INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet PARA

Parallélisme

Rocquencourt

THÈME 1C

Table des matières

Le début de 1999 a vu le départ d'Alain Deutsch dans la création de la start-up PolySpace Technologies, qui va développer et commercialiser la technologie d'analyse statique de programmes par interprétation abstraite qui avait été développée au sein du projet au cours de son intervention sur le logiciel du lanceur Ariane 502. Le projet Para va changer de responsable scientifique et de nom pour l'an 2000 ; il devient le projet Moscova (pour MObilité, Sécurité, COncurrence, Vérification et Analyse), sous la responsabilité de Georges Gonthier.

1 Composition de l'équipe

Responsable scientifique

Jean-Jacques Lévy [DR Inria]

Responsable permanent

Georges Gonthier [DR Inria]

Assistante de projet

Sylvie Loubressac [TR]

Personnel Inria

Alain Deutsch [CR, jusqu'au 1/2/1999]

Damien Doligez [CR]

Luc Maranget [CR]

Conseillère scientifique

Thérèse Hardin [Professeur d'université Paris VI]

Collaborateurs extérieurs

Alain Deutsch [PolySpace Technologies, depuis le 1/2/1999]

Thérèse Hardin [en délégation de Paris VI, depuis le 1/10/1999]

Doctorants

Bruno Blanchet [moniteur normalien, ENS]

Sylvain Conchon [allocataire MENRT]

Fabrice le Fessant [moniteur X, Ecole Polytechnique]

Alan Schmitt [boursier DGA, depuis le 1/10/1999]

Stagiaires

Alan Schmitt [Ecole Polytechnique]

Nicolas Tessaud [Ecole Polytechnique]

2 Présentation et objectifs généraux

Le projet Para s'intéresse à la programmation concurrente sous de multiples formes. La programmation de plusieurs processus concurrents est délicate. Elle demande de bien comprendre le modèle sous-jacent et de disposer de primitives rigoureusement définies. Notre projet ne traite ni du parallélisme fortement synchrone, ni de l'algorithmique pour des machines SIMD à parallélisme massif. Il ne traite pas non plus de l'algorithmique distribuée. Notre objectif est de construire des systèmes pour programmer des applications faiblement synchrones sur des architectures distribuées. Dans ce cadre, notre effort se dirige actuellement vers la programmation des processus mobiles qui permettent de tenir compte de la reconfiguration dynamique des interconnexions.

En effet, avec le développement des réseaux et des applications distribuées à grande échelle, les serveurs multi-sites doivent s'envoyer des informations de haut niveau pour contourner les limitations de bandes passantes. On peut envisager l'échange non seulement de données, mais de petits programmes que les serveurs (ou même les clients modulo quelques problèmes de sécurité) exécuteront à distance, le concept d'appel de procédures distantes étant remplacé par l'envoi d'un *agent*, c'est-à-dire la migration d'un *processus mobile* porteur d'une requête. Il existe déjà de tels langages avec agents, par exemple Facile [TLK96] de B. Thomsen à l'ECRC-Munich, Obliq de L. Cardelli à DEC/SRC ou Pict [PT97] de B. Pierce et D. Turner à Edimbourg/Indiana/Penn. C. Hewitt avait aussi autrefois développé le langage Plasma au MIT en 1970. Dans le monde commercial, il y a Telescript de General Magic. D'autres langages, comme Java de Sun Microsystems ou le Hot Caml du projet Cristal, permettent l'envoi de programmes sur le *World Wide Web*. Parmi ces propositions, seuls Obliq et Pict fournissent l'envoi de sous-programmes actifs, c'est-à-dire de processus avec leurs environnements actifs en cours d'exécution. En travaillant sur la définition d'un Pict distribué, nous avons introduit

[TLK96] B. Thomsen, L. Leth, T.-M. Kuo, «A Facile Tutorial», *in : CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)*, U. Montanari, V. Sassone (éditeurs), *LNCS, 1119*, Springer, p. 278–298, 1996.

[PT97] B. C. Pierce, D. N. Turner, «Pict: User Manual», Available electronically, 1997.

en 1995 un nouveau calcul, le *join-calcul*, dont nous avons montré la relation avec le π-calcul [Rob91] de Milner. Ce nouveau calcul est implémentable, même en présence de pannes. De tous les langages précédemment cités, il est actuellement le seul à être implanté dans un milieu distribué sur tout système Unix. Nous avons deux systèmes diffusés en *ftp* anonyme: le join-calcul 1.04 et *JoCaml* complètement compatible avec le langage Ocaml du projet Cristal.

L'intérêt de notre projet est porté non seulement vers les fondements sémantiques des langages de programmation, mais aussi vers leurs implémentations et à plus long terme vers la construction de systèmes informatiques concurrents. Nous avons naturellement des relations techniques suivies avec d'autres groupes de l'INRIA dans les domaines des systèmes d'exploitation, des langages de programmation, et des systèmes de preuves. Du coté théorique, nous poursuivons des actions sur les équivalences de processus, les protocoles de sécurité, particulièrement importants dans le cas d'applications distribuées, sur le typage et sur les objets concurrents. Le travail sur le join-calcul a été effectué par la grande majorité des membres de notre projet et par D. Rémy du projet Cristal. Nous participons au Working Group CONFER-2 du Framework 4, dont nous sommes le site coordinateur. En 1999, nous avons démarré un projet RNRT avec le projet Meije, l'ENST et le CNET.

Par ailleurs, le projet continue l'activité sur l'analyse statique de programmes, démarrée en 1997 avec l'arrivée d'Alain Deutsch. Avec le départ en disponibilité d'Alain Deutsch dans la création de PolySpace Technologies, cette activité s'est un peu réduite, autour des travaux de Bruno Blanchet sur l'optimisation de la compilation de Java avec l'OSF à Grenoble grâce à des analyses d'échappements. Le projet conserve toutefois une activité de conseil dans le domaine du code embarqué : G. Gonthier a encadré la rédaction de nouvelles spécifications de management de projets logiciel pour le projet Ariane 5, avec le CNES et l'ensemble des industriels concernés ; J.-J. Lévy a dirigé une expertise du logiciel du module européen Colombus de la future station spatiale internationale.

Enfin, notre projet maintient une certaine activité dans les preuves formelles, initialisée par la longue preuve de sureté du glaneur de cellules de Caml concurrent (dont la version monoprocesseur est le système incrémental de Caml et de Ocaml). Cette année, Damien Doligez a effectué un travail de 6 mois au laboratoire Compaq Systems Research Center de Palo Alto (USA) pour achever avec Leslie Lamport une longue preuve de la validité d'un protocole de cache des prochaines générations du processeur Alpha.

