INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Projet MOSCOVA*

*Mobilité, Sécurité, Concurrence, Vérification et Analyse*

*Rocquencourt*

THÈME 1C

# Table des matières

*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]
**Conseiller extérieur**
Dave B. Mac Queen [depuis le 01/09/2001]
**Post-Doctorant**
James Leifer
**Doctorants**
Sébastien Ailleret [moniteur X, École Polytechnique]
Bruno Blanchet [moniteur normalien, ENS, jusqu'au 31/08/2001]
Sylvain Conchon [allocataire MENRT, Paris VII]
Fabrice le Fessant [boursier Inria, jusqu'au 30/03/2001]
Virgile Prévosto [allocataire MENRT, Paris VI]
Alan Schmitt [boursier DGA]
**Stagiaires**
Ma Qin [DEA, SPP, Paris VI]
Rajneesh Kumar Singh [ITT, New-Dehli]
Gilles Peskine [ENS Ulm]

# 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 $\pi$-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 1.05 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. En 2001, nous avons achevé 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'A. Deutsch. Avec le départ en disponibilité d'A. Deutsch pour la création de PolySpace Technologies en 2000, puis le recrutement par le CNRS de B. Blanchet, cette activité sera probablement très réduite dans les années à venir. Toutefois, B. Blanchet a été très actif cette année, et le projet conserve une activité de conseil dans le domaine du code embarqué, concrétisée par un contrat avec la société danoise Terma.

Enfin, notre projet maintient une certaine activité dans les preuves formelles, initialisée par la longue preuve de sûreté 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

[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 $\pi$-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.

modulaire qu'il avait développé pour l'étude des GC et des protocoles de gestion de cache.

En 2001, nous avons deux nouveaux doctorants : Gilles Peskine dirigé par J.-J. Lévy, sur la modélisation des pannes, et Ma Qin co-dirigée par L. Maranget sur les objets dans la programmation concurrente. Ces deux étudiants ont réalisé leur stage de DEA au sein du projet cette année.

# 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'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.

[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.

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 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 extension 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.

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** : Sylvain Conchon, Fabrice le Fessant, James Leifer, Jean-Jacques Lévy, Luc Maranget, Alan Schmitt.

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

Seconde implémentation du join-calcul, JoCaml est une extension du langage 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

[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.

de communication et de synchronisation, et des agents mobiles du join-calcul, et d'objets distribués.

Nous choisissons de faire porter tous nos efforts sur cette seconde incarnation du join-calcul en tant que langage de programmation, car en la construisant à partir de Caml, nous bénéficions d'emblée d'un langage de programmation puissant. JoCaml est utlisé, entre autres, à France Télécom R&D (P. Crégut) et non rencontrons épisodiquement ces utilisateurs privilégiés.

Tandis que la maintenance et la diffusion de JoCaml se poursuivent (assurée par A. Schmitt et F. Le Fessant), nous avons commencé cette année le développement d'une nouvelle version. Le but est cette fois de stabiliser et pérenniser JoCaml en intégrant les modifications nécessaires aux sources actifs de Caml au lieu, comme dans le JoCaml de la première génération, de prendre une copie de ces sources. Dès lors, les évolutions de Caml profiteront aussi à JoCaml.

Certaines des évolutions de Caml depuis sa version 1.07 autorisent cette démarche sans modifier outre mesure Caml. D'autres sont nécessaires comme, par exemple, la possibilité de transmettre des fonctions dynamiquement d'un programme Caml à l'autre ; ou l'introduction d'un peu de typage lors de l'exécution. Cette année, A. Schmitt à participé à l'encadrement du stage de fin d'étude de l'élève polytechnicien Tomasz Blanc qui a porté sur cette première question, tandis que J. Leifer anime un groupe de travail sur la seconde, en collaboration avec P. Weiss et J. Furuse du projet CRISTAL. Ces évolutions devront, à terme, profiter à Caml.

