POLKA — qui prend en entrée un automate interprété — et a été appliquée à l'analyse des délais dans les programmes synchrones et à la vérification de systèmes temporisés et hybrides.

Le travail a essentiellement porté, cette année, sur l'amélioration de la bibliothèque d'opérations sur les systèmes linéaires (prise en compte d'inégalités strictes, calculs en nombres rationnels,...). Notre bibliothèque est notamment utilisée aux universités de Cornell (T. Henzinger) et Stanford (Z. Manna).

Ce thème constitue le travail de thèse qu'effectue Y.-E. Proy dans le cadre d'un contrat CIFRE avec Schneider-Electric.

3.5.3 Utilisation de démonstrateurs de théorèmes

Participants : Saddek Bensalem, Susanne Graf, Catherine Parent, Yassine Lakhnech, Hassen Saïdi

Les démonstrateurs de théorèmes offrent un cadre très général pour la vérification de programmes. Ils permettent en particulier la formalisation de la démonstration sur la structure du programme et de ses données.

Notre travail vise à utiliser un démonstrateur de théorèmes dans un but de généralisation de méthodes par traduction dans un modèle. Le démonstrateur choisi est PVS, développé par le SRI. Son langage d'entrée est suffisamment expressif pour nos besoins et il contient des procédures de décision pour certaines théories décidables qui peuvent être combinées à l'aide de stratégies relativement simples à définir.

Techniques de calcul d'invariants : pour prouver que tout état accessible d'un programme S satisfait une assertion P, il est nécessaire et suffisant de trouver une assertion Q, plus forte que P, qui soit préservée par toute instruction du programme.

Les techniques itératives de calcul de Q par approximations successives utilisées dans le cas des systèmes finis, souffrent en général de limitations dues au fait que la vitesse de convergence des itérations peut être faible et à l'absence de méthodes de simplification efficaces pour les théories non décidables.

Nous avons développé des techniques d'accélération du processus d'approximation qui utilisent des invariants extraits automatiquement du programme par interprétation abstraite [617].

Ces techniques ont été implantées dans PVS. Nous avons montré que nos techniques sont plus puissantes que d'autres déjà utilisées par exemple dans le démonstrateur STeP[3].

En plus de ces techniques, nous avons développé une méthode pour la génération d'un invariant pour un programme parallèle de la forme $S = S_1 \parallel \cdots \parallel S_n$. La méthode combine des invariants générés séparément pour chaque système S_i en un invariant de S. L'efficacité de ces techniques a été montrée sur plusieurs exemples non triviaux, comme l'algorithme d'exclusion mutuelle de Bakery.

Un outil de vérification basé sur PVS**:** nous avons réalisé une interface permettant de manipuler les programmes dans une syntaxe proche de celle des systèmes de transitions [630]. La propriété à vérifier — dans un premier temps un invariant — est exprimée dans le langage de spécification de PVS.

Étant donné un programme S et une assertion P dont on doit prouver l'invariance, l'interface génère un ensemble de formules dans le langage de spécification de PVS (obligations de preuve) qui tiennent compte des invariants déjà connus générés par les techniques décrites ci-dessus. Ces obligations de preuve sont soumises au démonstrateur de PVS. Dans le cas où ces obligations se trouvent dans un fragment décidable du langage d'assertions, leur validité peut être établie automatiquement grâce à des stratégies que nous avons écrites. Ces stratégies font essentiellement appel à des règles de réécriture et aux algorithmes de décision pour le calcul propositionnel et l'arithmétique de Presburger implémentés en PVS. Si P n'est pas préservé, l'interface propose des approximations par calcul itératif.

3.5.4 Diagrammes de Décision Multivalués

Participants : Marius Bozga, Jean-Claude Fernandez, Alain Kerbrat

Les diagrammes de décision binaire (BDDs) constituent un apport décisif pour la vérification de circuits. Dans le cas des systèmes distribués, les BDDs n'ont pas véritablement percé, même s'ils permettent sur certains exemples d'apporter une solution là où les méthodes énumératives échouent. Une raison majeure de ce manque relatif de succès est l'utilisation fréquente par les protocoles de communication de variables non booléennes, qui ne sont pas représentables directement par les BDDs. Nous avons étudié certaines variantes des diagrammes de décision, notamment des diagrammes autorisant des décisions n-aires, et des diagrammes utilisant une notion de décomposition différente de celle de Shannon, qui est à la base des BDDs. Sur la base de cette étude, nous avons défini un type de diagramme mieux adapté à la représentation des variables numériques bornées, les Diagrammes de Décision Binaires Multivalués (BMDDs) (DEA de M. Bozga). Ces diagrammes conservent un branchement binaire pour des raisons d'efficacité d'implémentation, mais utilisent une notion originale de décomposition.