En 1999, nous avons un nouveau doctorant: Alan Schmitt, dirigé par J.-J. Lévy, qui a commencé un travail sur l'implémentation en join-calcul du calcul des ambiants de Luca Cardelli (Microsoft Research).

Enfin, le projet Para a changé de responsable scientifique à la fin de 1999, et sera dirigé par Georges Gonthier. En même temps, le projet change de nom, et s'appellera Moscova (Mobilité, Sécurité, Concurrence, Vérification et Analyse) en 2000 ; ce nouveau nom reflétant mieux les orientations scientifiques prises par le projet ces dernières années.

[Rob91] Robin Milner, « The Polyadic π-Calculus: a Tutorial », *rapport de recherche n° ECS-LFCS-91-180*, LFCS, University of Edinburgh, octobre 1991, Also in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer, 1993.

3 Fondements scientifiques

3.1 Le join-calcul

R. Milner a démarré la théorie de la concurrence en 1980 à Edimbourg, en proposant un calcul des systèmes communicants (CCS)[Mil89]. Cette théorie a eu de nombreux développements algébriques ou logiques, qui ont démontré le coté non trivial de sa modélisation. En 1989, R. Milner, J. Parrow et D. Walker[MPW92] ont introduit un nouveau calcul, le *pi-calcul*, permettant de considérer les réseaux de communication reconfigurables. Cette théorie a été reprise puis affinée par D. Sangiorgi (Edimbourg/INRIA Sophia). Beaucoup d'autres calculs de processus ont fleuri pendant ces 15 dernières années. Notre projet s'intéresse principalement au pi-calcul dont la théorie de l'équivalence est loin d'être simple: bisimulations *early*, *late*, *open*, etc.

Nous avons introduit un nouveau calcul de processus, le join-calcul [6, 5], qui est facilement implémentable dans tout système distribué, car il ne nécessite aucun protocole de communication sophistiqué, tels que ceux qui permettent la diffusion atomique ou le consensus distribué.

Le join-calcul permet la programmation concurrente et distribuée, ainsi qu'un communication et une coopération simple entre deux tâches qui s'exécutent sur des machines différentes. Dès le départ, le join-calcul a été conçu en tenant compte de la "localisation" des objets manipulés (processus, canaux, groupes de tels objets). Ce souci a facilité la production d'un langage de programmation qui donne au programmeur une vision de relativement haut niveau d'un réseau de machines. Classiquement, cette vision de haut niveau cache les détails de la programmation distribuée sur un système particulier et permet au programmeur de se concentrer sur l'architecture de son programme.

Dans le cas du join-calcul, une première brique est la communication par canaux, qui peuvent eux-mêmes voyager sur les canaux. Les canaux relient des processus entre eux, une restriction fondamentale du join-calcul est d'imposer un récepteur unique sur chaque canal, ceci permet l'identification d'un canal avec son récepteur. La perte d'expressivité qui en résulte est compensée par la réception jointe, une certaine action ne se déclenchant que si plusieurs canaux reçoivent un message. La nature du calcul et du langage permet la localisation des canaux engagés dans une réception jointe sur la même machine. La réalisation de rendez-vous entre des processus réellement distribués (i.e. résidant sur des machines différentes) devient possible car ce rendez-vous aura lieu par une communication jointe qui se décomposera en deux phases, une phase de transport des messages vers la machine qui possède les canaux de la réception jointe, puis une phase de rendez-vous purement locale et donc assez simple.

La deuxième brique de base du join-calcul est la location. Le contrôle effectif de la localisation des canaux et processus se fait par ce biais: une location est un ensemble de canaux et de processus conçus pour migrer ensemble d'une machine à l'autre. Les locations pouvant être créées dynamiquement, il s'ensuit une notion naturelle de parenté entre locations, puis de structure arborescente des locations. La migration d'une location correspond au déplacement d'un sous-arbre vers un autre point d'attache dans l'arbre des locations. Ici encore, le langage

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

[MPW92] R. Milner, J. Parrow, D. Walker, «A Calculus of Mobile Processes, Parts I and II», *Journal of Information and Computation 100*, septembre 1992, p. 1–77.

reprend un concept issu des calculs de processus, les machines n'apparaissent pas dans le langage, seul apparaît une location racine par machine. Le réseau de machine est donc abstrait en une forêt de locations, les migrations sont explicites, dans le sens que le programmeur spécifie bien une migration, mais elles restent d'assez haut niveau, tant par ce qui migre (un ensemble cohérent de canaux et de processus contenu dans une location et toutes ses sous-locations) que par la façon dont est donnée la destination de la migration (une autre location dont la location migrante devient une sous-location). Cette approche est entièrement intégrée dans notre prototype et permet l'écriture de programmes distribués, avec communications distantes, migration de code, applets, etc.

3.2 La sémantique des langages de programmation et leur compilation

Le *join-calcul* sert de fondement à un langage de programmation. Dans ce langage, la synchronisation a une écriture proche de celle de l'appel des fonctions dans les langages fonctionnels, le langage est d'ailleurs interfaçable avec le système Caml.

Notre projet développe des compilateurs et des environnements pour un tel langage. Une première version de notre langage est maintenant disponible sur le réseau depuis mi-1997. Une deuxième version `jocaml` a été aussi diffusée à la mi-1998. Elle a un meilleur interfaçage avec Caml, car elle est une extention de la machine virtuelle Ocaml de X. Leroy (Cristal).

Le travail d'implémentation visera dans le futur à détecter les erreurs asynchrones, comme les pannes dues à la déconnexion d'un site. La détection d'erreurs en milieu asynchrone étant clairement indécidable, il sera nécessaire d'ajouter un minimum d'opérateurs synchrones à notre modèle. Ce domaine se rapproche par de multiples facettes de celui de l'algorithmique distribuée.

Le contexte scientifique du projet bénéficie également des travaux sur la compilation des langages fonctionnels, tels que Caml où se posent des problèmes assez proches comme le filtrage ou l'inférence de types. Il bénéficie également de travaux menés en commun avec le projet SOR sur les glaneurs de cellules distribués et les implémentation de références distantes.

3.3 Interprétation abstraite

Mots clés : analyse statique de programmes, vérification, optimisation, sémantique dénotationnelle.

L'analyse statique vise à prédire statiquement et automatiquement les comportements possibles des programmes, sans les exécuter. La découverte automatique du comportement exact des programmes étant indécidable, on cherche à découvrir des propriétés approchées des programmes. Ces propriétés sont typiquement utilisées pour optimiser les programmes on encore pour les vérifier.

