Projet PARA

Parallélisme

Localisation : *Rocquencourt*

Mots-clés : parallélisme, parallélisme asynchrone, preuves de programmes, programmation fonctionnelle, architecture multiprocesseur, concurrence, λ-calcul, ramasse-miettes, π-calcul, programmation distribuée, résistance aux pannes, agents mobiles.

1 Composition de l'équipe

Responsable scientifique

Jean-Jacques Lévy, directeur de recherche

Responsable permanent

Georges Gonthier, chargé de recherche

Personnel INRIA

Damien Doligez, chargé de recherche
Robert Ehrlich, ingénieur de recherche
Luc Maranget, chargé de recherche

Conseillère scientifique

Thérèse Hardin, professeur à Paris 6

Secrétaire

Ghislaine Le Corre, jusqu'au 1-9-96
Sylvie Loubressac, en commun avec CRISTAL et COQ

Chercheurs doctorants

Florent Guillaume, ENS, jusqu'au 1-9-96
Cédric Fournet, ingénieur des Ponts et Chaussées
Bruno Pagano, boursier MENESR, Univ. Paris 7

Stagiaire

Sylvain Huet, DEA, 1-5-96 au 31-8-96

2 Présentation du projet

La programmation de plusieurs processus concurrents n'est pas facile, car leur synchronisation est souvent délicate. Elle demande de bien comprendre le modèle sous-jacent et de disposer de primitives rigoureusement définies. C'est pourquoi le projet Para étudie la programmation concurrente. Notre projet ne traite ni du parallélisme fortement synchrone, ni de l'algorithmique pour des machines SIMD à parallélisme massif. Notre objectif est de construire des systèmes pour programmer des applications faiblement synchrones sur des architectures à mémoire partagée ou distribuée. Dans ce cadre, notre effort se dirige actuellement vers la programmation des processus mobiles qui permet de tenir compte de la reconfiguration dynamique des interconnexions.

En effet, le développement des réseaux et des applications distribuées à grande échelle rend encore plus critiques la programmation. Les serveurs multi-sites doivent s'envoyer des informations de très haut niveau pour contourner les limitations de bande passante. On peut donc envisager l'échange non seulement de données, mais de petits programmes que les serveurs (ou même les clients modulo des 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. Certains langages tels que Facile (de B. Thomsen à l'ECRC-Munich), Obliq (de L. Cardelli à DEC/SRC) et Pict (de B. Pierce et D. Turner à Cambridge/Edimbourg) ont été récemment définis, reprenant plus systématiquement la vieille idée du langage Plasma (de C. Hewitt au MIT en 1970). Plus commercialement, il y a le 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 permettent l'envoi de sous-programmes actifs, c'est-à-dire de processus avec leurs environnements actifs en cours d'exécution. Notre groupe travaille sur la définition d'un Pict distribué. Un nouveau calcul, le *join-calcul*, a été introduit en 95 et sa relation avec le π-calcul a été démontrée. Ce nouveau calcul est implémentable, même en présence de pannes, contrairement au π-calcul de R. Milner qui suppose une diffusion résistante aux pannes pour chaque communication. Le prototype d'une implémentation physiquement distribuée est en cours d'achèvement.

L'intérêt de notre projet est donc non seulement porté vers la sémantique des langages de programmation, mais aussi vers leur implémentation parallèle et à plus long terme la construction de systèmes informatiques concurrents. Pour ce faire, 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. Par exemple, les langages dont nous étudions les relations avec le parallélisme ou la distribution sont des langages fonctionnels, pour trois raisons : 1) ils sont rigoureusement définis, 2) ils sont facilement disponibles dans le contexte de l'INRIA, 3) ils correspondent à la culture technique des personnes de notre projet. Pour mener une activité pratique, nous utilisons les machines mono-processeur standards de l'INRIA, mais aussi des machines multi-processeurs (un Sequent B8000 à dix processeurs et à mémoire partagée, un Encore MultiMax avec 14 processeurs, et une KSR à 72 processeurs). Un glaneur de cellules concurrent, non bloquant et portable a été conçu et implémenté pour *Concurrent Caml-light* (CCL) ; sa version incrémentale mono-processeur est celle disponible sur Caml-light 0.7 et sur O'Caml. Il a été certifié mécaniquement en 95 grâce à une preuve de 9 hommes mois.

