INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Projet MOSCOVA

Mobilité, Sécurité, Concurrence, Vérification et Analyse

Rocquencourt

THÈME 1C

Table des matières

1 Composition de l'équipe **3**

2 Présentation et objectifs généraux **4**

3 Fondements scientifiques **5**
3.1 Le join-calcul 5
3.2 La sémantique des langages de programmation et leur compilation 7
3.3 Interprétation abstraite 7

4 Domaines d'applications **8**

5 Logiciels **8**
5.1 JoCaml 8
5.2 Le join-calcul 1.04 9
5.3 Optimisation de l'allocation et de la synchronisation en Java 9
5.4 Outil de vérification de protocoles cryptographiques 10
5.5 Participation au développement d'Objective Caml 10
5.6 Hevea 10
5.7 Efuns 11

6 Résultats nouveaux **11**
6.1 Le join-calcul 11
6.2 Concurrence et sécurité 12
6.3 Codage et implémentation des 'ambients' dans le join-calcul 12
6.4 Théorie des portées dynamiques 13
6.5 Programmation de réseaux actifs 13
6.6 Typage des librairies dynamiques 13
6.7 Substitutions explicites et logique 14
6.8 Environnement certifié de calcul formel 14
6.9 Analyse statique de protocoles cryptographiques 15
6.10 Amélioration de la compilation du filtrage dans Objective Caml 15

7 Contrats industriels (nationaux, européens et internationaux) **16**
7.1 Projet Marvel 16
7.2 Plateforme d'exécution unniverselle de Microsoft 16

8 Actions régionales, nationales et internationales **17**
8.1 Actions européennes 17
8.2 Collaboration avec R. Wilhelm et P. Laud, Université de Sarrebruck 17
8.3 Accueils de chercheurs étrangers 17

9 Diffusion de résultats **17**
9.1 Animation de la communauté scientifique 17
9.2 Enseignement universitaire 18
9.3 Participation à des colloques, séminaires, invitations 19

10 Bibliographie **20**

Le projet Moscova s'inscrit dans la continuité du projet Para (Parallélisme) dont il reprend les personnels et thèmes de recherche.

1 Composition de l'équipe

Responsable scientifique

Georges Gonthier [DR]

Responsable permanent

Luc Maranget [CR]

Assistante de projet

Sylvie Loubressac [TR]

Personnel Inria

Damien Doligez [CR]

Thérèse Hardin [en délégation de Paris VI]

Jean-Jacques Lévy [DR]

Collaborateur extérieur

Alain Deutsch [PolySpace Technologies]

Ingénieur expert

Bruno Pagano [(avec Cristal)]

Doctorants

Sébastien Ailleret [moniteur X, École Polytechnique, depuis le 1/10/1999]

Bruno Blanchet [moniteur normalien, ENS]

Sylvain Conchon [allocataire MENRT, Paris VII]

Fabrice le Fessant [moniteur X, École Polytechnique]

Virgile Prévosto [allocataire MENRT, Paris VI, depuis le 1/10/1999]

Alan Schmitt [boursier DGA]

Stagiaires

Abheek Anand [IIT New Delhi]

Hichem Bacha [IIT New Delhi]

Dan Fliderbaum [École Polytechnique]

Loïc Le Loarer [École Polytechnique]

Virgile Prévosto [DEA SPP, Paris VI]

Maneesh Varshney [IIT Kanpur]

2 Présentation et objectifs généraux

Le projet Moscova 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 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 bande passante. 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 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

[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.

[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.

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.

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 2000, nous avons poursuivi notre collaboration avec le projet Meije, l'ENST et France Télécom R&D dans le cadre du projet RNRT Marvel.

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 pour 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 : en collaboration avec une firme danoise, G. Gonthier a répondu à un appel d'offre de l'ESA pour le développement de méthodes visant à améliorer la robustesse des logiciels spatiaux.

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 mono-processeur est le système incrémental de Caml et de Ocaml). Avec l'accueil en délégation de Thérèse Hardin, nous nous sommes fortement impliqué dans le projet FoC de calcul formel certifié qu'elle dirige; en particulier Damien Doligez y applique la technologie de preuve modulaire qu'il avait développé pour l'étude des GC et des protocoles de gestion de cache.

En 2000, nous avons deux nouveaux doctorants : Sébastien Ailleret, dirigé par J.-J. Lévy, qui a commencé un travail sur la programmation des réseaux actifs, et Virgile Prévosto, qui travaillera sous la direction de Thérèse Hardin et de Damien Doligez sur la conception du langage utilisateur du système de calcul formel FOC.