La théorie de l'interprétation abstraite a été initiée en 1977 par P. Cousot et R. Cousot [CC79] permettant de justifier formellement la correction d'algorithmes d'analyse statique, de spécifier constructivement de nouvelles méthodes d'analyse et aussi de donner des critères de

[CC79] P. Cousot, R. Cousot, «Systematic design of program analysis frameworks», *in : 6th Annual ACM Symposium on Principles of Programming Languages*, p. 269–282, 1979.

complétude. Par exemple, la notion d'approximation des propriétés exactes des programmes est formalisée par des correspondances de Galois ou encore par des opérateurs de fermeture sur des treillis complets.

Dans ce cadre, notre agenda scientifique consiste à concevoir de nouveaux algorithmes d'analyse statique pouvant s'appliquer de manière effective aux programmes industriels, qui sont caractérisés par leur grande taille, ainsi que par des traits de langage non triviaux, comme les pointeurs ou la concurrence.

Sur le plan théorique, il s'agit de prouver formellement la correction de ces algorithmes, d'étudier leur complexité, et le cas échéant de démontrer leur complétude. Sur le plan expérimental, il s'agit de quantifier l'applicabilité, l'efficacité et la précision de ces nouveaux algorithmes.

4 Domaines d'applications

Mots clés : télécommunications, applications distribuées, sécurité, analyse statique de programmes, vérification.

La programmation distribuée avec mobilité intervient dans la programmation du World Wide Web et des systèmes mobiles autonomes. Elle permet la personnalisation des interfaces et de la communication entre plusieurs clients. Les télécommunications sont un autre domaine d'application, avec les commutateurs actifs ou les services dits intelligents.

L'analyse statique de programmes a des retombées dans la validation des logiciels critiques embarqués pour l'aéronautique, l'automobile ou l'espace. Il a par exemple été utilisé pour vérifier certaines propriétés du logiciel de bord des vols Ariane 502 et 503, de l'ARD et de divers satellites. Un autre domaine d'application est l'optimisation du code généré par les compilateurs de langages de programmation de haut niveau, comme Ocaml ou Java.

5 Logiciels

5.1 JoCaml

Participants : Sylvain Conchon, Fabrice le Fessant.

Ce logiciel est disponible sur le réseau en `http://pauillac.inria.fr/jocaml`.

Seconde implémentation du join-calcul, JoCaml est une extension du language Objective-Caml 1.07 intégrant le join-calcul. Comparée à la première implémentation, JoCaml permet de programmer en Caml tout en bénéficiant pour la programmation distribuée des canaux de communication et de synchronisation, et des agents mobiles du join-calcul, et d'objets distribués.

JoCaml a bénéficié des travaux de L. Maranget et F. Le Fessant sur la compilation et l'implémentation des canaux du join-calcul, et de F. Le Fessant sur les ramasse-miettes distribués. Outre les programmes développés pour la première implémentation du join-calcul, de nombreux exemples ont aussi été implémentés, parmi lesquels plusieurs jeux en réseau, tels que *bomberman*, *pong* et *bataille navale*, ainsi qu'un plugin Netscape permettant de lancer ces jeux

depuis une page Web. Des versions mobiles de HéVéa et de Efuns ont aussi été développée, montrant la robustesse de la plate-forme à la migration de programmes conséquents.

Cette nouvelle plate-forme doit permettre le développement et l'expérimentation de nouveaux algorithmes de ramasse-miettes, de gestion de pannes et sécurisation pour la programmation distribuée. JoCaml a aussi été utilisé au sein du projet SOR pour le développement d'un système d'objets répliqués pour les systèmes mobiles (Cadmium).

5.2 Le join-calcul 1.04

Participant : Luc Maranget.

Ce logiciel est disponible sur le réseau en `http://join.inria.fr`.

Le système 1.04 se compose d'un compilateur à byte-code, d'un environnement d'exécution, ainsi que d'une centaine de pages de documentation. Tant le compilateur que l'environnement d'exécution sont écrits en Ocaml, ce sont des programmes conséquents qui totalisent plus de 20000 lignes de code. Le développement proprement dit (comprenant l'écriture de la documentation) a duré environ 12 mois. Ont également été écrits un certain nombre d'exemples, aussi disponibles en ligne, qui illustrent les possibilités du langage.

5.3 Optimisation de l'allocation et de la synchronisation en Java

Participant : Bruno Blanchet.

L'implantation de l'analyse d'échappement pour Java dans le compilateur Java vers C turboJ de l'Opengroup Grenoble, présentée au paragraphe §6.9, s'est poursuivie. Ce travail a concerné cette année plus particulièrement l'élimination des synchronisations inutiles et des améliorations de précision de l'analyse d'échappement (analyse des appels d'interface, des sous-routines, utilisation d'une analyse de types pour améliorer la résolution des appels virtuels, algorithme visant à déterminer précisément quelle partie du code sera chargée à l'exécution).

Les résultats obtenus sont tout à fait satisfaisants. L'expérimentation a montré la faisabilité et le coût raisonnable de l'analyse, puisqu'on a pu l'appliquer à des applications de plus d'1Mo de fichiers *.class*, avec un coût d'analyse d'environ 10% du temps total de compilation du bytecode Java vers le code natif (en passant par le C comme intermédiaire). Cette étude a montré l'intérêt de l'allocation en pile et de l'élimination des synchronisations sur les objets locaux à un thread en Java, puiqu'on a obtenu des gains de temps d'exécution de 21% (moyenne géometrique sur nos tests), à plus de 40% dans des cas particulièrement favorables.

5.4 Glaneur de cellules pour Caml et portages sur Macintosh de Objective Caml

Participant : Damien Doligez.

Les systèmes Caml et Objective Caml du projet Cristal utilisent un glaneur de cellules conçu et programmé dans notre projet. Ce GC est critique pour la bonne performance de Caml. Il autorise aussi le compactage de la mémoire, utile pour certaines applications dréseau,

et les pointeurs faibles, qui interviennent pour optimiser l'implémentation du join-calcul ou de JoCaml.

Par ailleurs, le port de Objective Caml sur Macintosh par D. Doligez fournit un nouvel interface utilisateur plus robuste et plus facile à maintenir que celui de Caml-light, qui sert déjà pour l'enseignement.

5.5 Hevea

Participants : Luc Maranget, Nicolas Tessaud.

Hevea est un traducteur de LaTeX vers HTML. Il est entièrement écrit en Objective Caml. Luc Maranget a écrit Hevea initialement pour satisfaire ses besoins d'enseignant : présenter aux élèves à la fois un cours écrit et une version web de ce cours.