## 5.2 Le join-calcul 1.05

**Participant** : Luc Maranget.

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

Ce système, toujours disponible, ne connaîtra à priori plus d'évolution. L'ensemble de nos efforts de développement se concentrant sur JoCaml. Outre son intérêt historique, cette première incarnation du join-calcul en tant que langage de programmation a fourni la base de la documentation de JoCaml.

## 5.3 Outil de vérification de protocoles cryptographiques

**Participant** : Bruno Blanchet.

La réalisation d'un vérificateur de protocoles cryptographiques, commencée l'an dernier en collaboration avec M. Abadi, s'est poursuivie cette année. J'ai optimisé l'algorithme, réduisant le temps de calcul d'un facteur environ 100, et j'ai réalisé des extensions, permettant en particulier de modéliser le compromis de certaines clés de session. En effet, une faiblesse de certains protocoles est que le compromis d'une clé de session suffit pour que l'attaquant puisse découvrir les secrets de toutes les sessions. Cette faiblesse peut maintenant être détectée par notre vérificateur.

## 5.4 Participation au développement d'Objective Caml

**Participants** : Damien Doligez, Luc Maranget.

D. Doligez et L. Maranget assurent la maintenance des parties d'Objective Caml qu'ils ont

écrit (Glanage de cellules et compilation du filtrage).

Par ailleurs, D. Doligez a porté Objective Caml sur le système d'exploitation Mac OS X.

### 5.5 Hevea

**Participant** : Luc Maranget.

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

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. Le Lâcher d'une nouvelle version (1.06) est programmé pour décembre 2001. Si on en croit les statistiques du serveur FTP de l'Inria, la version 1.05 d'hevea a été téléchargée environ 4500 fois depuis un an.

### 5.6 Taml

**Participant** : Sébatien Ailleret.

Ce prototype est disponible sur le réseau en `http://pauillac.inria.fr/~ailleret/recherche/taml/`.

Taml est un compilateur de ML vers du bytecode typé. Il se compose d'un compilateur et d'un vérificateur. Il produit du code compatible avec OCaml, ce qui permet d'utiliser son environnement d'exécution.

Ce compilateur peut déjà gérer un sous-ensemble de Caml comprenant le polymorphisme, les références, la définition et l'utilisation de types concrets paramétrés ainsi que la définition et l'utilisation d'exceptions.

## 6 Résultats nouveaux

### 6.1 Le join-calcul

**Participants** : Sylvain Conchon, Georges Gonthier, Rajneesh Kumar Singh, Fabrice Le Fessant, Jean-Jacques Lévy, Luc Maranget, Gilles Peskine, Ma Qin, Alan Schmitt.

Nous avons poursuivi en 2001 nos travaux autour du join-calcul.

Sur le plan système, pendant son stage de 2 mois (mai-juin), Rajneesh Kumar Singh (IIT – Dehli) a conçu et implanté en JoCaml un petit serveur ftp avec réplication. Cela a imposé de programmer la récupération d'erreur sur des serveurs de transferts de fichiers (programmés en JoCaml) et de concevoir une politique de remplacement [23]. F. Le Fessant, auteur du système JoCaml, soutient sa thèse cette année [7].

Sur le plan langage, S .Conchon, avec F. Pottier (Cristal) ont généralisé et étendu dans [15] 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 (Microsoft-Research), et C. Laneve (U. Bologna) sur une extension objet du join-calcul amorcées depuis trois ans, se poursuivent, une soumission au journal *TOPLAS* est cours, tandis que le stage de DEA de Ma Qin (co-encadré par L. Maranget) a porté sur l'étude de l'implémentation du calcul de classes proposé [19].

Notre projet aborde aussi la résistance aux pannes, J.Leifer (qui a soutenu sa thèse [8, 10] alors qu'il collaborait déjà avec nous), est maintenant chercheur post-doctorant dans notre équipe. Il a commencé l'étude et l'implémentation en Caml d'algorithmes distribués robustes, tels que la diffusion sûre (*relieable broadcast*) et le consensus distribué, en utilisant les oracles (*failure detectors*) de S. Toueg. Il explore notamment la possibilité d'implémenter au niveau d'UDP et collabore avec P. Sewell de l'Université de Cambridge sur ce thème.