Quant à la théorie, c'est aussi une activité importante pour notre groupe. Elle se concentre sur les propriétés syntaxiques des systèmes avec variables liées et sur quelques problèmes fins de l'implémentation des langages fonctionnels ou concurrents : sémantique du *join-calculus* et relation avec le π-calcul, formalisation des *back ends* avec les substitutions explicites, filtrage, évaluations incrémentales, systèmes de réductions abstraits pour traiter des théorèmes des développements finis et de normalisation forte, ou des réécritures de graphes. En 1996, une bonne partie de l'effort théorique a été de publier la sémantique du langage Pict distribué avec adressage physique intégré. Ce travail a été effectué par un groupe de travail regroupant la grande majorité des membres de notre projet et D. Rémy du projet CRISTAL. Dans ce cadre, nous avons recréé la communauté de notre projet Esprit CONFER en lançant en novembre 96 le Working Group CONFER-2 du Framework 4, dont nous sommes le site coordinateur.

L'année 1996 est pour nous une année d'approfondissement et de recentrage autour de notre projet principal de construction d'une implémentation prototype du *join-calcul* distribué. Elle est aussi marquée par la soutenance de thèse de P.-A. Melliès, qui est parti pour 2 ans en post-oct chez S. Abramsky à l'université d'Edimbourg en février 1996. S. Huet a été stagiaire du DEA SPP au printemps. F. Guillaume a rejoint le corps des Telecom. G. Le Corre est partie en mobilité à l'INRIA Rhone-Alpes.

3 Actions de recherche

3.1 Un modèle du calcul distribué d'ordre supérieur

Participants : D. Doligez, C. Fournet, G. Gonthier, J.-J. Lévy , L. Maranget, D. Rémy (projet CRISTAL)

Mots-clés : parallélisme, parallélisme asynchrone, pi-calcul, langage fonctionnel, programmation répartie, tolérance aux fautes, agents mobiles, concurrence, ramasse-miettes.

Nous avons poursuivi en 1996 le développement de notre modèle formel, le *join-calcul*, afin d'y intégrer les primitives de distribution qui avaient été identifiées par notre groupe de travail en 1995. Nous avons ainsi défini le *join-calcul distribué*, qui est un des premiers calculs de processus prenant réellement en compte les problèmes liés à la répartition des calculs.

Dans le join-calcul distribué, chaque sous-processus est lié lexicalement à une *location* qui détermine l'endroit où il s'exécute. Le join-calcul distribué comporte de plus des primitives pour créer, déplacer, arrêter, et tester l'échec de locations.

Techniquement, le join-calcul distribué se distingue par trois innovations :

l'organisation hiérarchique de locations : une location peut contenir d'autres locations, et ainsi de suite récursivement. On peut ainsi associer une location à chaque niveau de mobilité : une location peut aussi bien représenter un site qu'un agent, une ressource ou un objet, et l'imbrication des locations peut aussi bien représenter la situation géographique d'un agent que l'attachement de ses ressources à un agent. Enfin, la primitive de déplacement permet de modifier dynamiquement l'arbre des locations, par exemple pour faire migrer un agent ou lui rajouter de nouvelles ressources.

le controle lexical de la mobilité : les primitives de migration et d'arrêt n'affectent que leur location lexicale. Ceci permet d'avoir un contrôle syntaxique simple de la mobilité, par exemple on peut aisément vérifier qu'une location est un site physique, c'est-à-dire qu'elle ne peut se déplacer. Cette propriété lexicale permet plus généralement d'offrir des garanties de sûreté de fonctionnement; par exemple, un serveur peut autoriser une *applet* à s'installer, sans lui permettre de déplacer ou fermer le serveur.

une primitive de détection d'échec : la primitive *fail* permet de détecter si une location s'est arrêtée, y compris dans le cas où cet arrêt est dû à une panne du site où elle se trouve. En effet, il nous est apparu qu'il était impossible de programmer la récupération d'erreur sans détection fiable de l'échec, notamment lorsqu'on délègue des pouvoirs à un agent. Or la détection de panne est théoriquement impossible dans un modèle asynchrone comme le join-calcul, mais souvent possible en pratique, en utilisant par exemple des propriétes de temporisation, ou une hiérarchie administrative, ou toute autre méthode pour atteindre un consensus distribué. La primitive *fail* permet d'encapsuler ces mécanismes particuliers et de conserver la simplicité du modèle asynchrone dans le reste du modèle.