Nous avons intégré ces BMDDs dans la boîte à outils CADP, en lui ajoutant notamment le module SMI (Symbolic Model Interface). Ce module permet la gestion d'un modèle représenté par plusieurs types de diagrammes de décision. Il est actuellement interfacé à trois bibliothèques de BDDs différentes ainsi qu'à notre bibliothèque BMDDs. SMI prend en entrée des automates étendus communicants, dont la syntaxe et la sémantique ont été définis par la même occasion.

Enfin, deux outils existants ont été interfacés à SMI : un outil de génération de modèle minimal permettant la minimisation d'un modèle pour une relation de bisimulation donnée, et un outil d'eévaluation de formules du μ-calcul.

[3] STeP a été développé par l'équipe de Z. Manna à Stanford

3.6 Classes décidables de systèmes infinis

Participants : Ahmed Bouajjani, Peter Habermehl

La motivation de ce travail est la vérification des systèmes à espaces d'états infinis. Des modèles des systèmes infinis peuvent être obtenus en étendant les modèles réguliers :

- soit par adjonction de types de données infinies. On peut étendre ainsi les automates pour obtenir les systèmes hybrides (automates à variables réelles), les automates temporisés (automates à horloges), les automates à compteurs, à files, etc.
- soit par adjonction d'opérateurs ou de constructions engendrant des comportements non réguliers. On peut étendre ainsi les algèbres de processus réguliers en autorisant la récursion à travers la composition séquentielle (pour définir des processus hors-contexte) ou parallèle.

L'objectif du travail est l'identification de classes de systèmes et de propriétés pour lesquelles des méthodes de décision existent. Nous nous intéressons à la vérification de propriétés de systèmes infinis décrites dans des formalismes de spécification basés sur des logiques temporelles. Les résultats positifs qui existent sur ce sujet concernent l'extension aux processus hors-contexte et aux automates à pile des techniques de vérification pour des propriétés exprimables dans un fragment du μ-calcul propositionnel arborescent. Ces propriétés sont des propriétés *régulières*, c'est-à-dire que l'on peut caractériser par des automates *finis*. Cependant, les propriétés qui portent sur les valeurs de variables non bornées ne sont pas régulières de manière générale. En particulier, sont non-régulières des propriétés portant sur le nombre d'occurrences de certains événements dans une séquence d'exécution. Par exemple, des propriétés d'un protocole telles que *entre le début et la fin d'une session, il y a autant d'envois de messages que d'accusés de réception*.

Pour pouvoir exprimer de telles propriétés, nous avons défini l'année précédente de nouvelles logiques qui sont des logiques temporelles où des contraintes sur des nombres d'occurrences d'événements sont décrites à l'aide de formules de l'arithmétique de Presburger. Ainsi, le problème que nous considérons est celui de la vérification de propriétés *non régulières* pour des systèmes *non réguliers*.

Dans [618], nous étudions le problème de la vérification pour la logique CLTL (Constrained LTL) pour deux classes de systèmes infinis: les réseaux de Petri (non bornés) et les systèmes *semi-linéaires* qui englobent les automates à pile et les processus PA (récursion avec composition séquentielle et parallèle). Nous montrons en particulier que le problème de la vérification d'un fragment de CLTL strictement plus expressif que le μ-calcul linéaire (et les ω-automates de Büchi), est décidable aussi bien pour les automates à pile que pour les réseaux de Petri.

4 Actions industrielles

Coopération avec VÉRILOG : Cette coopération est évidemment très active dans le cadre de l'Unité Mixte VÉRIMAG.

- Nous assistons quotidiennement VÉRILOG qui développe à partir de Lustre l'atelier SAGA/SAO+.

  Cet atelier a été développé en collaboration avec Schneider-Electric et Aérospatiale, qui ont signé cette année avec VÉRILOG un accord de 10 ans assurant la pérennité du produit. Celui-ci connaît un succès remarquable : utilisation dans des projets opérationnels par Schneider-Electric, Aérospatiale, CS-Transport (réingéniérie du métro de Hong-Kong), Volvo; expérimentation par SFIM, Honeywell, Bœing, Eurocopter-Deutschland, SAAB Military Aircraft, Framatome.