Hevea est toujours en cours de développement et de perfectionnement. La version courante est la 1.04 [20] (1.02 l'année dernière). La principale amélioration réalisée cette année est l'ajout d'autres formats de sortie. À l'heure actuelle Hevea peut traduire LaTeX vers du HTML (version 4.0 transitionnelle), vers du texte formatté ou du format `info` (format hypertexte limité utilisé par GNU-Emacs). Cette extension a été réalisée de mai à juillet par Nicolas Tessaud, élève de dernière année à l'École polytechnique, dans le cadre d'un stage encadré par Luc Maranget [21].

La diffusion d'Hevea se poursuit a un rythme raisonnable, la dernière version, disponible sur le site ftp de l'Inria à partir de juillet a été téléchargée 800 fois à la date du 31 octobre.

De plus des chercheurs exterieurs à l'institut participent à la diffusion d'Hevea :

- Philip A. Viton, *Associate Professor of City and Regional Planning* à l'Université d'État de l'Ohio, maintient et diffuse une version pour Windows d'Hevea.
- Georges Mariano de l'Institut national de recherche sur les transports et leur sécurité (Inrets) a mis en place une liste des utlisateurs d'Hevea en septembre. Cette liste compte actuellement de l'ordre de cinquante abonnés.

Hevea a été utilisé à plusieures reprises par des équipes ou des chercheurs de l'Inria pour produire des versions HTML de manuels de logiciels, on peut citer le projet CRISTAL (manuel Objective Caml), Daniel Diaz du projet LOCO (manuel GNU-prolog) et le projet COQ (manuel de COQ). À l'exterieur de l'institut, Christian Queinnec (Professeur à l'Université de Paris VI) utilise Hevea dans son système ViDeOC de production de CD-ROMs pédagogique, Pierre Boulet, du Laboratoire d'informatique fondamentale de Lille, utlise Hevea dans son système MLDoc de documentation de source Objective Caml.

Luc Maranget a écrit un article sur l'architecture interne d'Hevea, article qui a été publié à l'occasion des dixièmes Journées francophones des langages applicatifs [15]

5.6 Efuns

Participant : Fabrice le Fessant.

Ce logiciel est disponible sur le réseau en `http://pauillac.inria.fr/efuns`.

Efuns est un petit éditeur de texte, clone d'Emacs, développé en Objective-Caml et configurable en Objective-Caml. Initialement conçu comme un exemple d'application mobile pour

JoCaml, Efuns est aujourd'hui utilisé quotidiennement au sein du projet comme éditeur intégré pour le développement de nouvelles applications en Caml et JoCaml.

Efuns est distribué avec un ensemble de librairies et de programmes indépendants: une librairie de communication avec le serveur X (clone d'Xlib), une librairie d'objets graphiques, un interprète de bytecode Caml, un gestionnaire de fenêtres générique et configurable (GwML), et un optimiseur de code assembleur Caml.

6 Résultats nouveaux

6.1 Le join-calcul

Participants : Sylvain Conchon, Fabrice le Fessant, Georges Gonthier, Luc Maranget, Jean-Jacques Lévy, Alan Schmitt.

Nous avons consolidé en 1999 nos travaux autour du join-calcul. Sur le plan système, le support pour la programmation distribuée (gestion d'allocation, terminaison) a été fortement renforcé dans JoCaml par F. le Fessant §5.1§6.2. S. Conchon a programmé plusieurs exemples de jeux distribués exploitant ces nouvelles facilités.

Sur le plan langage, S. Conchon a développé, avec Didier Rémy (Cristal) une nouvelle présentation du système de type du Join-Calcul avec variables de rang, permettant un typage plus précis des définitions mutuellement récursives. S. Conchon a d'autre part construit un système de sous-typage pour le Join-Calcul s'inspirant des travaux de François Pottier (Cristal) sur la synthèse de type en présence de sous-typage. S. Conchon prépare, dans le cadre du projet Marvel, un rapport sur le fonctionnement de la machine virtuelle du join-calcul, et, avec L. Maranget, un autre rapport sur l'interface du système de types du join-calcul avec celui d'Objective-Caml.

Enfin, sur le plan théorique, G. Gonthier a rédigé avec C. Fournet un article décrivant les techniques de preuve d'équivalences qu'ils ont développées au cours de leurs travaux théoriques sur le join-calcul. Ces techniques sont dérivées des techniques de preuve de confluence de réécriture de V. Van Oostrom [Oos94]. Leur utilisation informelle au début [6], s'est imposée au cours des travaux sur les preuves de sécurité §6.3, et a été systématique et indispensable pour l'étude de l'implantation des ambients §6.4.

6.2 Ramasse-miettes distribués

Participant : Fabrice le Fessant.

En 1999, nous avons poursuivi notre collaboration avec le projet SOR pour le développement de ramasse-miettes distribués à large-échelle, en particulier l'extension du système des *Stub-Scion Pair Chains* (SSPC) pour la gestion des références vers les agents mobiles d'une part, et pour la détection des cycles de miettes distribués d'autre part.

Ainsi, JoCaml intègre un système de références distantes proches des chaînes de paires souches-scions, permettant à la fois la mobilité des objets, le raccourcissement rapide des

[Oos94] V. Oostrom, « Confluence by Decreasing Diagrams », *Theoretical Computer Science 126*, 1994, p. 259–280.

chaînes de références et le ramassage automatique et complet des objets non utilisés, en utilisant un nouveau détecteur de cycles de miettes. Ce détecteur est basé sur trois nouvelles techniques de propagation de marques lors des traçages locaux: une première technique, appelée marquage min-max, consiste à réunir les traces locales par paires pour détecter les confluences de marques différentes. Ce mécanisme permet de détecter les cycles les plus simples. Une seconde technique, appelée sous-génération, doit être utilisée pour les cycles complexes. Cette technique consiste à recouvrir une partie du graphe cyclique par un graphe acyclique, pour supprimer les sous-cycles. Lors de ce processus, une variante optimiste du back-tracing est utilisée pour trouver les arcs du graphe acyclique. Ces trois techniques rendent l'algorithme simple à implémenter, économe en ressources et adapté aux réseaux large-échelle.

L'ensemble de ce travail a fait l'objet de soumissions à PODC'99 et CFSE'99.

6.3 Concurrence et sécurité

Participant : Georges Gonthier.

En 1999, nous avons poursuivi notre collaboration avec M. Abadi (Lucent Technology) et C. Fournet (Microsoft Research) sur la spécification et la preuve d'abstractions sûres de protocoles cryptographiques.