Dans ce même thème, le stage de DEA de G. Peskine [20] a porté sur la simplification de la machine abstraite de JoCaml (MAJ). Il s'agit d'un modèle intermédiaire entre le join-calcul (avec la mobilité) et l'implémentation JoCaml. Contrairement au join-calcul, la MAJ explicite le routage des messages entre les machines participant à l'application distribuée. La MAJ modélise ainsi toutes les communications entre machines. La simplification de ce modèle est un pré-requis à l'étude des pannes.

Enfin, G. Gonthier a rédigé avec C. Fournet deux articles de revue à partir de leur présentations à ICALP 98 et APPSEM 2000, ce sont des travaux de synthèse sur le join-calcul.

## 6.2 Concurrence et sécurité

**Participant** : Sylvain Conchon.

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.

S. Conchon a continué dans cet axe il a écrit un article généralisant cette approche et l'a soumis à ESOP'02 (cf. `http://pauillac.inria.fr/~conchon/publis/jcflow-long.ps.gz`) Cet article présente une manière systématique d'étendre le système de type d'un langage de programmation distribué avec des annotations de sécurité qui permettent de vérifier une propriété de flot d'information, la non-interférence, basée sur une bisimulation.

Le nouveau système de type ainsi obtenu peut détecter, par exemple, des fuites d'infor-

mation dues aux contraintes de synchronisation sur des ressources distribuées, qui ne sont pas détectés, d'un manière satisfaisante, si la propriété de non-interférence est fondée sur une équivalence de tests.

De plus, notre méthode pour ajouter des annotations de sécurité permet très facilement de prouver cette propriété de non-interférence pour des systèmes de types avec polymorphisme et l'inférence de type.

Nous formalisons notre travail à l'aide du join-calcul, un modèle de programmation distribué, dérivé du pi-calcul, et utilisé comme base formel pour l'implémentation de plusieurs langages distribués comme par exemple Jocaml et Functional Nets.

### 6.3 Programmation de réseaux actifs

**Participant** : Sébastien Ailleret.

S. Aillert poursuit son travail de thèse sur l'application des techniques de programmation fonctionnelles à la programmation des réseaux actifs.

Typer le bytecode Caml permet d'assurer à un utilisateur que le code compilé qu'il possède s'exécutera sans erreur, et ceci sans aucune connaissance du code source du programme ni de la façon dont il a été compilé.

Dans les langages de programmation fortement typés, la sûreté de l'exécution d'un programme est garantie par les propriétés de typage du langage source. Toutes les vérifications étant entièrement statiques, les informations de type sont donc logiquement effacées à la compilation car inutiles ensuite. En revanche, en présence de code mobile, il est impératif de garantir la correction du programme lors de sa réception, car on ne dispose plus du code source dont le typage assurait la sûreté à l'exécution. Pour ce faire, notre approche consiste à ajouter au bytecode quelques informations de type, qui, sans rien dévoiler du code source du programme, permettent de vérifier rapidement qu'un code compilé possède les mêmes garanties de sûreté que le langage source.

Un article a été soumis à l'édition 2002 des JFLA.

### 6.4 Substitutions explicites et logique

**Participant** : Thérèse Hardin.

En collaboration avec Gilles Dowek et Claude Kirchner, T. Hardin a continué ses travaux sur la déduction modulo, les langages avec lieurs[18] et les substitutions explicites. 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[9]. Le travail a porté sur la rédaction de plusieurs articles, en cours de parution ou de soumission à des revues.