  Notre assistance porte sur l'évolution du langage, la génération automatique de code, et la vérification de propriétés des programmes. Nous coopérons également étroitement avec VÉRILOG au sein du projet EUREKA-SYNCHRON, pour la conception des formats communs des langages synchrones.

– L'étude d'un an soutenue par la DGA/STEI (avec comme partenaires VÉRILOG, Cap Sesa Régions, le CNET, l'Inria (projets PAMPA et SPECTRE)) dont l'objectif était de montrer la faisabilité d'une utilisation industrielle des méthodes formelles pour générer automatiquement des séquences de test s'est terminée en décembre 95.

  Le prototype TGV, réalisé par l'Inria (projets PAMPA et SPECTRE), a permis de montrer l'intérêt d'utiliser les méthodes développées dans le cadre de la vérification, pour générer des séquences de test.

  D'autre part, cette étude a montré que la génération automatique de séquences de test apportait un gain en qualité des tests générés et en productivité. Une proposition d'industrialisation des résultats, par VÉRILOG pour le compte du CNET est en cours de négociation. Les projets PAMPA et SPECTRE apparaîtraient comme sous-traitants de VÉRILOG.

Coopération avec Schneider-Electric : cette coopération comporte plusieurs volets. Dans le cadre du projet EUREKA-SYNCHRON, Schneider-Electric joue un rôle d'expérimentateur des formats communs des langages synchrones et des outils associés. Nous avons par ailleurs joué un rôle de conseil dans le traitement d'étude de cas en vérification de programmes SAGA. Schneider-Electric nous a aussi chargé d'écrire la sémantique formelle du langage SAGA. Enfin, nos travaux concernant l'outil POLKA de vérification approchée de systèmes hybrides, sont effectués dans le cadre d'une convention CIFRE avec Schneider-Electric.

Coopération avec Bull : au sein du GIE BULL/Inria DYADE, nous sommes fortement impliqués dans l'action VASY (*Validation de Systèmes*) qui a pour objectif l'utilisation des méthodes formelles (langage LOTOS et boîte à outils CADP) pour la validation et la vérification des architectures multiprocesseurs développées par Bull. Cette collaboration, qui s'est renforcée par l'installation de Gh. Chehaibar (BULL) dans les locaux de l'UR Rhône-Alpes, comporte deux volets :

– nous avons d'abord pu vérifier la correction de l'arbitre de bus de l'architecture POWERSCALE mise en œuvre dans les stations de travail et serveurs de la gamme ESCALA [622] ;

– nous étudions actuellement le protocole de cohérence de caches défini pour l'architecture POLYKID des futures gammes de machines Bull.

Coopération avec Matra Marconi Space : Notre collaboration de recherche Post-Doc Entreprise avec Matra Marconi Space dans le cadre du post-doctorat de E. Conquet a pris fin au mois de mars.

Par ailleurs, nous avons fini le projet que nous avons réalisé en collaboration avec les sociétés Matra Marconi Space et Dornier pour l'ESA, l'agence spatiale européenne (VÉRIMAG était sous-contractant de Matra Marconi Space). Le but principal de cette coopération était l'étude de la pertinence de l'utilisation des méthodes formelles dans la conception des systèmes embarqués très critiques. Nos activités principales ont été :

– la sélection de la méthode (et de l'outil correspondant) la plus appropriée à la réalisation du système cible ;

– pendant la phase de spécification, l'assistance aux ingénieurs de Matra Marconi Space pour l'utilisation de la méthode et de l'outil choisi ;

– la vérification des spécifications fournies par Matra Marconi Space.

Les résultats obtenus ont été très encourageants. Un colloque est prévu à l'ESA au mois de décembre 1996 pour les présenter.

Coopération avec le CNET **:** nous travaillons, dans le cadre de la convention CNET-Inria #97 B, sur la spécification et l'analyse de qualités de service de systèmes temporisés, notamment multimédia. Nous avons analysé une version du protocole *Fast Reservation Protocol with Delayed Transmission* FRP/DT (voir section 3.5.1).

Une autre convention avec le CNET, en coopération avec le projet MEIJE de l'Inria (Sophia-Antipolis) concerne, entre autres, la vérification booléenne et numérique des programmes synchrones.