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

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

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'une 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, ce qui 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 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 proto-

[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.

type 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 ou 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 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.

[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.

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 : télécommunication, application répartie, 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. Elle a par exemple été utilisée 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 : Abheek Anand, Hichem Bacha, Sylvain Conchon, Fabrice le Fessant, Dan Fliderbaum, Loïc Le Loarer, Virgile Prévosto, Alan Schmitt, Maneesh Varshney.

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).

L'effort de développement d'applications a été poursuivi avec l'implantation d'un service Internet Relay Chat (IRC) en JoCaml, en collaboration avec France Télécom R&D Lannion. De plus, l'interopérabilité de JoCaml a été augmentée par l'ajout de deux interfaces, l'une avec Java RMI (Remote Method Invocation), l'autre avec CORBA par l'intermédiaire d'Orbix.

En 2000, A. Schmitt et F. Le Fessant ont effectué le portage de JoCaml vers les systèmes d'exploitation Windows 32 bits (Windows 98, Windows NT et Windows 2000) [19]. Ce portage s'est en grande partie basé sur le travail réalisé par Xavier Leroy pour le portage de OCaml, et a bénéficié de son expertise. Nous avons été sponsorisés par Microsoft Research qui nous a prêté des machines pour ce travail.

Ce portage donne donc la possibilité d'écrire des programmes distribués et mobiles dans un environnement hétérogène: un programme JoCaml tournant sous Linux peut migrer, en cours d'exécution, vers une machine sous Windows NT.

Enfin, un tutoriel pour JoCaml a été écrit, en partant du tutoriel existant pour le join-calcul v 1. Il est disponible sur le web [21] ainsi que dans les distributions récentes de JoCaml [16].

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 s'est conclu cette année avec la soutenance de la thèse de B. Blanchet [7]. Cette analyse permet 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 Outil de vérification de protocoles cryptographiques

Participant : Bruno Blanchet.

En collaboration avec M. Abadi (Bell Labs Research, Palo Alto), B. Blanchet a commencé la réalisation d'un vérificateur de protocoles cryptographiques décrit au paragraphe §6.9.

L'application de cet outil à des exemples concrets de protocoles (de taille modeste pour l'instant) tirés de la littérature a donné des résultats très encourageants: pour tous les protocoles testés, soit l'outil a prouvé la correction, soit il a exhibé une attaque effectivement dangereuse quand le protocole était incorrect. Le temps de calcul et l'espace mémoire nécessaires à la vérification d'un protocole sont très raisonnables (au plus une minute de calcul sur un Pentium 233 MHz et de l'ordre de 10 Mo de mémoire pour des protocoles échangeant 4 ou 5 messages).

5.5 Participation au développement d'Objective Caml

Participants : Damien Doligez, Luc Maranget.

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

Par ailleurs, le port d'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.

Enfin, L. Maranget a entièrement réécrit le compilateur de filtrage d'Objective Caml en 2000.

5.6 Hevea

Participant : Luc Maranget.

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 1.05 (1.04 l'année dernière). La principale amélioration réalisée cette année est l'ajout des packages latex et l'implémentation de quelques packages (par exemple *listings* pour présenter des programmes dans toute une variété de langages informatiques).

Si on en croit les statistiques du serveur FTP de l'Inria, la version 1.05 d'hevea a été téléchargée environ 2300 fois en sept mois. Une utilisation marquante d'hevea cette année a été la réalisation par John Reppy des Laboratoires Bell des pages de description du problème posé au concours de programmation de la conférence *International Conference on Functional Programming* de cette année.

5.7 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 : Abheek Anand, Sylvain Conchon, Fabrice le Fessant, Georges Gonthier, Hichem Bacha, Luc Maranget, Jean-Jacques Lévy, Alan Schmitt, Maneesh Varshney.

Nous avons amplifié en 2000 nos travaux autour du join-calcul. Sur le plan système, L. Maranget et F. Le Fessant ont amélioré la compilation et l'implémentation des canaux du join-calcul, et F. le Fessant a continué à développer le ramasse-miettes distribué. Des interfaces avec Java RMI et Corba ont été développées par A. Anand at M. Varshney; F. Le Fessant et A. Schmitt ont porté le système sur Windows NT et Windows 2000, en collaboration avec Microsoft. Enfin, un service Internet Relay Chat (IRC) en JoCaml a été développé en collaboration avec France Télécom R&D Lannion.