Enfin, soulignons la symétrie entre la définition de processus du join-calcul, qui réalise une liaison statique avec une portée dynamique, et la définition de locations, dont la portée est statique, mais dont les liaisons d'imbrication sont dynamiques.

Ce cadre théorique à la fois simple et rigoureux a permis une étude précise de la sémantique de l'échec. Ceci a aboutit à la définition d'une primitive de test d'échec compatible avec le calcul asynchrone, et à une formalisation des équivalences modulo échec.

Plusieurs exemples de protocoles classiques ont été modélisés dans le join-calcul distribué, afin d'en valider la puissance d'expression. Nous avons en outre étudié un protocole client-agent-serveur (*CASA*) afin de démontrer l'adéquation de notre modéle aux systèmes répartis et *mobiles*.

Tous ces résultats ont fait l'objet d'une communication à CONCUR'96 [596].

3.2 Concurrence et typage

Participants : C. Fournet, L. Maranget, D. Rémy

Mots-clés : concurrence, typage, agents mobiles, parallélisme, pi-calcul, programmation répartie.

Le *join-calcul* de base n'est pas typé. Comme pour le π-calcul de Milner, on peut ajouter un système de *sortes* pour au moins savoir si les valeurs circulant sur les canaux de communication sont de simples scalaires ou des n-uplets. Mais il est plus difficile d'ajouter un système de type polymorphe, car le coté très symétrique du π-*calcul* interdit la généralisation des types. Dans le *join-calcul*, comme un canal ne peut recevoir des données que vers les définitions des canaux de communication, l'endroit où l'on peut généraliser les types est naturel. Alors, un simple critère syntaxique permet de distinguer entre vrai polymorphisme ou polymorphisme *faible* (comme pour les références de ML).

Dans le *join-calcul*, le typage a une double fonction. Non seulement, il permet de déterminer le type des valeurs passant sur les canaux. Mais il autorise également de déterminer si un canal est synchrone ou asynchrone. Dans le *join-calcul* de base, cette distinction se fait en notant les canaux synchrones par un nom commençant par une majuscule. Avec le typage, cette convention quelque peu fastidieuse disparait. Or la dualité synchrone / asynchrone est fondamentale pour obtenir un système d'exécution efficace, et pour écrire les parties synchrones comme des expressions d'un langage de programmation normal. Avec le typage, nous arrivons donc à aller vers l'intégration du *join-calcul* dans un langage de prgrammation réaliste comme C ou ML.

Ce travail [602] est soumis à la conférence CONCUR'97.

3.3 Concurrence et sécurité

Participants : C. Fournet, G. Gonthier, S. Huet

Mots-clés : concurrence, securite, agents mobiles, cryptographie, parallélisme, pi-calcul, programmation répartie.

Pour pouvoir répondre aux questions de sécurité soulevées par l'implantation d'applications à code mobile, nous avons entamé des recherches pour intégrer le contrôle de la sécurité à notre modèle de calcul distribué. Cette recherche s'est déroulée en deux temps, et sur deux axes différents.

De janvier à juin, S. Huet a accompli au sein du projet PARA un stage de DEA sur le sujet "Sécurité et processus mobiles" [603]. Il s'agissait de comprendre comment on pouvait programmer des mécanismes de sécurité en join-calcul, et éventuellement quelles primitives il convenait d'ajouter au calcul distribué pour faciliter l'écriture d'applications sécurisées.

La portée lexicale du calcul correspond naturellement à un mécanisme de capacités, et il est rapidement apparu que le calcul permettait de coder d'autres mécanismes tels que les listes d'accès, et ce même sans primitive supplémentaire pour tester l'égalité de deux noms.

Par contre, de tels mécanismes de contrôle microscopique des droits et capacités se sont révélés lourds et inadaptés aux modifications globales de contexte générées par la migration : un même processus mobile peut agir sur des systèmes différents, et souvent pour le compte de mandataires différents, ne serait-ce que parce que le lieu d'où un message est émis est souvent tenu responsable de son contenu.