Coopération avec EDF **:** nous étudions l'utilisation de LUSTRE et LÉSAR pour la spécification et la vérification de systèmes de contrôle de centrale, développés par EDF. En particulier, nous sommes amenés à nous intéresser à la formalisation des activités de test de logiciel, notamment lorsqu'une spécification synchrone est implantée sur une architecture asynchrone.

Expertise pour la RATP **:** la RATP nous a demandé d'expertiser l'architecture à haute sécurité proposée par MATRA Transport pour la nouvelle ligne de métro automatique METEOR. Cette étude, commencée fin 1994, s'effectue dans le cadre d'une convention d'assistance technique entre la RATP et l'université Joseph Fourier et doit se poursuivre jusqu'en 1997.

Collaboration avec Hewlett-Packard : une collaboration avec Hewlett-Packard s'est déroulée cette année sur l'intégration des méthodes formelles dans le cycle de développement de logiciels de télécommunications.

Il s'agit, en particulier, d'évaluer l'adéquation des outils industriels ou universitaires liés à SDL pour la description et la validation de services de télécommunications et de leurs interactions. Cette évaluation s'est faite autour d'une étude de cas fournie par HP, le protocole ISDN User Part ISUP qui est un protocole standard des services de signalisation des réseaux téléphoniques. Cette évaluation a débouché sur un rapport [649] décrivant l'utilisation conjointe de l'outil industriel OBJECT-GEODE et de la boîte à outils CÆSAR-ALDÉBARAN.

La collaboration se poursuit notamment par l'étude des modalités pratiques pour l'intégration d'outils formels dans les outils existants chez HP pour le développement de services de télécommunications.

5 Actions nationales et internationales

5.1 Actions nationales

GDR-PRS : nous participons à son comité scientifique et nous animons le thème "Description formelle et vérification". Son objectif est, d'une part de développer des modèles, de nouvelles méthodes et algorithmes de vérification et, d'autre part, de renforcer les liens entre les différents outils déjà développés par les partenaires. Les extensions des modèles concernent la sémantique du parallélisme, l'introduction du temps et le raffinement d'actions.

Groupe de travail C2A : nous participons à l'action coopérative de recherche C2A (CAO-Automatique), dont la composante académique constitue un pôle commun des GDR "PRS" et "Automatique". Ce groupe organise trois réunions d'un jour et demi par an.

5.2 Actions internationales

Groupe de travail ISO/JTC1/SC21/WG7: nous participons à l'action de normalisation intitulée "*Enhancements to* LOTOS" dans le cadre du groupe de travail de l'ISO consacré aux systèmes distribués ouverts (ODP).

5.2.1 Europe de l'ouest

Projets ESPRIT-LTR SYRF : nous avons coordonné la proposition de ce projet qui a été accepté : le projet devrait démarrer début 97, et réunit les trois projets Inria travaillant sur les langages synchrones, la GMD (A. Poigné, Allemagne), l'université de Linköping (S. Nadjm-Tehrani, Suède), et les sociétés Logikkonsult (Suède), SAAB-MA(Suède), Schneider-Electric et EDF.

Action européenne COST 247 : nous participons à ce projet sur la vérification et la validation de descriptions formelles ; nous coordonnons le groupe de travail consacré à LOTOS et E-LOTOS.

Projet Euro-Canadien Eucalyptus-2 : depuis janvier 1995, nous coordonnons Eucalyptus-2, projet de coopération entre l'Union Européenne et le Canada, ayant pour objectif le développement concerté d'outils pour la vérification des spécifications LOTOS.

Projet EUREKA SYNCHRON : nous participons à ce projet pour le développement d'un environnement multi-langages pour la programmation synchrone. Deux réunions ont eu lieu cette année pour ce projet.
Partenaires : TNI, VÉRILOG, LOGIKKONSULT (Suède), SCHNEIDER ELECTRIC, SAAB (Suède), VTT (Finlande), SNECMA, GMD (Allemagne), Inria.

5.2.2 Europe de l'est

Projet INTAS-94-0697 : nous sommes les coordinateurs de ce projet sur les systèmes hybrides. Les participants sont VÉRIMAG, IPPI - Académie des Sciences de Moscou et l'université de Padoue.

5.2.3 Amérique

Projet Esprit-NSF Hybrid EC-US-043 : nous sommes les coordinateurs de ce projet qui regroupe des équipes européennes et nord-américaines travaillant sur le thème des systèmes hybrides. Participants: VÉRIMAG, Irisa, universités de Lund, Lyngby, Cornell, Notre-Dame, Stanford, Chicago, et MIT.