Sur le plan langage, S .Conchon, avec F. Pottier (Cristal) ont généralisé et étendu dans [18] le système de types du join-calcul de façon à disposer d'un système de types qui puisse prendre en compte des contraintes de type arbitraires. La preuve de correction de ce système utilise une nouvelle technique de preuve dite "semi-syntaxique", qui interprète de manière logique les jugements de typage dérivables dans ce système, comme un ensemble de jugements d'un système sous-jacent disposant d'une preuve de correction syntaxique.

Les recherches menées par L. Maranget avec D. Rémy (Cristal), C. Fournet, et C. Laneve (U. Bologna) sur une extension objet du join-calcul amorcée depuis deux ans a enfin donné lieu à une publication cette année [11].

Les travaux de G. Gonthier et C. Fournet sur les preuves d'équivalences par diagrammes décroissants ont été présentés à HOOTS 2000. G. Gonthier et C. Fournet ont rédigé un cours complet sur le join-calcul pour l'École d'été APPSEM; les notes de cours de cette école ont été publiées en rapport Inria [12].

6.2 Concurrence et sécurité

Participants : Sylvain Conchon, Georges Gonthier.

De nos jours, l'importance de la sécurité des systèmes informatiques oblige les concepteurs de logiciels à porter une attention toute particulière à la confidentialité des informations manipulées par leurs programmes. L'étude théorique des propriétés de sécurité permet de développer des outils d'analyse qui aident à détecter certaines fuites d'informations liées à la conception de ces systèmes.

S. Conchon a développé, en collaboration avec F. Pottier (Cristal) une manière systématique de construire de tels outils à "moindre frais". Leur approche consiste à étendre les systèmes de type existants d'un langage de programmation fonctionnel pour l'analyse de flot d'information. L'avantage de cette approche est qu'elle permet de construire de nouveaux systèmes de type qui réutilisent les caractéristiques des systèmes de type modernes comme l'inférence de type, le polymorphisme et le sous-typage, permettant ainsi une analyse plus fine des propriétés de sécurité, sans alourdir la conception et les preuves de correction de ces outils. Parce que cette approche est à la fois simple et générale, nous avons commencé à l'adapter au join-calcul et au π-calcul. Cependant, le non-déterminisme et les propriétés d'équité des systèmes distribués rendent l'analyse des fuites d'informations plus complexe que dans les systèmes "traditionnels". Ce travail a été présenté à la conférence ICFP [14].

Enfin, G. Gonthier a poursuivi le travail mené en collaboration avec M. Abadi (Lucent Technology) et C. Fournet (Microsoft Research) sur la spécification et la preuve d'abstractions sûres de protocoles cryptographiques. Ils ont présenté leur travail sur la modélisation des primitives d'authentification à Popl 2000 [10], et finalisé le long article développant les fondements de ces travaux [17].

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

Participants : Jean-Jacques Lévy, Alan Schmitt.

En 2000, J.-J. Lévy et A. Schmitt, en collaboration avec C. Fournet (Microsoft Research), ont poursuivi le travail théorique consistant à prouver la correction de l'implémentation du calcul des ambients dans le join-calcul réalisée l'année dernière [22]. Ce travail 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.

Cette preuve utilise des techniques récentes et se décompose en deux grandes parties. La première partie utilise des simulations couplées barbues pour démontrer la correction de l'algorithme implémentant les synchronisations distribuées présentes dans les ambients. La seconde partie est une preuve de bisimulation hybride entre les deux calculs, elle utilise la technique des diagrammes décroissants.

La preuve de correction a été présentée à IFIP-TCS 2000 [13]. Une version longue du papier

est disponible sur le web [20].

6.4 Théorie des portées dynamiques

Participants : Sylvain Conchon, Alan Schmitt.

Les programmes mobiles ont souvent besoin d'ajuster leur comportement à leur localisation courante, ne serait-ce que pour accéder à des ressources locales. Le système JoCaml offre plusieurs solutions à ce problème (utilisation de modules non migrants mais liés localement, serveur de noms locaux), mais ces solution ne sont pas présentes dans le modèle théorique. De plus, l'utilisation de cette liaison dynamique aux ressources locales reste assez difficile à mettre en œuvre pour le programmeur.