### 6.5 Environnement certifié de calcul formel

**Participants** : Damien Doligez, Thérèse Hardin, Virgile Prévosto.

T. Hardin dirige le projet de développement d'un environnement de programmation certifiée

pour le Calcul formel, appelé FOC. Les participants de ce projet sont S. Boulmé, M. Jaume, S. Fechter, V. Ménissier-Morain et R. Rioboo, membres du LIP6, C. Dubois et V. Viguié Donzeau-Gouge du CEDRIC (CNAM), ainsi D. Doligez du projet Moscova et V. Prévosto, co-encadré par D. Doligez et T. Hardin. 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 (dites *espèces* et *collections*), le source de l'utilisateur pouvant être vu comme l'ajout d'une nouvelle unité à cette librairie. L'ajout se fait en combinant les paradigmes de la programmation objet et par modules.

Le rapport qui suit porte plus spécifiquement sur les activités directement liées au projet MOSCOVA (abstraction faite de ce que T. Hardin dirige l'ensemble de FOC).

Côté programmation, la bibliothèque doit fournir des programmes à la fois corrects et efficaces. L'efficacité est à prendre en compte à différents niveaux : expressivité, distance réduite entre le source et sa spécification de manière à faciliter les preuves, portabilité sur différentes architectures, efficacité à l'exécution. Nous abordons cette question en fixant une discipline d'utilisation des traits objet et modules de OCaml (après l'écriture d'un certain nombre de prototypes par R. Rioboo). Le développement du langage de FOC est effectué par D. Doligez et V. Prévosto. A partir de la spécification en Coq et des études déterminant la cible pour la programmation, ils ont défini une syntaxe concrète, très expressive, qui permet de regrouper dans un même texte des déclarations, définitions, énoncés et preuves.

La syntaxe interdit tout écart à la discipline de programmation et les outils offerts pour la création de nouvelles espèces implantent les spécifications des mécanismes d'extension spécifiés dans Coq. Ces propriétés du source FOC sont garanties d'abord par une synthèse/vérification des types et des mécanismes d'importation, ensuite par une analyse statique de ces dépendances, fondée sur la spécification en Coq. Une fois analysé, le source FOC est compilé vers un source OCaml, qui respecte la discipline de programmation. L'efficacité du code ainsi obtenu est excellente.

Alors que la compilation vers OCaml peut être considérée comme achevée, la construction de la preuve demande encore un certain nombre de développements, tant au point de vue théorique qu'au point de vue pratique. Sur le plan théorique, D. Doligez et T. Hardin étudient les propriétés d'une extension de la logique du premier ordre introduite par D. Doligez, utilisée pour effectuer la preuve d'un énoncé sous hypothèses. Avec V. Prévosto, ils étudient également certaines restrictions des langages de types dépendants, pouvant suffire à construire une preuve complète. Il reste de nombreuses extensions à effectuer sur le plan pratique, en particulier le traitement de l'induction.

En l'état actuel, la librairie fournit les algorithmes standard en en calcul formel (algèbre commutative, arithmétiques entières et rationnelles, polynômes) ainsi que certains algorithmes avancés (algorithme de Loos pour le calcul des résultants, algorithme de Cantor-Zassenhaus revu par Quitte-Naudi pour la factorisation des polynômes à une variable sur un corps fini). Il faut cependant noter que nombre de preuves restent à faire pour ces différentes espèces et collections. Leur complexité justifie les efforts faits par D. Doligez et V. Prévosto pour assurer une bonne ergonomie de FOC.

### 6.6 Analyse statique de protocoles cryptographiques

**Participant** : Bruno Blanchet.

D'une part, B. Blanchet a continué la réalisation d'un vérificateur de protocoles fondé sur la programmation logique : le protocole est traduit en clauses de Horn, telles que si un certain fait ne peut pas être prouvé à partir de ces clauses, alors le secret de la donnée correspondante est préservé par le protocole (même en présence d'un attaquant). La principale originalité de ce travail est que le vérificateur est capable de traiter un nombre quelconque (non borné) de sessions du protocole, même en parallèle. Ce travail a donné lieu à une publication à CSFW'2001 [14], la version longue est à paraître dans le numéro special CSFW du *Journal of Computer Securit y*, et une implantation a été réalisée (voir section logiciels).