Notre méthodologie consiste à aborder l'étude des propriétés de sécurité avec les outils de la sémantique des langages de programmation. En effet la sécurité est une notion à la fois informelle et globale; il n'est pas aisé de trouver une formulation de la "sécurité" qui soit précise sans être réductrice. Notre approche [Aba98] consiste à définir un ensemble d'opérations de haut niveau, qui garantissent automatiquement des propriétés de securité élémentaires comme le secret ou l'authenticité, et d'en donner une implantation en terme de primitives cryptographiques. Ceci nous permet de ramener le problème insoluble de la sécurité à celui de la complète adéquation de l'implémentation au langage défini par les opérations de haut niveau.

Nous avons choisi comme premier cas d'étude la communication par canaux privés du join-calcul. En 1999 nous avons poursuivi ces travaux. D'une part nous avons montré que la sécurité de l'implantation pouvait être assurée de façon transparente en interposant un module d'interface entre chaque unité de programme (ou *principal*) et le réseau [7], plutot que par une compilation spéciale chaque principal comme en [2]. et d'autre part en étudiant les propriétés sémantiques qui doivent êtres satisfaites par le protocole de transmission utilisé entre les modules d'interface [8]. Un long article [18] regroupant l'ensemble de ces travaux est en cours de soumission.

Notre principal résultat pour 1999, toutefois, est l'ajout d'opérations d'authentification, assorti d'une notion primitive de principal, au langage de haut niveau. Cet ajout présente deux intérêts majeurs. D'une part, il augmente considérablement l'expressivité du langage de haut niveau, puisqu'il permet de décrire directement des politiques de contrôle d'accès (les canaux privés ne permettant d'exprimer que des capacités). D'autre part, et de façon un peu surprenante, il permet d'avoir des implantations beaucoup plus efficaces. En effet, la présence d'identités explicites dans le langage de haut niveau libère l'implémentation de la contrainte

[Aba98] M. Abadi, « Protection in programming-language translations », *in : Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, p. 868–883, juillet 1998.

d'anonymat imposée par le modèle du join-calcul pur; ainsi, on peut multiplexer toutes les communications entre deux principaux sur un unique session, ce qui réduit grandement le nombre d'opérations cryptographiques nécessaires. Ces travaux seront présentés à la conférence POPL 2000 [9].

6.4 Codage et implémentation des 'ambients' dans le join-calcul

Participants : Jean-Jacques Lévy, Alan Schmitt.

En 1999, nous avons effectué une implémentation distribuée du calcul des ambients dans le join-calcul. Cette implémentation, réalisée en collaboration avec Cédric Fournet (Microsoft Research), a consisté à tout d'abord écrire une traduction théorique, qui a ensuite été implémentée puis vérifiée. Ce travail nous a permis de mieux appréhender les similarités et différences des deux calculs, ainsi que d'étudier l'implémentabilité du calcul des ambients. Ce résultat est intéressant à plusieurs titres. Cette première implémentation distribuée des ambients présente un algorithme décrivant de manière très explicite les synchronisations nécessaire à la traduction de la sémantique opérationnelle des ambients dans un calcul fonctionnant par passage de messages, exhibant ainsi les nombreuses synchronisations non locales implicites dans le calcul des ambients. La traduction théorique a été présentée à MOS'99 [13], l'implémentation est disponible sur le web [19].

Une deuxième partie du travail à consisté à prouver la correction de la traduction. Cette preuve, qui utilise les simulations couplées barbues ainsi que les diagrammes décroissants, est en cours d'achèvement. Elle est composée de deux parties. La première, la plus complexe, est la preuve de l'algorithme. Les étapes de synchronisations sont décomposées en étapes élementaires dans un calcul d'ambients étendus qui est mis en relation avec le calcul initial par des simulations couplées barbues. La deuxième étape, correspondance du calcul étendu avec le join-calcul, établit l'existence d'une bisimulation faible entre ces deux calculs en utilisant la technique des diagrammes décroissants.

6.5 Objets distribués

Participant : Luc Maranget.

La recherche sur une extension objet du join-calcul s'est poursuivie cette année, il s'agit d'une collaboration entre Cédric Fournet (Microsoft Reseach), Cosimo Laneve (Université de Bologne), Luc Maranget et Didier Rémy (Projet CRISTAL). Une notion de vue a été introduite.

6.6 Substitutions explicites et logique

Participant : Thérèse Hardin.

En collaboration avec Gilles Dowek (Coq) et Claude Kirchner (Prothéo), T. Hardin a continué ses travaux sur la déduction modulo. Ils ont proposé une formulation de la théorie des types simples dans le cadre de la déduction modulo utilisant le calcul des substitutions explicites [1, 3] comme langage d'expression des fonctions. Ils ont montré que la résolution

modulo appliquée à cette théorie simulait la résolution d'ordre supérieur étape par étape. Ce travail a été publié dans [12].

6.7 Substitution explicite et lambda-calcul faible

Participants : Jean-Jacques Lévy, Luc Maranget.

Ce travail précise la notion de lambda calcul faible souvent employée dans l'étude des langages de programmation fonctionnels. Dans le lambda calcul faible, on ne réduit pas les redexes sous les lambda abstractions. Çağman et Hindley ont donné une définition du calcul faible[cH98] dans le cadre du lambda calcul et de son rapport avec la logique combinatoire. Nous reformulons cette définition pour le lambda calcul et analysons sa relation avec le calcul des substitutions explicites. Plusieurs notions de calcul faible avaient été introduites dans le formalisme de la substitution explicite [1]. Mais notre travail souligne la simplicité du calcul de substitutions ainsi obtenu (confluent et sans indices de de Bruijn). Par exemple, on peut se souvenir que dans le cas général le calcul des substitutions explicites n'est pas confluent [3]. Dans notre calcul faible, on peut exprimer de manière naturelle les stratégies standard d'évaluation de lambda-expressions. Mais la formulation précise des diverses propriétés d'adéquation conduit à la démonstration de théorèmes ou de propriétés non évidentes (confluence, standardisation, partage, évaluation avec des piles). Cela redonne un peu plus de vigueur à un formalisme souvent ignoré ou considéré comme facile.

Le travail a été présenté à Trente au Workshop Westapp et à la conférence FST/TCS 99 [14].

6.8 Environnement certifié de calcul formel

Participants : Thérèse Hardin, Damien Doligez.