S. Conchon et A. Schmitt ont donc commencé un travail théorique ayant pour objectif d'intégrer des canaux dynamiques au join-calcul, ainsi que de fournir un support pour leur utilisation dans le langage. La notion de canal dynamique implique la possibilité d'avoir plusieurs définitions pour un nom donné, ainsi que de spécifier le mécanisme permettant de choisir une définition pour un message donné (ie déterminer à quel moment modifier la liaison dynamique). Le choix que nous étudions actuellement se base sur une liaison qui est modifiée lors de la migration. La sémantique correspondante reste très proche du join-calcul. Nous étudions également un système de types pouvant donner statiquement des garanties sur la présence de ressources.

6.5 Programmation de réseaux actifs

Participant : Sébastien Ailleret.

Le stage de DEA de Sébastien Ailleret a porté sur l'application des techniques de programmation fonctionnelles à la programmation des réseaux actifs. Le but était d'étudier une façon de rendre les réseaux plus flexibles, tout en respectant des impératifs de sécurité. Le principal résultat de ce travail a été la définition du langage FLAN (Functionnal Language for Active Network), qui est destiné à écrire des programmes transportés par les paquets et exécutés lors de leur arrivée sur un routeur pour décider de leur évolution future.

Ce langage est muni d'une sémantique à réductions qui a permis de prouver des propriétés de subject reduction et de terminaison. La deuxième partie du travail a consisté à créer un bytecode typé vers lequel sont compilés les programmes FLAN. Le but recherché de ce bytecode est de disposer de programmes compilés qui conservent les propriétés de sûreté des programmes sources. S. Ailleret va continuer ce travail en thèse.

6.6 Typage des librairies dynamiques

Participant : Georges Gonthier.

Dans la plupart des systèmes d'exploitation récents, les applications accèdent aux bibliothèques de programmes système par le biais de *librairies dynamiques*. Celles-ci permettent à des exécutables différents de coexister en partageant l'implémentation des ressources système, et donc d'interopérer efficacement. Plus généralement, chaque module d'une application est

conçu en supposant qu'il sera exécuté avec une librairie adéquate, et que tous les modules utiliseront la même librairie.

Cette liaison dynamique entre module et librairies est difficile à intégrer dans l'approche classique du typage des modules, fondée sur les types existentiels : il est en effet nécessaire de raisonner sur une représentation explicite du mécanisme d'édition de liens ! Or cette lacune est regrettable, au vu de la fréquence des problèmes dûs à des incompatibilités de librairies.

Au cours de son séjour en juillet-août chez Martin Abadi (Lucent Technologies), G. Gonthier a découvert qu'un typage des modules fondé sur l'axiome du choix, ou plus précisément sur l'opérateur de choix ε de Hilbert, permettait de capturer très précisément la notion de librairies partagées avec leur mécanisme de liaison dynamique. Avec M. Abadi, il a aussi montré que ce mécanisme est en partie incompatible avec l'utilisation de modules d'ordre supérieur, en raison de propriétés particulières de l'opérateur ε; ils ont proposé une restriction syntaxique simple qui permet d'éviter ces incompatibilités. Un article sur ces travaux est en préparation.

6.7 Substitutions explicites et logique

Participant : Thérèse Hardin.

En collaboration avec Gilles Dowek (LogiCal) et Claude Kirchner (Prothéo), T. Hardin a continué ses travaux sur la déduction modulo et les substitutions explicites[8]. Ils ont proposé une formulation de la théorie des types simples dans le cadre de la déduction modulo utilisant le calcul des substitutions explicites comme langage d'expression des fonctions. Le travail a porté sur la rédaction de plusieurs articles, en cours de parution[9] ou de soumission à des revues.

6.8 Environnement certifié de calcul formel

Participants : Damien Doligez, Thérèse Hardin, Virgile Prévosto.

T. Hardin dirige le projet FOC, projet de développement d'un environnement de programmation certifiée pour le Calcul formel. Les participants à ce projet sont S. Boulmé, V. Ménissier-Morain, V. Prévosto et R. Rioboo, membres du LIP6, D. Doligez du projet Moscova, C. Dubois et V. Viguié Donzeau-Gouge du CEDRIC (CNAM). Cet environnement doit offrir à l'utilisateur un langage lui permettant de décrire à la fois les propriétés et les traitements des données qu'il souhaite manipuler. De ce source doivent être extraits un programme OCaml décrivant les traitements et un script Coq décrivant les énoncés et leurs preuves. Cet environnement doit fournir à l'utilisateur une bibliothèque très complète d'unités pré-définies, le source de l'utilisateur pouvant être vu comme l'ajout d'une nouvelle unité à cette librairie.