Coopération avec le projet PATH : dans le cadre d'un projet de coopération "Inria-NSF", nous participons au projet américain PATH pour le développement d'un système d'autoroutes automatiques. Nous contribuons au développement d'un environnement pour la spécification et la vérification de systèmes hybrides complexes.

Projet KIT # 139 HYBSYS : ce projet s'inscrit dans un programme européen de coopération scientifique avec des pays en voie de développement. Les partenaires sont : VÉRIMAG (coordinateur), l'institut FORTH (Héraklion, Grèce) et l'Instituto de Computación (Montevideo, Uruguay). Le thème du projet porte sur l'extension de l'outil KRONOS à l'analyse de systèmes hybrides.

6 Diffusion des résultats

6.1 Diffusion de logiciels

La diffusion de la boîte à outils CADP (regroupant les outils de vérification ALDÉBARAN, BCG, CÆSAR, CÆSAR.ADT et OPEN/CÆSAR) s'est poursuivie. En 1996, 9 nouveaux sites ont demandé une mise à disposition, ce qui porte le nombre total de sites à 125.

Nous avons effectué de nombreuses démonstrations publiques de nos outils de vérification, notamment à l'occasion de TACAS'96 (Passau, mars 1996), du colloque COST-247 (Maribor, juin 1996), CAV'96 (New Brunswick, août 1996), FORTE/PSTV'96 (Kaiserslautern, octobre 1996).

Les logiciels suivants sont distribués par FTP public :

- ARGONAUTE : environnement de programmation et vérification de systèmes réactifs, multi-langages et multi-outils.
- KRONOS : outil de vérification de systèmes temporisés par évaluation symbolique : comparaison d'automates temporisés à des propriétés exprimées en TCTL.
- LÉSAR : outil de vérification de propriétés de programmes écrits en LUSTRE.
- POLKA : outil d'analyse des propriétés numériques des programmes, fondé sur une interprétation abstraite à base de polyèdres convexes.
- BAC : outil de vérification d'automates booléens à base de BDDs.
- MAGEL : outil de réduction à l'aide de BDD et d'analyse à l'aide de polyèdres convexes s'appliquant aux réseaux de Petri produits à partir de programmes LOTOS.

6.2 Animation scientifiques

Organisation de Colloques : nous avons organisé cette année le colloque FORMA.

Comités de programme : nous participons au *steering committee* de la conférence "Computer Aided Verification" (CAV) et aux comités de programme des conférences CAV, AMAST, RTSS (IEEE), TACAS, SAS et de l'école MOVEP.

H. Garavel est membre du comité de rédaction de la revue "*Technique et Science Informatique*".

J. Sifakis participe au comité de lecture du journal "*Formal Methods in System Design*".

Visites et chercheurs invités :

- Dans le cadre du Projet KIT # 139 HYBSYS, nous avons accueilli Alfredo Olivero (1 mois) et Luis Sierra (2 mois) de l'université de Montevideo.
- Dans le cadre du Projet Inria-NSF, nous avons accueilli pendant une semaine Anuj Puri et Akash Deshpande de l'université de Californie à Berkeley.
- Yassine Lakhnech de l'université de Kiel (Allemagne) a séjourné 3 mois dans notre laboratoire.
- Pendant son année sabbatique, Saddek Bensalem a passé 9 mois au SRI à Stanford.

6.3 Actions d'enseignement

Tous les membres permanents du projet, et en particulier les enseignants-chercheurs des établissements d'enseignement supérieur de Grenoble, participent activement à leurs formations.

Nous donnons des cours spécifiquement dédiés aux thèmes développés dans le projet, dans le cadre du DEA de Grenoble.

Par ailleurs, N. Halbwachs a assuré dans le cadre du DEA IMA commun à l'Ecole Normale Supérieure, l'Ecole Polytechnique, et les universités Paris 6, 7 et 11 un cours sur les méthodes de vérification de systèmes finis.

6.4 Séminaires et formations permanentes

Les membres du projet ont participé à de nombreux séminaires de recherche, conférences invitées, ainsi qu'aux actions de formations permanentes et écoles d'été suivantes :

Séminaire Greco Informatique : H. Garavel a assuré (conjointement avec Didier Bert, du laboratoire LSR) un séminaire intitulé "*Génie logiciel : spécification et validation des logiciels critiques*" dans le cadre des journées du Greco Informatique (Transfert Industriel) à Paris du 6 au 7 juin 1996.