Aussi, S. Huet a proposé de compléter le modèle en ajoutant un système d'*identités* aux locations, et de compléter la migration physique par un système de *migration de responsabilité* permettant à un agent mobile de gagner ou perdre des identités.

Un deuxième axe de recherche a été entamé lors de la visite de M. Abadi (Digital) en septembre 1996. Il s'agissait de déterminer comment la sureté de la portée lexicale du join-calcul pouvait être implémentée avec des moyens cryptographiques élémentaires. Nous avons montré qu'on pouvait avoir une telle implémentation générique en associant à chaque nom une clé publique, ce qui correspond aux pratiques usuelles dans ce domaine. Le même cadre formel permet également de démontrer la correction d'optimisations courantes, comme générer et utiliser des clés privées par session. Il faut noter que le résultat correspondant pour le π-calcul de Milner, qui associerait une clé privée à chaque canal, est faux, à cause de problèmes de confidentialité future (*forward secrecy*).

3.4 Un prototype pour le *join-calcul*

Participants : C. Fournet, L. Maranget

Mots-clés : parallélisme, parallélisme asynchrone, pi-calcul, langage fonctionnel, programmation répartie, tolérance aux fautes, agents mobiles, concurrence, ramasse-miettes.

C. Fournet et L. Maranget ont commencé au printemps 96 la réalisation d'un prototype pour un langage fondé sur le join-calcul distribué. Ce langage comporte des noms synchrones et asynchrones, et les primitives pour la distribution (création et mobibilité des agents). Le compilateur est muni d'un typeur à la ML, et génère du *bytecode* qui peut ensuite s'exécuter sur plusieurs machines reliées par le réseau local. Une version 1.0 de cet important travail de programmation sera achevée en janvier 97. Le système est codé en O'Caml et en C.

3.5 Preuves mécaniques

Participants : Damien Doligez, Georges Gonthier

Mots-clés : parallélisme asynchrone, concurrence, ramasse-miettes, preuves formelles.

D. Doligez a entrepris en 96 la construction d'un prototype de démonstrateur de preuves du 1er ordre pour arriver à refaire plus simplement la preuve du GC concurrent achevée en 95. Le démonstrateur en cours de construction est déclaratif, ce qui augmente la lisibilité des preuves. Il n'est pas interactif, et est donc parallélisable. Le système est construit en deux parties: le prouveur, contenant les différentes tactiques, et le vérificateur, très petit et donc dont on peut vérifier la bonne correction. L'objectif a moyen terme est de refaire la preuve menée en 95 dans le système TLP [598], et de comparer sa facilité d'écriture par rapport à l'ancienne preuve.

3.6 Langages de programmation fonctionnels

Participant : Damien Doligez

Mots-clés : programmation fonctionnelle.

D. Doligez est le "mainteneur en titre" des versions Macintosh de Caml. Il est aussi expert dans la mise au point du glaneur de cellules incrémental pour le système Caml Special light, qui est aussi celui de O'Caml. En collaboration avec X. Leroy (du projet CRISTAL), D. Doligez a porté sur Macintosh le système Objective Caml. Pour l'instant, le système n'utilise pas toutes les caractéristiques modernes des système 7 de Mac-OS. Une nouvelle version est en cours pour en bénéficier.

3.7 Substitutions explicites

Participants : Thérèse Hardin, Luc Maranget, Jean-Jacques Lévy, Bruno Pagano

Mots-clés : lambda-calcul, programmation fonctionnelle.

T. Hardin, en collaboration avec G. Dowek (projet COQ), C. Kirchner (projet PROTHEO) et F. Pfenning (CMU), a prolongé le travail sur l'unification à l'ordre supérieur en utilisant le formalisme des substitutions explicites ($\lambda\sigma$-calcul). Elle a étudié le cas particulier des patterns et montré que leur unification est décidable, à l'aide d'un algorithme spécifique d'unification dans le $\lambda\sigma$-calcul. L'algorithme obtenu est plus simple que l'algorithme correspondant dans le λ-calcul et l'étude a permis de mieux comprendre les raisons de cette décidabilité. Ce travail a été présenté à la conférence IJCWLP'96 (Int. Joint Conf. and Workshop on Logic Programming).

B. Pagano, T. Hardin et L. Maranget ont complété la description des machines de compilation à environnement dans le cadre unifié du $\lambda\sigma$-calcul, avec réduction faible. Ce travail a été présenté à ICFP'96.