Pour mesurer la pertinence d'une telle approche, il faut montrer que le texte Ocaml et le texte en Coq peuvent tous deux rester suffisamment proches de la spécification des algorithmes et des données, il faut ensuite montrer que l'ajout d'une nouvelle unité peut être faite de façon sûre, enfin, il faut montrer que cette approche conduit à des programmes efficaces. Pour établir le premier point, plusieurs prototypes de taille conséquente ont été réalisés en Ocaml (essentiellement par R. Rioboo), permettant de dégager une méthodologie d'écriture des unités reposant sur une utilisation assez originale des traits objet et module de Ocaml. Les

temps d'exécution obtenus attestent que cette approche n'est pas pénalisante côté efficacité. Le second point a conduit à décrire formellement le graphe des dépendances entre les unités et à définir des opérations primitives de construction de nouvelles unités. C'est le sujet de la thèse de S. Boulmé, dirigée par T. Hardin et soutenue en décembre 2000, thèse qui fournit une implantation en Coq de la description de la hiérarchie et sa modélisation dans une famille de catégories "à la Cartmell".

La mise en place du langage utilisateur a été entamée (thèse de V. Prévosto, co-encadré par T. Hardin et D. Doligez). La compilation du source utilisateur ne pose pas de difficultés majeures et fournit un résultat identique aux unités développées à la main dans la phase de prototypage. La recherche à mener porte sur la mise en place des énoncés et surtout de leurs preuves, l'idée initiale de s'appuyer sur Coq dès obtention de la syntaxe abstraite a été invalidée au cours des travaux de S. Boulmé (temps de calcul trop longs). En nous appuyant sur les techniques utilisées par D. Doligez pour construire un prouveur pour la logique modale, nous cherchons à dégager une méthode modulaire de construction puis de vérification de la preuve.

6.9 Analyse statique de protocoles cryptographiques

Participant : Bruno Blanchet.

B. Blanchet a passé deux mois et demi à Palo Alto (de août à debut octobre) pour appliquer les techniques d'analyse statique à l'étude et à la vérification de protocoles cryptographiques, dans le cadre d'une collaboration avec M. Abadi.

Ce travail a débouché sur un système de types permettant de vérifier que certaines données sont gardées secrètes par un protocole cryptographique utilisant des canaux à récepteur fixe et de l'encryption à clé publique. Un article concernant ce travail a été soumis.

D'autre part, nous avons commencé la réalisation d'un vérificateur de protocoles utilisant des techniques de réécriture. Ce vérificateur est capable de traiter des protocoles à clé partagée ou publique (encryption et signatures), ainsi que les fonctions de hachage à sens unique et des protocoles de style Diffie-Hellman. Ce vérificateur est capable de prouver qu'une certaine donnée ne peut être connue de l'attaquant.

6.10 Amélioration de la compilation du filtrage dans Objective Caml

Participants : Fabrice Le Fessant, Luc Maranget.

Le filtrage des valeurs est une des constructions les plus expressives des langages fonctionnels. Il permet en particulier de choisir une action en fonction de la structure parfois très complexe d'une valeur, tout en passant en argument de l'action choisie certaines sous-parties de la valeur.

En examinant le résultat de la compilation de certains filtres par le compilateur Objective-Caml, nous avons observé que le code obtenu n'est pas toujours optimal. Nous avons alors développé quelques optimisations, basées sur les principes fondamentaux de la sémantique du filtrage, qui ont permis d'améliorer le code compilé par ce compilateur.

Ces optimisations permettent de diminuer la taille du code généré et de réduire le nombre de tests effectués, ainsi que les accès mémoire, soit par factorisation, soit par élimination des tests inutiles. En particulier, la compilation des filtres disjonctifs a été nettement améliorée.

Enfin, ces optimisations ont conduit à l'extension du système de filtrage du compilateur Objective-Caml, par l'ajout de variables dans les filtres disjonctifs. Ce travail est en cours de diffusion, à travers la soumission d'une publication à ICFP 2001.

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), France Télécom R&D 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 seconde année a été consacrée à l'élaboration du modèle de programmation Marvel. L'originalité de ce modèle est qu'il permet de représenter directement le partitionnement d'un système en plusieurs *domaines* de communication et d'exécution, ainsi que l'évolution dynamique de ces domaines. La mise au point de ce modèle [23] s'est avérée plus difficile que prévue, et nous avons lancé plusieurs actions de recherche (voir §6.4 et §6.6) sur des thèmes collatéraux.

La définition de la plateforme d'exécution a cependant été complétée [24]; la troisième année pourra donc être consacrée à la réalisation du prototype Marvel comme initialement prévu.