Asian School on Computer Science : N. Halbwachs assure un cours dans le cadre de cette école qui a lieu du 22 au 29 novembre 1996 à Rayong (Thailande) et qui est co-organisée par le *Asian Institute of Technology* et l'Inria.

Journées Uruguayennes d'Informatique 1995 S. Yovine a assuré un cours d'une semaine sur la vérification symbolique de systèmes temporisés.

DIMACS 1995-1996 Special Year on Logics and Algorithms S. Yovine a participé en tant que chercheur invité. Il a assuré un séminaire intitulé "*Symbolic Model-Checking for Timed Automata and* TCTL*: The tool* KRONOS".

7 Publications

Articles et chapitres de livre

[613] J.-C. FERNANDEZ, C. JARD, T. JÉRON, L. NEDELKA, C. VIHO, «An Experiment in Automatic Generation of Test Suites for Protocols with Verification Technology», *Science of Computer Programming*, 1996, Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Also available as Inria Research Report RR-2923.

[614] H. GARAVEL, L. MOUNIER, «Specification and Verification of various Distributed Leader Election Algorithms for Unidirectional Ring Networks», *Science of Computer Programming*, 1996, Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Full version available as Inria Research Report RR-2986.

[615] S. GRAF, G. LÜTTGEN, B. STEFFEN, «Compositional Minimisation of Finite State Systems using Interface Specifications», *Formal Aspects of Computation 3*, 1996, appeared as Passauer Informatik Bericht MIP-9505.

Communications à des congrès, colloques, etc.

[616] B. CAILLAUD, P. CASPI, A. GIRAULT ET C. JARD, «Un modèle pour la distribution d'automates réactifs sur réseau asynchrone de processeurs», *in : Conférence Afcet sur la Modélisation des systèmes réactifs*, Brest, mars 1996.

[617] S. BENSALEM, Y. LAKHNECH, H. SAIDI, «Powerful Techniques for the Automatic Generation of Invariants», *in : Conference on Computer Aided Verification CAV'96*, *LNCS 1102*, juillet 1996.

[618] A. BOUAJJANI, P. HABERMEHL, «Constrained Properties, Semilinear Systems, and Petri Nets», *in : Proc. Intern. Conf. on Concurrency Theory (CONCUR'96)*, LNCS 1119, 1996.

[619] A. BOUAJJANI, Y. LAKHNECH, S. YOVINE, «Model Checking for Extended Timed Temporal Logics», *in : Proc. Intern. Symp. on Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'96)*, LNCS 1135, 1996.

[620] A. BOUAJJANI, Y. LAKHNECH, «Logics vs. Automata: The Hybrid Case», *in : Hybrid Systems III : Verification and Control*, LNCS 1066, 1996.

[621] P. CASPI, M. POUZET, «Synchronous Kahn networks», *in : Int. Conf. on Functional Programming, Philadelphia*, ACM SIGPLAN, mai 1996.

[622] 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 (réd.), IFIP, Chapman & Hall, p. 435–450, octobre 1996. Full version available as Inria Research Report RR-2958.

[623] C. DAWS, A. OLIVERO, S. TRIPAKIS, S. YOVINE, «The tool KRONOS», *in : Hybrid Systems III, Verification and Control*, *Lecture Notes in Computer Science 1066*, Springer Verlag, p. 208–219, 1996.

[624] C. DAWS, S. YOVINE, «Reducing the number of clock variables of timed automata», *in : Proc. 1996 IEEE Real-Time Systems Symposium, RTSS'96*, IEEE Computer Society Press, Washington, DC, USA, décembre 1996.

[625] L. DOLDI, V. ENCONTRE, J.-C. FERNANDEZ, T. JÉRON, S. L. BRICQUIR, N. TEXIER, M. PHALIPPPOU, «Assessment of automatic generation of conformance test suites in an industrial context», *in : International Workshop on Testing of Communicating Systems, IWTCS'96*, 1996.

[626] 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 (réd.), *LNCS*, *1102*, Springer Verlag, p. 437–440, août 1996.