T. Hardin coordonne le projet, appelé FOC, de développement d'un environnement pour la programmation certifiée en Calcul formel. Les participants de ce projet sont R. Rioboo, V. Ménissier-Morain et S. Boulmé, membres du LIP6 ainsi que D. Doligez depuis l'automne 1999. Ce projet participe à l'action de recherche coordonnée CFC, avec les projets COQ, CRISTAL et Lemme, qui prend fin en décembre 1999. La bibliothèque proposée est écrite en Ocaml et utilise fortement les traits objet de ce langage. La méthodologie de son développement a été complètement établie, après réalisation de plusieurs prototypes, de manière à faciliter au maximum l'expression des énoncés mathématiques sous-jacents et le couplage avec les preuves de correction à faire avec Coq. Sa robustesse a été testée auprès de stagiaires (bac+3) chargés d'intégrer d'une part, des outils pour les fractions et d'autre part, des outils permettant de choisir par simple édition de liens entre les bibliothèques Bignum et GMP de grands entiers.

La bibliothèque comprend actuellement une soixantaine de classes (3000 lignes de Ocaml) et propose plusieurs implantations des polynômes, dont une inédite. Les temps de calcul obtenus sont excellents et même, les meilleurs pour certains algorithmes de calcul de racines de polynômes.

[cH98] N. Çağman, J. R. Hindley, « Combinatory Weak Reduction in Lambda Calculus », *Theoretical Computer Science 198*, 1998, p. pp. 239–249.

L'architecture de la bibliothèque exprime des dépendances de différente nature entre éléments de cette bibliothèque : héritage, raffinement, spécialisation, oubli, etc. Pour certifier des programmes de cette bibliothèque, il faut d'abord formaliser ces liens de dépendance entre unités. Le travail, effectué par S. Boulmé en thèse avec T. Hardin, a bien progressé sur ce point, avec l'introduction de la notion des mixtel (une sorte d'enregistrement avec champs dépendants et opérations traduisant les différentes notions de dépendance).

6.9 Analyse d'échappements et optimisation de Java

Participants : Alain Deutsch, Bruno Blanchet.

L'analyse d'échappement (*escape analysis*) [4] est une analyse par interprétation abstraite, qui vise à déterminer si la durée de vie d'une donnée dépasse sa portée statique. Elle peut être utilisée pour remplacer l'allocation dynamique avec utilisation d'un glaneur de cellules (GC), par de l'allocation en pile, plus efficace car elle diminue la charge du GC et elle améliore la localité des données (très importante pour optimiser la gestion des caches dans les processeurs modernes).

L'étude de cette analyse s'est poursuivie cette année, dans le cadre de son application au langage Java. Nous avons d'une part développé une preuve de correction complète de l'analyse bidirectionelle qui est nécessaire pour traiter avec suffisament de précision les affectations de Java. D'autre part, nous avons étendu l'application de l'analyse à l'optimisation des opérations de synchronisation de Java : ces opérations, fort coûteuses, peuvent en effet être supprimées pour les accès à des données locales allouées en pile et plus généralement pour toutes les données accessibles depuis un seul thread. Nous avons réalisé de nombreuses mesures sur l'implantation de ces optimisations §5.3, afin de démontrer leur efficacité, et d'étudier l'impact de différentes variantes. Les gains sont en moyenne de 20%, pour un surcoût de compilation modeste. Ces résultats ont été présentés à la conférence OOPSLA'1999 [11].

6.10 Langages de programmation fonctionnels

Participants : Damien Doligez, Luc Maranget.

D. Doligez a continué le travail sur Objective Caml, pour ajouter les valeurs finalisées et pour le portage sur Macintosh (Mac OS 8, Mac OS X Server (en collaboration avec X. Leroy (Cristal)), et Linux), ainsi que pour participer au déboguage et à la maintenance de Objective Caml.

Luc Maranget a perfectionné les messages d'erreur du compilateur d'Objective Caml, émis en cas de filtrage non-exhaustif (dont il avait déjà réalisé la détection en 1995). Grâce à un nouvel algorithme, le compilateur OCaml calcule maintenant un exemple spécifique de valeur non-filtrée, qu'il inclut dans ses diagnostics.

7 Contrats industriels (nationaux, européens et internationaux)

7.1 Projet Marvel

Participants : Sylvain Conchon, Georges Gonthier, Luc Maranget, Fabrice le Fessant, Jean-Jacques Lévy, Alan Schmitt.

Le projet Marvel est un projet RNRT commencé en novembre 1998, et coordonné par Jean-Bernard Stéfani, en collaboration avec l'Inria Sophia (projet Meije), le Cnet Issy et Lannion (Jean-François Monin et Pascal Brisset), l'ENST (Elie Najm). Il s'agit de développer un langage et une plate-forme avec agents mobiles et objets distribués qui puisse être utile pour la programmation des applications en télécommunications.

La première année a consisté en l'étude des différents modèles formels de description des langages de programmation d'applications distribuées, et des différentes implantations de ces langages de programmation. L'étude s'est terminé par la rédaction d'un document de synthèse, qui servira de base aux choix qui devront être pris pour la définition du langage de programmation et l'implantation de la plate-forme.

7.2 Nouvelles spécifications de management pour le logiciel d'Ariane 5

Participant : Georges Gonthier.

A la demande du CNES, G. Gonthier a dirigé un groupe de travail chargé de la remise à jour des spécifications de développement des logiciels du projet Ariane 5, en concertation avec l'ensemble des intervenants industriels. Le groupe a effectué une réécriture complète de la spécification générale des logiciels Ariane 5, et une réécriture partielle des spécifications de Management.

7.3 Création de PolySpace Technologies

Participant : Alain Deutsch.

Après le succès de l'expérimentation en grandeur réelle de l'utilisation de l'analyseur statique IABC pour Ariane 5, l'année 1998 a permis de lancer l'industrialisation de l'analyseur par la création d'une société de technologie, PolySpace Technologies, basée à Grenoble. Un effort de développement soutenu a permis à la société de lancer dès 1999 ses deux produits principaux, un analyseur de code Ada et un analyseur de code C.

7.4 Expertise du logiciel du module Colombus de la station spatiale internationale

Participants : Bruno Blanchet, Georges Gonthier, Jean-Jacques Lévy.

A la demande de Matra Marconi Space (MMS), B. Blanchet, G. Gonthier, J.-J. Lévy, et G. Muller (projet Compose) ont participé à un comité international d'expertise sur le logiciel embarqué par le module européen Columbus de la station spatiale internationale. Ce comité,

incluant des spécialistes de l'avionique et des logiciels embarqués de Marconi Electronics, TU-Darmstadt, et DASA-M, a conclu son travail par un rapport interne à MMS édité par J.-J. Lévy.