7.2 Plateforme d'exécution unniverselle de Microsoft

Participants : Luc Maranget, Bruno Pagano.

Luc Maranget a été partie intégrante dans la collaboration entre Microsoft et les projets Cristal et Moscova.

Ce contrat a duré dix-huit mois de l'été 1999 à décembre 2000. Il portait sur l'étude de faisabilité d'une implantation de Caml sur la plate-forme d'exécution "*universelle*" de Microsoft (commercialement dénommée `.net`). Il est notable qu'une dizaine d'autres équipes de recherche ou d'entreprises étaient partie prenante dans ce projet chacune avec son langage (Eiffel, Scheme, Mercury...).

Le montant du contrat était d'environ un million de francs qui ont été dépensés d'une part en l'achat d'ordinateurs (deux pour le projet) et d'autre part dans l'embauche pour un an de Bruno Pagano, qui a développé le prototype.

Au titre de ce contrat, Luc Maranget s'est rendu à Seatle (USA) avec Bruno Pagano en janvier 2000, pour participer à une semaine de discussion sur les spécifications de la plate-forme `.net`.

Son travail a ensuite consisté essentiellement en une interaction continue avec Bruno Pagano (qui partageait son bureau) qui a su développer son prototype dans les délais.

L'étude a conclu à la difficulté d'implémenter les langages fonctionnels de façon efficace de cette manière. Plus généralement, il paraît hasardeux d'envisager l'exécution de langages arbitraires sur une plate-forme conçue selon un modèle de langage de programmation particulier (ici les langages objets). Les conclusions portent aussi sur un aspect moins étudié faute de temps : l'interaction en pratique entre des programmes écrits dans des langages différents exige une spécification formelle des dites interactions qui dépasse la description niveau machine des objets que les compilateurs sont autorisés à produire ou peuvent consommer.

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 France Télécom R&D 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 Stockholm (3 jours en juin) et à Cambridge (2 jours en septembre).

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

8.2 Collaboration avec R. Wilhelm et P. Laud, Université de Sarrebruck

Participant : Bruno Blanchet.

B. Blanchet a passé un mois à l'université de Sarrebruck (avril 2000) pour travailler sur une analyse statique de programme destinée à coder en ligne des objets dans le langage Java. Cette transformation de programme permettrait de diminuer à la fois le temps d'exécution et les besoins en mémoire des programmes.

8.3 Accueils de chercheurs étrangers

Nous avons reçu James Leifer, Cambridge University, Leslie Lamport, Compaq SRC, Cédric Fournet, Microsoft Research Cambridge, Cosimo Laneve, Bologne.

9 Diffusion de résultats

9.1 Animation de la communauté scientifique

Sébastien Ailleret, Damien Doligez, Robert Harley, Fabrice Le Fessant, Xavier Leroy et Alan Schmitt des projets CRISTAL et MOSCOVA on remporté la deuxième place du concours de programmation organisé dans le cadre d'ICFP 2000, avec un programme écrit en O'Caml.

Le sujet était de réaliser en 72h un programme de synthèse d'images par lancer de rayons, et le concours était ouvert au monde entier. Notons que la première place a aussi été remportée par un programme écrit en O'Caml.

G. Gonthier est vice-président du comité des projets de l'INRIA Rocquencourt, et président de la comission détachement de Rocquencourt. Il représente l'INRIA au comité directeur des Écoles d'été CEA-EDF-INRIA, et a organisé en 2000 une École sur l'application des méthodes formelles à la vérification de systèmes dynamiques, dont les principaux conférenciers étaient E. M. Clarke (Carnegie-Mellon University), D. Harel (Weizmann Institue) et P. Raymond (IMAG). G. Gonthier est également Professeur chargé de cours à temps partiel à l'École Polytechnique. Il a également été membre du comité de programme de la conférence *Practical Applications of Declarative Languages* qui se tiendra en février 2001.

T. Hardin est professeur à l'Université Paris 6, en délégation au projet Moscova 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, point 6, dans le cadre de ses travaux sur la déduction modulo. Elle est membre du comité d'administration de SPECIF.

T. Hardin a été présidente du comité de programme et organisatrice du workshop WESTAPP2000, satellite de la conférence RTA2000, qui a eu lieu en juillet 2000 à Norwich (UK). Ce workshop a reçu le soutien de l'INRIA. Ce workshop a réuni des chercheurs travaillant sur différents aspects des substitutions explicites et des types (logique, programmation, etc.). T. Hardin a également participé au comité de programme de Calculemus, " 8th symposium sur Integration of Symbolic Computation and Mechanized Reasoning", satellite de ISSAC2000.