[627] J.-C. FERNANDEZ, C. JARD, T. JÉRON, L. NEDELKA, C. VIHO, «Using On-the-Fly Verification Techniques for the Generation of Test Suites», *in : Proceedings of the 8th International Conference on Computer-Aided Verification (Rutgers University, New Brunswick, NJ, USA)*, R. Alur, T. A. Henzinger (réd.), *LNCS*, *1102*, Springer Verlag, p. 348–359, août 1996. Also available as Inria Research Report RR-2987.

[628] H. GARAVEL, M. SIGHIREANU, «On the Introduction of Exceptions in 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 (réd.), IFIP, Chapman & Hall, p. 469–484, octobre 1996.

[629] 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 (réd.), University of Maribor, Slovenia, juin 1996.

[630] S. GRAF, H. SAIDI, «Verifying invariants using theorem proving», *in : Conference on Computer Aided Verification CAV'96*, *LNCS 1102*, juillet 1996.

[631] M. JOURDAN, F. MARANINCHI, «Vérification de systèmes réactifs en Argos temporisé», *in : Congrès AFCET : Modélisation des systèmes réactifs*, Brest (France), mars 1996.

[632] D. LESENS, N. HALBWACHS, P. RAYMOND, «Automatic Construction of Network Invariants», *in : International Workshop on Verification of Infinite State Systems (INFINITY)*, Pisa, août 1996.

[633] O. MALER, S. YOVINE, «Hardware Timing Verification using KRONOS», *in : Proceedings of 7th Conf. on Computer-based Systems and Software Engineering, Herzeliya, Israel*, IEEE Press, 1996.

[634] F. MARANINCHI, N. HALBWACHS, «Compiling Argos into Boolean equations», *in : Formal Techniques for Real-Time and Fault Tolerance (FTRTFT)*, LNCS 1135, Springer Verlag, Uppsala (Sweden), septembre 1996.

[635] F. MARANINCHI, N. HALBWACHS, «Compositional Semantics of Non-deterministic Synchronous Languages», *in : European Symposium On Programming, Linköping (Sweden)*, LNCS 1058, Springer Verlag, avril 1996.

[636] R. MATEESCU, «Formal Description and Analysis of a Bounded Retransmission Protocol», *in : Proceedings of the COST 247 International Workshop on Applied Formal Methods in System Design (Maribor, Slovenia)*, Z. Brezočnik, T. Kapus (réd.), University of Maribor, Slovenia, juin 1996. Also available as Inria Research Report RR-2965.

[637] P. CASPI ET M. POUZET, «Réseaux de Kahn synchrones», *in : Journées Francophones des langages applicatifs, Val Morin, Quebec*, Inria, janvier 1996.

[638] P. RAYMOND, «Recognizing Regular Expressions by means of Dataflows Networks», *in : 23rd International Colloquium on Automata, Languages, and Programming, (ICALP'96)* Paderborn, Germany, LNCS 1099, Springer Verlag, juillet 1996.

[639] H. SAIDI, « A tool for proving invariance properties of concurrent systems automatically », *in : Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'96*, *LNCS 1055*, mars 1996.

[640] J. SIFAKIS, S. YOVINE, « Compositional specification of timed systems », *in : 13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96, Grenoble*, *Lecture Notes in Computer Science 1046*, Springer Verlag, p. 347–359, février 1996.

[641] S. TRIPAKIS, S. YOVINE, « Analysis of timed systems based on time–abstracting bisimulations », *in : Proc. 8th Conference Computer-Aided Verification, CAV'96*, *Lecture Notes in Computer Science 1102*, Springer Verlag, p. 232–243, Rutgers, NJ, juillet 1996.

Rapports de recherche et publications internes

[642] E. ASARIN, O. MALER, P. CASPI, « A Kleene Theorem for Timed Automata », *Rapport Technique* SPECTRE *n°96-08*, VÉRIMAG, 1996, submitted for publication.

[643] E. ASARIN, O. MALER, A. PNUELI, « Data-Structures for the Verification of Timed Automata », *Rapport Technique* SPECTRE *n°96-07*, VÉRIMAG, 1996, submitted for publication.

[644] M. BOZGA, *Vérification formelle de systèmes distribués*, Mémoire de DEA , Université Joseph Fourier, Grenoble, juin 1996.

[645] H. GARAVEL, R. MATEESCU, « French-Romanian Proposal for Capture of Requirements and Expression of Properties in E-LOTOS Modules », *Rapport Technique* SPECTRE *n°96-04*, VÉRIMAG, mai 1996, Input document [KC4] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Kansas City, Missouri, USA, May, 12–21, 1996.