D'autre part, en collaboration avec M. Abadi, B. Blanchet a poursuivi le travail sur la vérification de protocoles cryptographiques par typage. L'an dernier, ils avaient conçu un système de types pour prouver des propriétés de secret dans les protocoles utilisant le chiffrement à clé publique. Ce système a été publié cette année [11], et la version longue est à paraître dans le numéro spécial FoSSaCS'2001 de *Theoretical Computer Science*. De plus, cette année, le système a été étendu pour obtenir un traitement générique et flexible de nombreuses primitives cryptographiques (dont chiffrement, signatures, fonctions de hachage, ...).

En outre, les deux approches mentionnées ci-dessus ont été comparées, et leur équivalence démontrée : on peut prouver par le système de types qu'une donnée est gardée secrète si et seulement si cela est détecté par le vérificateur fondé sur la programmation logique. Ce résultat, ainsi que le système de types étendu, sont à paraître à POPL'2002 [12].

### 6.7 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, réalisé l'année dernière, a fait l'objet d'une publication cette année [16].

# 7 Contrats industriels (nationaux, européens et internationaux)

## 7.1 Projet Marvel

**Participants** : Sylvain Conchon, Georges Gonthier, 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.

Initialement, nous pensions que notre modèle d'évaluation concurrente (le join-calcul) serait au cœur du modèle de programmation Marvel, et que nos travaux au sein de Marvel porteraient plutôt sur le développement langage. Or, au cours de nombreuses réunions, il est apparu qu'il n'y avait pas consensus autour du join-calcul. De façon un peu inattendue notre contribution de cette dernière année a porté sur l'élaboration théorique du modèle de programmation Marvel. En particulier, A. Schmitt a passé deux mois à l'Inria Rhône-Alpes, où, en compagnie J.-B. Stéfani, ils ont continué le $\kappa$-calcul (modèle de programmation Marvel). Le modèle de programmation donne lieu à une soumission à ECOOP 2002 [21] et est détaillé dans [22].

## 7.2 Méthodologie pour la robustesse des logiciels spatiaux

**Participants** : Georges Gonthier, Jean-Jacques Lévy.

La proposition conjointe de Moscova et de la société danoise Terma A/S à la proposition de l'ESTEC (Agence Spatiale Européenne) pour une étude sur une méthodologie pour la robustesse des logiciels spatiaux, émise en mars 2000, a finalement été acceptée en mai 2001. G. Gonthier et J.-J. Lévy ont rédigé la principale contribution de l'Inria, une étude de l'état de l'art, de mai à octobre. Il y a eu deux déplacements à l'ESTEC à Noorwijk (Pays-Bas) dans le cadre de ce contrat.

## 7.3 CoordinAtion et Répartition des Applications Multiprocesseurs en OCaml

**Participants** : Georges Gonthier, Gilles Peskine.

Cette Action Concertée Initiative 2001, dirigée par Gaëtan Hains (U. Orléans), et comprenant également l'équipe PPS de Paris VII, France Télécom R&D Bretagne, et les projets Cristal et Moscova de l'Inria Rocquencourt, concerne le développement de bibliothèques pour le calcul haute-performance et globalisé en OCaml, avec, entre autres JoCaml comme implémentation de référence. Elle a été acceptée et a démarré en Octobre 2001.

# 8 Actions régionales, nationales et internationales

## 8.1 Actions européennes

En 2001, nous avons contribué au lancement du projet PEPITO (*Peer to Peer, Implementation and Theory*), accepté dans l'appel d'offre *Global Computing* de Esprit. Ce projet regroupe des équipes de KTH (Seif Haridi, coordinateur), Cambridge (Peter Sewell), EPFL (Martin Odersky), SICS (Per Brand), UCL (Peter van Roy). Ce projet traitera des aspects théoriques et pratiques des applications symétriques et distribuées, avec mobilité. Il contribuera à fédérer les efforts déjà développés autour de Mozart (Mobile Oz), Jocaml et Funnel (Functional Nets). Il pourra aussi être un point de rencontre entre les communautés des langages de programmation et des systèmes distribués.