3.8 Maintenance des systèmes d'exploitation

Participants : Damien Doligez, Robert Ehrlich

Mots-clés : systèmes d'exploitation.

R. Ehrlich continue une activité multiforme de soutien système. Il a récupéré les données de disques endommagés ou maintenu divers systèmes de stockage : par exemple, il a récupéré les disques du projet RODIN, il a réparré la connexion SCSI avec la machine stockage d'images du service audio-visuel, il a modifié avec D. Doligez le programme Retrospect de sauvegarde des Macintosh pour permettre le backup sur DAT Gigatape.

Il a mené quelques actions plus ponctuelles comme la mise au point d'une carte TAK'ASIC pour réseau Ethernet sans fil sur un PC pour P. Muhlethaler du projet ALGO. Il a aussi fait un programme de lecture de bandes backup VMS qui sont le seul moyen de communication avec l'aérospatiale (cf. actions industrielles).

Enfin et plus généralement, R. Ehrlich continue de contribuer au bon fonctionnement des ordinateurs de notre bâtiment et à la très grande disponibilité de la messagerie électronique.

D. Doligez participe à l'entreprise de sécurisation des moyens informatiques. Avec X. Leroy, P. Weis du projet CRISTAL et M. Loyer, il fait fonctionner le protocole S/Key sur les machines de notre groupe de projets. Cette expérience sera par la suite généralisée à l'ensemble du site de Rocquencourt.

4 Actions industrielles

4.1 Aérospatiale et Ariane 502

Participants : Damien Doligez, Georges Gonthier, Jean-Jacques Lévy, Robert Ehrlich, Alain Deutsch, François Rouaix, Marcin Skubiszewski

Mots-clés : programmation fonctionnelle.

En novembre 96, nous avons commencé une action transversale avec les projets A3, CRISTAL et RODIN pour expertiser le logiciel du lanceur Ariane 5 avec le centre de l'Aérospatiale des Mureaux. A la suite des recommandations de la commission d'enquête du vol 501, nous sommes chargés d'effectuer une approche *bottom-up* sur les logiciels embarqués. Sur les codes Ada et Assembleur, nous pensons faire fonctionner le programme de détection des débordements de tableau, des variables non initialisées de A. Deutsch, en détournant le compilateur Ada de Gnu pour le transformer en frontal des analyseurs déjà développés pour C ou C++. Par ailleurs, nous voulons mener une modélisation en Promela des automates de synchronisation et lancer des détections de blocages ou de non atteignabilité avec le

programme Spin de G. Holtzman. Enfin, nous espérons pouvoir donner des avis sur le style de la programmation et la définition des interfaces.

Ce travail est assez conséquent, puisque la masse des programmes embarqués est imposante, et que les contraintes de temps pour l'étude sont réduits, puisque cela doit se faire avant les commissions de qualification du vol 502.

4.2 Ramasse miettes pour Java

Participants : Damien Doligez, Jean-Jacques Lévy

Nous avons mené une approche auprès d'OSF pour expliquer que le GC concurrent, portable et non bloquant développé par D. Doligez et G. Gonthier en 94 et 95 pouvait être avantageusement intégré dans les interpréteurs concurrents de Java, tels que l'OSF Grenoble veut développer. Après plusieurs discussions et visites, cette action ne s'est toujours pas conclue favorablement, mais les contacts sont toujours en cours.

5 Actions nationales et internationales

5.1 Projet Esprit CONFER

Le projet Esprit CONFER, dont nous étions coordinateur, *Basic Research* (BRA) 6454 s'est terminé en novembre 95. Après quelques difficultés administratives dues principalement au déménagement de l'équipe de B. Thomsen de l'ECRC (Munich) à ICL (Londres), nous avons redémarré le 1 novembre 96 le groupe de travail CONFER-2, (*working group 21836*), avec la même communauté, ie. Cambridge (Milner), CWI (Klop), Edimbourg (Abamsky), ENS (Curien), KTH (Parrow), Inria Sophia (Boudol), Pise (Montanari), augmentée des universités de Bologne (Asperti), Sussex (Hennessy), et Warwick (Walker) et du Cnet Lannion (Monin). Ce groupe, dont nous sommes également coordinateurs, constitue au niveau européen le principal groupe de réflexion sur les relations entre la fonctionnalité et la sémantique des processus mobiles.