8 Actions régionales, nationales et internationales

8.1 Actions européennes

Nous sommes le site coordinateur du groupe de travail CONFER-2, (working group 21836), avec Cambridge (Milner), CWI (Klop), Edimbourg (Abamsky), ENS (Curien), KTH (Parrow), INRIA Sophia (Boudol), Pise (Montanari), Bologne (Asperti), Sussex (Hennessy), Warwick (Walker) et le CNET Lannion (Monin). Ce groupe constitue au niveau européen le principal groupe de réflexion sur les relations entre la fonctionnalité et la sémantique des processus mobiles.

Nous avons eu deux réunions à Pise (3 jours en mars) et à Paris (2 jours en novembre).

Les rapports d'avancement se trouvent en `http://para.inria.fr/confer`.

8.2 Collaboration avec Compaq, Systems Research Center

D. Doligez a passé 6 mois à Palo Alto (de novembre 1998 à mai 1999) pour travailler sur la preuve de la cohérence du système de cache asynchrone du processeur EV7 (nouvelle génération du processeur Alpha). Ce travail est conjoint avec L. Lamport (Compaq SRC), et a été présenté à la conférence FM'1999 [10].

F. Le Fessant a travaillé pendant 3 mois à Palo Alto de juin à septembre pour retirer des hypothèses de synchronisation d'un système de disques virtuels partagés (Petal), par l'utilisation d'un système de gestion de groupe. Ce travail a été effectué en collaboration avec Mark Hayden et Ed Lee (Compaq SRC).

8.3 Accueils de chercheurs étrangers

Nous avons reçu Peter Selinger, University of Illinois, John Fields, IBM Watson, Leslie Lamport, Compaq SRC, et James Leifer, Cambridge University.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

D. Doligez, L. Maranget, et A. Schmitt, avec X. Leroy et P. Cuoq (projet CRISTAL), ont brillament remporté le concours de programmation organisé dans le cadre d'ICFP'99, avec un programme écrit en OCAML. Le sujet était de réaliser en 72h un optimiseur de programmes de décision combinatoire, et le concours était ouvert à tous les centres de recherche du monde.

G. Gonthier est représentant de l'INRIA au comité directeur des Ecoles d'été CEA-EDF-INRIA, membre de la commission des abonnements de Rocquencourt et maître de conférence à l'Ecole Polytechnique. Il a été membre du comité de programme de la conférence COORD'1999. Il a été membre du jury Chargé de Recherche de Rocquencourt.

T. Hardin est professeur à l'Université Paris 6, en délégation au projet Para depuis octobre 1999. Elle anime, en collaboration avec V. Donzeau-Gouge (CNAM), un groupe de recherches sur l'utilisation des systèmes d'aide à la preuve (Coq, PVS, B) dans la spécification et la programmation. Elle participe au projet RNRT Calife, points 5 et 6, dans le cadre de ses travaux sur la déduction modulo. Elle est est membre du comité d'administration de SPECIF, et a été membre du comité de programme de WESTAPP.

T. Hardin a encadré, conjointement avec Gilles Dowek, le stage de DEA de Stéphane Vaillant et son travail de thèse. Elle encadre plusieurs autres thèses sur des sujets liés à l'utilisation de méthodes formelles dans le développement du logiciel. Elle a été présidente des jurys de thèse de C. Rinderknecht (CRISTAL), D. Hirschkoff (CERMICS), X. Thirioux (ENSIEHT), rapporteur des thèses de X. Thirioux et B. Barras (Coq).

J.-J. Lévy est professeur à temps-partiel à l'Ecole polytechnique, correspondant du concours d'entrée de l'X pour l'informatique, membre des commissions de recrutement de mathématiques et de mathématiques appliquées de l'Ecole polytechnique. Il a été rapporteur de l'habilitation de Khaled Bsaies à l'Université de Tunis et a participé au jury de la thèse de Sylvano Dal Zillio (projet Meije). Il a été aussi membre du comité de programme de HOOTS'99. Il est coordinateur du groupe de travail Esprit Confer-2 (WG-21836). Il est membre de la commission de spécialistes de Paris 7 en informatique, vice-président de la Commission d'Evaluation de l'INRIA, délégué de la CE pour la création du projet Oasis, et membre du jury d'admission 1999 de l'INRIA.

L. Maranget est chef de travaux pratiques à l'Ecole Polytechnique, il y a assuré en 1999 l'encadrement des travaux pratiques du cours de tronc commun, ainsi qu'un cours de compilation. Il est l'auteur et le correcteur du sujet d'informatique du concours de l'Ecole Polytechnique en 1999. Luc Maranget est membre du comité de programmation de la conférence française JFLA'2000.

Luc Maranget a donné un exposé "Programmation d'un réseau de machines : la solution des agents mobiles", dans le cadre d'un cycle "Science et technologies de l'information et de la communication : la recherche en direct", organisé par la Cité des sciences de la Villette.

9.2 Enseignement universitaire

Notre projet participe aux enseignements suivants:

- Algorithmes et programmation à l'Ecole polytechnique (G. Gonthier, J.-J. Lévy, L. Maranget, F. le Fessant). J.-J.Lévy était responsable du cours de 1ère année 1998-99, G. Gonthier y a enseigné et a été responsable des projets informatique, F. le Fessant et L. Maranget y ont assuré les travaux pratiques. [16]

- Introduction à la programmation à l'Ecole polytechnique (J.-J.Lévy), cours d'une semaine en juin 1999 (cf. `http://w3.edu.polytechnique.fr/informatique/Init0/semaine2.html`)

- Langages et compilation à l'Ecole polytechnique (L. Maranget). L. Maranget a assuré les travaux dirigés de ce cours de 2ème année de l'Ecole polytechnique, enseigné en 1999 par D. Rémy.

- Concurrence et mobilité au DEA Sémantique, preuves et programmation (G. Gonthier et J.-J. Lévy).

- Interprétation abstraite au DEA Sémantique, preuves et programmation (A. Deutsch avec P. Granger de l'ENSTA).

- Cours de lambda-calcul au DEA Sémantique, preuves et programmation (T. Hardin, J.-J. Lévy) [17].

- Cours de join-calcul à l'école de printemps d'informatique à Marseille (J.-J. Lévy).

En collaboration avec V. Viguié Donzeau-Gouge, T. Hardin a mis en place un DESS, co-habilité par le CNAM et Paris 6, intitulé "Développement des logiciels sûrs". Ce DESS (`http://www-spi.lip6.fr/~hardin`) a pour objectif de former ses étudiants aux métiers de la sûreté de fonctionnement et de la sécurité de systèmes informatiques à forte composante logicielle, métiers dans lesquels l'interêt des méthodes formelles commence à être reconnu. Elle en a assuré la rentrée en septembre 1999 et a donné également une partie du cours de sémantique des programmes séquentiels.