Nous avons eu une réunion de préparation à Bruxelles (fin janvier), la proposition a été soumise en avril, acceptée en juin, le projet démarre au 1 janvier 2002.

## 8.2 Accueils de chercheurs étrangers

Nous avons reçu Peter Sewell, Cambridge University, Leslie Lamport, Compaq SRC, Martin Abadi, Compaq SRC, Cédric Fournet, Microsoft Research Cambridge, Cosimo Laneve, Université de Bologne et le Professeur K. Gopinath (IISc-Bengalore).

# 9 Diffusion de résultats

## 9.1 Animation de la communauté scientifique

S. Ailleret, S. Conchon, M. Guesdon, F Le Fessant, J. Leifer et T. Hirschovitz des projets CRISTAL et MOSCOVA on remporté la troisième place ex-equo du concours de programmation organisé dans le cadre d'ICFP 2001, avec un programme écrit en O'Caml. Le sujet était de réaliser en 72h un optimiseur d'un langage de *Mark-Up* inspiré de HTML.

D. Doligez, L. Maranget et P. Weiss (CRISTAL) ont organisé cette compétition, qui a attiré 140 concurrents, totalisant 263 soumissions (voir `http://pauillac.inria.fr/cristal/ICFP2001/prog-contest/`).

B. Blanchet a donné un exposé au Colloquium Junior de l'UR, le 27 février 2001,"Interprétation abstraite. Application à l'allocation en pile et à l'élimination des synchronisations en Java".

G. Gonthier est vice-président du comité des projets de l'INRIA Rocquencourt, et président de la commission détachement de Rocquencourt, et s'est occupé de l'accueil des canddidats du corps des Télécom et de la campagne de recrutement en 2001 à ce titre. Il représente l'INRIA au comité directeur des Écoles d'été CEA-EDF-INRIA G. Gonthier est également Professeur chargé de cours à temps partiel à l'École polytechnique.

T. Hardin anime, en collaboration avec V. Donzeau-Gouge (CNAM), un groupe de travail sur l'utilisation des systèmes d'aide à la preuve (Coq, PVS, B) et plus généralement des méthodes formelles dans la spécification et la programmation. T. Hardin est vice-présidente de SPECIF depuis juin 2001. Elle est chargée de la recherche, en collaboration avec B. Lorho.

J.-J. Lévy est professeur à temps partiel à l'École polytechnique, vice-président en charge du recrutement du département informatique de l'X, 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'École polytechnique. 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. L. Maranget est aussi Professeur chargé de cours à temps partiel à l'École Polytechnique, et correcteur du concours d'entrée. Il a participé au comité de programme de la conférence *Practical Applications of Declarative Languages*, qui s'est tenue à Las Vegas les 11 et 12 mars 2001.

## 9.2 Enseignement universitaire

Notre projet participe aux enseignements suivants :

- Algorithmes et programmation à l'École 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 [17].
- Introduction à la programmation à l'École 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://www.enseignement.polytechnique.fr/informatique/Init0/semaine0.html`)
- Langages de programmation à l'École polytechnique (J.-J. Lévy). J.-J. Lévy est responsable (avec G. Dowek) de ce cours (40h) dans le cadre de la majeure de seconde année de l'École Polytechnique.
- Compilation des langages de programmation à l'École polytechnique. L. Maranget a assuré les travaux pratiques de ce cours (40h) dans le cadre de la majeure de seconde année de l'École Polytechnique.
- L.Maranget a participé aux journées Algorithme et Programmation, CIRM, Luminy, mai 2001, avec les professeurs d'informatique en classes préparatoires.
- Journées X-UPS, 30-31 octobre. J.-J. Lévy a organisé ce cours destinés aux professeurs de classes préparatoires, en vue de la préparation à la nouvelle petite épreuve d'informatique pour tous du concours d'admission (cf. `http://www.enseignement.polytechnique.fr/informatique/concours/`).
- DEA Sémantique, preuves et programmation : Cours de lambda-calcul (J.-J. Lévy, 20h), de Concurrence et mobilité (G. Gonthier, 20h), d'Analyse des échappements pour ML et Java (B. Blanchet).