[646] H. GARAVEL, M. SIGHIREANU, « French-Romanian Comments regarding some Proposed Features for E-LOTOS Data Types », *Rapport Technique* SPECTRE *n°95-19*, VÉRIMAG, décembre 1995, Input document [LG3] of the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Liège (Belgium), December, 18–21, 1995.

[647] H. GARAVEL, M. SIGHIREANU, « French-Romanian Integrated Proposal for the User Language of E-LOTOS », *Rapport Technique* SPECTRE *n°96-05*, VÉRIMAG, mai 1996, Input document [KC3] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Kansas City, Missouri, USA, May, 12–21, 1996.

[648] H. GARAVEL, « A Wish List for the Behavioural Part of E-LOTOS », *Rapport Technique* SPECTRE *n°95-21*, VÉRIMAG, décembre 1995, Input document [LG5] of the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Liège (Belgium), December, 18–21, 1995.

[649] A. KERBRAT, D. PENKLER, N. RAGUIDEAU, « Using Aldébaran and Object-Géode for the development of telecommunications services », *rapport de recherche*, Hewlett Packard/VÉRIMAG, 1996.

[650] M. SIGHIREANU, H. GARAVEL, « Application of the Proposed E-LOTOS Datatype Language to the Description of OSI and ODP Standards », *Rapport Technique* SPECTRE *n°95-17*, VÉRIMAG, novembre 1995, Input document [LG2] of the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Liège (Belgium), December, 18–21, 1995.

[651] M. SIGHIREANU, H. GARAVEL, « Defect Report concerning the LOTOS Description of OSI TP Protocol », *Rapport Technique* SPECTRE *n°95-18*, VÉRIMAG, novembre 1995, AFNOR CGTI/CN 21 F 2503.

[652] M. SIGHIREANU, H. GARAVEL, « A Proposal for the Data Type Part of E-LOTOS Applicable to the Formal Description of OSI and ODP Standards », *Rapport Technique* SPECTRE *n°95-20*, VÉRIMAG, décembre 1995, Input document [LG4] of the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Liège (Belgium), December, 18–21, 1995.

[653] M. SIGHIREANU, H. GARAVEL, « E-LOTOS User Language », *Rapport Technique* SPECTRE *n°96-06*, VÉRIMAG, octobre 1996, In ISO/IEC JTC1/SC21 Third Working Draft on Enhancements to LOTOS (1.21.20.2.3). Output document of the edition meeting, Kansas City, Missouri, USA, May, 12–21, 1996.

[654] B. VIVIEN, « Etude du système Syntax/Fnc-2 pour la génération de compilateurs », *Mémoire de probatoire en informatique*, CNAM, Grenoble, juin 1996.

8 Abstract

The SPECTRE project aims at aiding designers of concurrent and real-time systems, for which the use of formal methods appears both necessary and reasonably feasible. Methods and tools are provided for the main design tasks: specification, programming and validation. The SPECTRE project is involved in research in the following three directions:

- Design and use of specification languages for concurrent and real-time systems. Declarative languages (temporal logics) are particularly considered, together with their connections with imperative formalisms (process algebras and transition system based languages).
- Design and/or implementation of programming languages for concurrent and real-time systems. The semantics of such languages and the associated compilation techniques are studied.
- Definition and implementation of validation methods, well-suited for these languages. In the considered application domains (communication protocols, real-time control systems and hardware), automated validation seems achievable. The problem of error identification and correction is also addressed.

One goal of the work is to confront theoretical results to practical problems raised by actual system design. It is intended to produce, on one hand, prototype tools open to industrialization, and on the other hand, practical design methods.

The project consists of several coordinated research actions oriented to specific applications — reactive systems, communication protocols, timed systems — and one transversal action providing the basic technology for validation.

SPECTRE is a project of VÉRIMAG a joint laboratory of CNRS, Institut National Polytechnique de Grenoble, Université Joseph Fourier and VÉRILOG SA. The objectives of VÉRIMAG include the transfer of research results of the project.

SPECTRE participates in several international projects: the EC-Canada Eucalyptus project for the development of protocol validation tools, the ESPRIT-EUREKA project SYNCHRON and the ESPRIT-LTR project SYRF on the development of synchronous languages, the Inria-NSF project PATH on Automatic Highways and an ESPRIT-NSF project on hybrid systems.