J.-J. Lévy est professeur à temps partiel à l'Ecole polytechnique, correspondant du concours d'entrée de l'X pour l'informatique (cette année l'X a décidé l'introduction d'une nouvelle épreuve d'informatique pour tous pour le concours 2002), membre des commissions de recrutement de mathématiques et d'informatique de l'Ecole polytechnique. 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.

L. Maranget est membre élu de la commission d'évaluation de l'Inria. À ce titre, il s'est rendu aux évaluations des programmes de l'Inria, et a été membre du jury de recrutement d'un Chargé de recherche en bio-informatique (en mai). L. Maranget est aussi Chargé d'enseignement à l'École Polytechnique, et correcteur du concours d'entrée. L. Maranget était membre du comité de programmation de la conférence *Journée Francophone des Langages Applicatifs*, à ce titre il a assisté à la conférence en Janvier 2000. Il a également été membre du comité de programme de la conférence *Practical Applications of Declarative Languages*, qui se tiendra à Las Vegas les 11 et 12 mars 2001.

9.2 Enseignement universitaire

Notre projet participe aux enseignements suivants:

- Algorithmes et programmation à l'Ecole polytechnique (S. Ailleret, F. le Fessant, G. Gonthier, J.-J. Lévy, L. Maranget, A. Schmitt). G. Gonthier y a enseigné (40h) et a été responsable des projets informatique ; S. Ailleret, F. le Fessant, L. Maranget et A. Schmitt y

ont assuré les travaux pratiques. J.-J. Lévy est responsable du tronc commun 2000-2001 et y enseigne [15].

- Introduction à la programmation à l'Ecole polytechnique. J.-J. Lévy a fait ce cours pendant une semaine les 22-26 mai 2000, et le 30-31 octobre aux journées X-UPS avec les professeurs de classes préparatoires(cf. `http://w3.edu.polytechnique.fr/informatique/Init0/semaine0.html`)

- Langages de programmation à l'Ecole polytechnique (J.-J. Lévy, A. Schmitt). J.-J. Lévy a donné ce cours (40h) dans le cadre de la majeure de seconde année de l'École Polytechnique, dont il est le responsable (avec G. Dowek en 2000-2001); A. Schmitt l'y assistait pour les travaux dirigés.

- Compilation des langages de programmation à l'Ecole polytechnique. L. Maranget a donné ce cours (40h) dans le cadre de la majeure de seconde année de l'École Polytechnique.

- J.-J. Lévy a participé aux journées Algorithme et Programmation, CIRM, Luminy, 9-11 mai 2000, avec les professeurs d'informatique en classes préparatoires.

- DEA Sémantique, preuves et programmation : Cours de lambda-calcul (J.-J. Lévy, 20h), et de Concurrence et mobilité (G. Gonthier, 20h).

En collaboration avec V. Viguié Donzeau-Gouge, T. Hardin a supervisé la mise en place du 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'intêret des méthodes formelles commence à être reconnu. Elle a encadré plusieurs stages (FT-RD Lannion, Trusted Logic, Price-WaterHouse, Surlog) et a donné le cours de sémantique des programmes séquentiels.

T. Hardin a été rapporteur de la thèse de Horatiu Cirstea (LORIA), soutenue en octobre 2000. J.-J. Lévy est rapporteur de l'habilitation de Ian Mackie.

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

9.3 Participation à des colloques, séminaires, invitations

POPL 2000, 18-22 janvier Boston (USA) : Présentation de G. Gonthier (avec M. Abadi et C. Fournet), participation de J.-J. Lévy et A. Schmitt.

JFLA 2000, Mont Saint-Michel, 31 janvier 1er février Présentation de T. Hardin (avec la société Surlog), participation de L. Maranget (comité de programme).

Confer-2 workshops : KTH, Stockholm, 12-14 juin 2000 (présentation de S. Conchon, participation de F. Le Fessant, J.-J. Lévy) et Cambridge, 13-15 septembre 2000 (présentation de S. Conchon, participation de J.-J. Lévy, L. Maranget, A. Schmitt).

RTA 2000, Norwich (GB) 10–12 juillet. Participation de T. Hardin.

École d'été CEA-EdF-INRIA, Le Bréau (78), 26 juin–7 juillet. Organisée par G. Gonthier; participation de D. Doligez.