En collaboration avec V. Viguié Donzeau-Gouge, T. Hardin a géré 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'intérêt des méthodes formelles commence à être reconnu. Elle a encadré plusieurs stages (Gem-Plus, Surlog) Logic, Price-WaterHouse, Surlog) et a donné le cours de sé-

mantique des programmes séquentiels. Elle a également participé à l'encadrement de stagiaires de différents niveaux dans le projet FOC.

J.-J. Lévy a été rapporteur de l'habilitation de Ian Mackie (École polytechnique) et *External examiner* de thèse de James Leifer (Cambrige, GB). J.-J. Lévy est co-directeur (avec Pierre Crégut, France télécom, Lannion) de la thèse en cours de G. Chatelet, sur l'addition d'un typage fort et de primitives de mobilité au langage Erlang.

T. Hardin a été membre des jurys de thèse de M. Mayero et D. Delahaye en décembre 2001.

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

## 9.3 Participation à des colloques, séminaires, invitations

### Participation à des congrès

POPL 2001, 17-19 janvier, à Londres : participation de S. Ailleret, B. Blanchet, J.-J. Lévy et A. Schmitt.

PADL 2001 et SAC 2001, 11–14 mars, à Las Vegas (USA) : participation de L. Maranget (comité de programme PADL).

ETAPS'2001, 2–6 avril, à Gênes, Italie : exposé de B. Blanchet, participation de D. Doligez.

CSFW'2001, 11–13 juin, à Keltic Lodge (Canada) : exposé de B. Blanchet.

IJCAR 2001, juin, à Sienne Italie : participation de T. Hardin (membre du comité de programme de CALCULEMUS 2001, *Symposium on integration of Symbolic Computation and Mechanized Reasoning*, intégré au groupe de conférence IJCAR)

ConCord 2001, 6–8 juin, aux Îles Lipari (Italie) : participation de J.-J. Lévy.

SAS 2001, 16–18 juillet, à Paris : exposé invité de B. Blanchet.

ICALP 2001, et Bohm's theorem workshop, 13 juillet, à Heraklion (Grèce) : participation de J.-J. Lévy (organisteur du workshop) (cf. `http://pauillac.inria.fr/~levy/both2001.html`).

ICFP 2001, 2–5 septembre, à Florence (Italie) : exposé de L. Maranget, participation de D. Doligez (compte rendu du concours de programmation) et de F. Le Fessant.

MKM 2001 (Mathematical Knowledge Management), septembre, à Linz (Autriche), présentation du projet FOC par T. Hardin.

École d'été "Foundations of Wide Area Network Programming", 1–15 juillet 2001, aux iles Lipari (Italie), participation de S. Ailleret, J. Leifer, Ma Qin, et G. Peskine.

### Visites

J. Leiffer a rendu viste à R. Milner, University of Cambridge (GB) du 18 au 20 juin et à B. Pierce, University of Pensylvania (USA), du 25 novembre au 2 décembre .

D. Doligez à passé un mois à Compaq Systems Research Center (Palo Alto, USA) pour participer à la specification et à la conception du protocole de mémoire virtuelle partagée du processeur EV8.

T. Hardin a été invitée par C. Muñoz, chercheur à ICASE (Hampton, USA), pour un séjour de quinze jours. Elle a présenté les travaux du projet FOC et travaillé sur la certification d'algorithmes utilisant des méthodes de calcul formel pour la gestion du contrôle aérien.

### Exposés

Exposé de J. Leifer au séminaire PPS (Paris VII), sur les systèmes de transition étiquetés.

Exposés à l'Inria Loraine de B. Blanchet le 12 janvier 2001 (sur l'analyse d'échappements) et de L. Maranget le 5 décembre 2001 (sur la compilation du filtrage de ML).