Para est équipe d'accueil de l'école doctorale EDITE.

9.3 Participation à des colloques, séminaires, invitations

JFLA 1999, Morzine: présentation de T. Hardin (programmation en Foc, par R. Rioboo).

Confer-2, Pise: présentation d'A. Schmitt, participation de J.-J. Lévy, L. Maranget et F. le Fessant.

IEEE S&P 1999, Berkeley: Présentation de G. Gonthier, avec C. Fournet et M. Abadi (modèle d'implantation de canaux sécurisés).

MOS 1999, Lisbonne: Présentation d'A. Schmitt, avec C. Fournet (codage des ambients).

FLOC 1999 (LICS, CAV, CADE, RTA, CALCULEMUS et WESTAPP), Trento: Présentations de A. Deutsch (détection d'erreurs d'exécution par analyse statique), de J.-J. Lévy (lambda-calcul faible), de T. Hardin (déduction modulo, certification en Foc, par S. Boulmé), participation de G. Gonthier.

FM 1999, Toulouse: Présentation de D. Doligez, avec L. Lamport (vérification de l'architecture de l'EV7).

PLI 1999 (ICFP, PPDP, HOOTS), Paris: Présentation invitée de G. Gonthier (modélisation de la mobilité en join-calcul), participation de J.-J. Lévy, A. Schmitt, B. Blanchet, T. Hardin.

OOPSLA 1999, : Présentation de B. Blanchet (application de l'analyse d'échappement à l'optimisation de code Java).

Confer-2, Paris: présentations d'A. Schmitt, de G. Gonthier, participation de J.-J. Lévy, L. Maranget, de S. Conchon, de T. Hardin et de F. le Fessant.

FST & TCS 1999, Madras: Présentations invitées de J.-J. Lévy (lambda-calcul faible et substitutions explicites), de M. Abadi (messages sécurisés, avec C. Fournet et G. Gonthier).

ASIAN 1999, Phuket : Participation de J.-J. Lévy.

10 Bibliographie

Ouvrages et articles de référence de l'équipe

[1] M. Abadi, L. Cardelli, P.-L. Curien, J.-J. Lévy, « Explicit Substitutions », *Journal of Functional Programming 1*, 4, 1991, p. 375–416.

[2] M. Abadi, C. Fournet, G. Gonthier, «Secure implementation of channel abstractions», *in : Proceedings of the Thirteenth Annual IEEE Symposium on Logic in Computer Science*, p. 105–116, juin 1998.

[3] P.-L. Curien, T. Hardin, J.-J. Lévy, « Confluence Properties of Weak and Strong Calculi of Explicit Substitutions», *Journal of the ACM 43*, 2, mars 1996, p. 362–397.

[4] A. Deutsch, « On The Complexity of Escape Analysis », *in : 24th Annual ACM Symp. on Principles of Programming Languages*, ACM Press, p. 358–371, janvier 1997.

[5] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, D. Rémy, « A Calculus of Mobile Agents», *in : CONCUR '96: Concurrency Theory (7th International Conference, Pisa, Italy, August 1996)*, U. Montanari, V. Sassone (éditeurs), *LNCS, 1119*, Springer, p. 406–421, 1996.

[6] C. Fournet, G. Gonthier, «The Reflexive Chemical Abstract Machine and the Join-Calculus», *in : Proceedings of the 23rd Annual Symposium on Principles of Programming Languages (POPL) (St. Petersburg Beach, Florida)*, ACM, p. 372–385, janvier 1996.

Communications à des congrès, colloques, etc.

[7] M. Abadi, C. Fournet, G. Gonthier, «Secure Communications Processing for Distributed Languages», *in : 20th IEEE Symposium on Security and Privacy*, IEEE, p. 74–88, mai 1999.

[8] M. Abadi, C. Fournet, G. Gonthier, «A top-down look at a secure message», *in : Proc. of the 19th Foundations of Software Technology and Theoretical Computer Science, Chennai*, décembre 1999.

[9] M. Abadi, C. Fournet, G. Gonthier, « Authentication primitives and their compilation », *in : 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM Press, janvier 2000. à paraître.

[10] H. Akhiani, D. Doligez, P. Harter, L. Lamport, J. Scheid, M. Tuttle, Y. Yu, «Cache Coherence Verification with TLA+», *in : FM'99 – Formal Methods*, J. M. Wing, J. Woodcock, J. Davies (éditeurs), *Lecture Notes in Computer Science, 1709*, Springer-Verlag, p. 1871, septembre 1999.

[11] B. Blanchet, «Escape analysis for Object-Oriented Languages. Application to Java(TM)», *in : Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press, p. 20–34, novembre 1999.

[12] G. Dowek, T. Hardin, C. Kirchner, « HOL-$\lambda\sigma$: an intentional first-order expression of higher-order logic », *in : Rewriting Techniques and applications*, P. Narendran, M. Rusinowitch (éditeurs), *Lecture Notes in Computer Science*, 1630, Springer-Verlag, p. 317–331, 1999.

[13] C. Fournet, A. Schmitt, « An Implementation of Ambients in JoCAML », *in : 5th Mobile Object System Workshop: Programming Languages for Wide Area Networks*, juin 1999.

[14] J.-J. Lévy, L. Maranget, «Explicit Substitutions and Programming Languages», *in: Proc. of the 19th Foundations of Software Technology and Theoretical Computer Science, Chennai*, December 1999.

[15] L. Maranget, «Hevea, un traducteur rapide de LaTeX vers HTML en Caml», *in: JFLA'99: 10^{ièmes} Journées Francophones des Langages Applicatifs*, février 1999.

Rapports de recherche et publications internes

[16] R. Cori, J.-J. Lévy, *Algorithmes et Programmation*, Ecole polytechnique, octobre 1999, `http://w3.edu.polytechnique.fr/informatique/TC/polycopie-1.6`.

[17] T. Hardin, *Lambda-Calcul*, DEA Programmation, 1999.

Divers

[18] M. Abadi, C. Fournet, G. Gonthier, «Secure Implementation of Channel Abstractions», à paraître, 1999.

[19] C. Fournet, A. Schmitt, «An Implementation of Ambients in JoCAML, preliminary version», disponible en `http://join.inria.fr/ambients.html`, 1999.

[20] L. Maranget, «Hevea 1.04», logiciel et documentation disponibles en `http:para.inria.fr/~maranget/hevea`, 1999.

[21] N. Tessaud, «Hevea: traduction de LaTeX», rapport de stage d'option scientifique de l'École polytechnique, juillet 1999.