5.2 Actions nationales

Séminaire Coq-Cristal-Para Notre groupe a un séminaire commun avec les projets COQ et CRISTAL de Rocquencourt.

Séminaire d'informatique de l'X J.-J. Lévy participe à l'organisation du séminaire mensuel d'informatique de l'École Polytechnique qui a reçu en 1996 M&Mme. Haton, Sillion, Abadi, Hullot, Kenyon.

Comité TSI T. Hardin est depuis janvier 1994 membre du comité de rédaction de TSI, et a été responsable d'un numéro spécial sur la programmation fonctionnelle.

Administration INRIA J.-J. Lévy est président du comité des projets de Rocquencourt, membre de la commission d'évaluation, membre du jury de recrutement de l'UR de Rocquencourt, 20-21 mai et 12 juin. Il quitte cette fonction au 31 décembre 96.

6 Diffusion des résultats

6.1 Diffusion de produits

Le système Caml est distribué par le projet CRISTAL en *ftp* anonyme. Son glaneur de cellules a été écrit par D. Doligez.

6.2 Dépot de brevet

Le travail sur les calculs de dépendances [594] a fait l'objet d'un dépot de brevet américain par l'entreprise *Digital Equipment Corporation*. Les signataires du brevet sont M. Abadi, B. Lampson, R. Levin, J.-J. Lévy, et Yuan Yu.

6.3 Actions d'enseignement

6.3.1 Enseignement universitaire

Ecole Polytechnique J.-J. Lévy est professeur à l'École Polytechnique. Il a fait le cours "Systèmes et Réseaux" de majeure 2ème année [601] au printemps 96, en étant assisté de D. Doligez, R. Ehrlich, F. Rouaix (projet CRISTAL) et F. Ruget (Chorus systèmes). Il est par ailleurs responsable du cours "Algorithmes et Programmation" de tronc commun de première année (30 enseignants, 440 élèves, rédaction du polycopié disponible sur le Web [600], cours en amphi et petites classes, projets informatique), qu'il organise toutes les années paires en alternance avec R. Cori. Il l'a donc effectué à l'automne 96.

G. Gonthier est maître de conférence à l'École Polytechnique. Il est responsable des contrôles de classement et des projets informatiques.

D. Doligez et R. Ehrlich ont effectué des vacations pour les travaux pratiques du cours "Systèmes et Réseaux" au printemps 96.

Université Paris 6 T. Hardin est professeur à l'université Paris 6. Elle y assure le cours de programmation et structures de données dans la maîtrise de sciences et techniques d'informatique et y encadre plusieurs stages en entreprise. Elle est également chargée du cours de licence d'informatique sur les fondements de la programmation.

Elle anime à l'Université Paris 6, 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.

DEA SPP T. Hardin effectue le cours "Aspects syntaxiques pour le calcul" du tronc commun du DEA Sémantique, Preuves et Programmation, affilié à Paris 6, l'ENS et Orsay. J.-J. Lévy et G. Gonthier participent à la filière concurrence asynchrone du même DEA. T. Hardin et J.-J. Lévy sont membres du comité de direction de ce DEA.

ENSTA L. Maranget a assuré une série de séances de travaux dirigés à l'École nationale supérieure des techniques avancées (ENSTA), à raison d'une séance hebdomadaire de deux heures pendant les mois de janvier et février. Le cours portait sur la programmation fonctionnelle en SML et s'adressait aux élèves de deuxième année.

6.3.2 Autres actions d'enseignement

Cours INRIA R. Ehrlich a donné pour la cinquième fois un cours sur le noyau Unix dans le cadre des cours de formation interne à l'INRIA.

6.3.3 Jurys de thèse

Nous avons été membres de plusieurs jurys de thèse: C. Lavatelli (Hardin, Lévy), P. Ferreira (Lévy), P.-A. Melliès (Gonthier, Lévy).

6.4 Participation aux manifestations

Conférence POPL G .Gonthier et C. Fournet ont presenté le join-calcul et la machine chimique réflexive à la conférence POPL'96 à St. Petersburg Beach, Floride, USA en janvier[597].