Exposé au séminaire Coq-Cristal-Moscova le 30 mars 2001 de B. Blanchet (sur la vérification de protocoles cryptographiques).

## 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, `ftp://theory.lcs.mit.edu/pub/jacm/jacm.bib`.

[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] F. Le Fessant, *JoCaml : conception et implémentation d'un langage à agents mobiles*, thèse de doctorat, École polytechnique, 2001.

[8] J. J. Leifer, *Operational congruences for reactive systems*, thèse de doctorat, Computer Laboratory, University of Cambridge, 2001, Available in revised form as Technical Report 521, Computer Laboratory, University of Cambridge, 2001.

### Articles et chapitres de livre

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

[10] J. J. Leifer, R. Milner, « Shallow linear action graphs and their embeddings », *Formal Aspects of Computing*, 2002, à paraître.

## Communications à des congrès, colloques, etc.

[11] M. Abadi, B. Blanchet, « Secrecy Types for Asymmetric Communication », *in : Foundations of Software Science and Computation Structur es (FoSSaCS 2001)*, F. Honsell, M. Miculan (éditeurs), *LNCS*, *2030*, Springer, p. 25–41, Genova, Italy, avril 2001.

[12] M. Abadi, B. Blanchet, « Analyzing Security Protocols with Secrecy Types and Logic Programs », *in : 29th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'2002)*, ACM Press, Portland, Oregon, janvier 2002. à paraître.

[13] B. Blanchet, « Abstracting Cryptographic Protocols by Prolog Rules (inv ited talk) », *in : 8th International Static Analysis Symposium (SAS'2001)*, P. Cousot (éditeur), *LNCS*, *2126*, Springer, p. 433–436, Paris, France, juillet 2001.

[14] B. Blanchet, « An Efficient Cryptographic Protocol Verifier B ased on Prolog Rules », *in : 14th IEEE Computer Security Foundations Workshop (CSFW-1 4)*, IEEE Computer Society, p. 82–96, Cape Breton, Nova Scotia, Canada, juin 2001.

[15] S. Conchon, F. Pottier, « JOIN(X) : Constraint-Based Type Inference for the Join-Calculus », *in : Proceedings of the 10th European Symposium on Programming (ESOP'01)*, D. Sands (éditeur), *Lecture Notes in Computer Science*, *2028*, Springer Verlag, p. 221–236, avril 2001, `http://pauillac.inria.fr/~fpottier/publis/conchon-fpottier-esop01.ps.gz`.

[16] F. Le Fessant, L. Maranget, « Optimizing Pattern Matching », *in : 6th ACM SIGPLAN International Conference on Functional Programming(ICFP'01)*, ACM Press, septembre 2001.

## Rapports de recherche et publications internes

[17] R. Cori, J.-J. Lévy, *Algorithmes et Programmation*, Ecole polytechnique, octobre 2000.

[18] G. Dowek, T. Hardin, C. Kirchner, « A Completeness theorem for an extension of first-order logic with bind ers », *rapport de recherche*, 2001.

## Divers

[19] Ma Qin, « Prototyping on Object-Oriented Join », Rapport de DEA, Université de Paris VI, octobre 2001.

[20] G. Peskine, « The JoCaml abstract machine », Rapport de DEA, Université de Paris VI, octobre 2001.

[21] A. Schmitt, J.-B. Stéfani, « The M-calculus, a higher-order distributed calculus », Soumission à ECOOP 2002, novembre 2001, `http://pauillac.inria.fr/~aschmitt/m-calculus.short.ps.gz`.

[22] A. Schmitt, J.-B. Stéfani, « The Marvel Programming Model : a higher-order distributed calculus », Rapport interne, novembre 2001, `http://pauillac.inria.fr/~aschmitt/m-calculus.long.draft.ps.gz`.

[23] R. K. Singh, « Peer to Peer File Sharing in JoCaml », Summer Training Repport, Indian Institute of Technology, octobre 2001, `http://pauillac.inria.fr/~kumarsin/project/report.html`.