Workshop on Foundations and Computations, Edinbourg (GB), juillet. Conférence invitée de T. Hardin (*The calculus of contexts of lambda-sigma*) à ce workshop organisé par F. Kamareddine à la Hyatt University of Edimburgh.

IFIP TCS'2000 (IFIP International Conference on Theoretical Computer Science), 17-19 août, Sendai, Japon. Présentation de J.-J. Lévy et A. Schmitt.

École d'été APPSEM, Caminha (Portugal), 9–15 septembre. Cours de G. Gonthier.

Security through Analysis and Verification, Dagstuhl (Allemagne) 10–15 décembre; exposé de B. Blanchet "Secrecy Types for Asymmetric Communication".

G. Gonthier a donné une conférence invitée sur le join-calcul à l'Irisa le 12 avril, et sur le typage des librairies dynamiques à Chevaleret le 20 novembre.

G. Gonthier a été invité deux mois (juillet-août) au Bell Labs Silicon Valley Research Center par M. Abadi; B. Blanchet y a également effectué un séjour de août à octobre. B. Blanchet a aussi été invité un mois an avril à l'université de Sarrebruck.

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.

Thèses et habilitations à diriger des recherches

[7] B. Blanchet, *Analyse d'échappement. Applications à ML et Java^{TM}*, thèse de doctorat, École Polytechnique, 7 décembre 2000.

Articles et chapitres de livre

[8] G. Dowek, T. Hardin, C. Kirchner, « Higher-order unification via explicit substitutions », *Information and Computation 157*, 2000, p. 183–235.

[9] G. Dowek, T. Hardin, C. Kirchner, « HOL-$\lambda\sigma$: an intentional first-order expression of higher-order logic», *Mathematical Structures in Computer Science*, 11, 2001, p. 1–25, A paraître.

Communications à des congrès, colloques, etc.

[10] 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.

[11] L. M. C. Fournet, C. Laneve, D. Rémy, « Inheritance in the Join Calculus », *in : Proc. of the 19th Foundations of Software Technology and Theoretical Computer Science, Delhi*, décembre 2000.

[12] C. Fournet, G. Gonthier, « The Join Calculus: a Language for Distributed Mobile Programming », *in : Proc. of the International summer school on applied semantics - APPSEM'2000*, G. Barthe, P. Dybjer (éditeurs), septembre 2000.

[13] C. Fournet, J.-J. Lévy, A. Schmitt, « An Asynchronous, Distributed Implementation of Mobile Ambients».

[14] F. Pottier, S. Conchon, « Information Flow Inference for Free », *in : Proceedings of the the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, p. 46–57, Montréal, Canada, septembre 2000, `http://pauillac.inria.fr/~~conchon/publis/fpottier-conchon-icfp00.ps.gz`.

Rapports de recherche et publications internes

[15] R. Cori, J.-J. Lévy, *Algorithmes et Programmation*, Ecole polytechnique, octobre 2000, `http://w3.edu.polytechnique.fr/informatique/TC/polycopie-1.6`.

Divers

[16] « The JoCaml system», Available from `http://pauillac.inria.fr/jocaml/`, 2000.

[17] M. Abadi, C. Fournet, G. Gonthier, «Secure Implementation of Channel Abstractions», à paraître dans Information and Computation, 2001, `http://pauillac.inria.fr/join`.

[18] S. Conchon, F. Pottier, « JOIN(X): Constraint-Based Type Inference for the Join-Calculus », Submitted to *ESOP'01*, octobre 2000, `http://pauillac.inria.fr/~~conchon/publis/conchon-fpottier-esop01.ps.gz`.

[19] F. L. Fessant, A. Schmitt, « JoCaml on Windows NT », Available from `http://pauillac.inria.fr/jocaml/jocnt.html`, 2000.

[20] C. Fournet, J.-J. Lévy, A. Schmitt, « A Distributed Implementation of Ambients», Long version of this paper, available from `http://join.inria.fr/ambients.html`, 1999.

[21] C. Fournet, L. Maranget, A. Schmitt, « The JoCaml tutorial and reference manual », Available from `http://pauillac.inria.fr/jocaml/htmlman/index.html`, 2000.

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

[23] A. S. Gérard Boudol, J.-B. Stefani, « Marvel Programming Model v1 », Projet RNRT Marvel, décembre 2000.

[24] M. Lacoste, F. Le Fessant, K. Milsted, « Machine Répartie Virtuelle et Langage pour Objets Mobiles», Projet RNRT Marvel, octobre 2000.