Conférence ICFP L. Maranget et B. Pagano ont présenté leur articles sur *backend* des compilateurs fonctionnels [599] à la conférence ICFP'96, Philadelphie, en juin. La présentation de l'article sur le calcul des dépendances [594] de J.-J. Lévy a été effectuée par M. Abadi (Digital).

Conférence CAV G. Gonthier a présenté la preuve de récupérateur de mémoire concurrent réalisée en 1995 à la conférence CAV'96 à New Brunsick, New Jersey, début août[598].

Conférence CONCUR C. Fournet, J.-J. Lévy, L. Maranget ont présenté leur article sur le calcul des agents mobiles [596] à CONCUR'96, qui se tenait à Pise, du 26 au 29 août.

Conférence WWW C. Fournet, J.-J. Lévy ont participé à la conférence Web, à Paris, du 6 au 10 mai.

Colloque FMOODS C. Fournet a participé au colloque sur les Méthodes formelles pour les systèmes répartis ouverts (FMOODS) à l'ENST à Paris, du 4 au 6 mars

Journées du CIRM C. Fournet a participé aux journées sur La logique et les modèles de calcul au CIRM à Marseille, du 16 au 20 septembre.

Workshop on Type Theory and Term Rewriting T. Hardin et J.-J. Lévy ont participé à cette réunion à Glasgow, du 9 au 14 septembre, et ont fait deux présentations sur les Métavariables et la substitution explicite et sur le calcul des dépendances pour les *Makefiles* de Vesta [594].

6.5 Activités extérieures

Visite Bell laboratories C. Fournet a présenté son travail sur le join-calcul à Lucent Technologies, Bell laboratories, Murray Hill en janvier, dans le séminaire du groupe de D. Macqueen.

Visite INRIA Sophia C. Fournet a présenté son travail au séminaire du groupe Meije à l'INRIA Sophia en mars.

Séminaire INRIA Sophia G. Gonthier a présenté la preuve de récupérateur de mémoire concurrent réalisée en 1995 à l'équipe CROAP (Inria Sophia-Antipolis) fin mai.

7 Publications

Thèses

[592] P.-A. MELLIÈS, *Description Abstraite des Systèmes de Réécritures*, Thèse de doctorat, Université Paris 7, décembre 1996.

Articles et chapitres de livre

[593] 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.

Communications à des congrès, colloques, etc.

[594] M. ABADI, B. LAMPSON, J.-J. LÉVY, «Analysis and Caching of Dependencies», *in : ACM Sigplan International Conference on Functional Programming, Philadelphia*, 1996.

[595] G. DOWEK, T. HARDIN, C. KIRCHNER, F. PFENNING, «Unification via Explicit Substitutions:The Case of Higher-Order Patterns», *in : International Joint Conference and Symposium on Logic Programming, Bonn, sept. 96*, 1996.

[596] C. FOURNET, G. GONTHIER, J.-J. LÉVY, L. MARANGET, D. RÉMY, «A Calculus of Mobile Agents», *in : 7th International Conference on Concurrency Theory (CONCUR'96)*, Springer-Verlag, p. 406–421, Pisa, Italy, août 26-29 1996. LNCS 1119.

[597] C. FOURNET, G. GONTHIER, «The Reflexive Chemical Abstract Machine and the Join-Calculus», *in : Proceedings of the 23rd ACM Symposium on Principles of Programming Languages*, ACM, p. 372–385, St. Petersburg Beach, Florida, janvier 21-24 1996.

[598] G. GONTHIER, «Verifying the safety of a practical concurrent garbage collector», *in : Proceedings of the Eighth International Conference on Computer-Aided Verification*, ACM, New Brunswick, New Jersey, juillet 1996.

[599] T. HARDIN, L. MARANGET, B. PAGANO, «Functional Back-Ends within the Lambda-Sigma Calculus», *in : ACM Sigplan International Conference on Functional Programming, Philadelphia*, 1996.

Cours polycopiés

[600] R. CORI, J.-J. LÉVY, *Algorithmes et Programmation*, Ecole polytechnique, octobre 1996, `http://www.polytechnique.fr/poly/~www/poly/`.

[601] J.-J. LÉVY, *Systèmes et Réseaux*, Ecole polytechnique, mars 1996, `http://www.polytechnique.fr/poly/~www/poly/majifa-syst-96/`.

Rapports de recherche et publications internes

[602] C. FOURNET, C. LANEVE, L. MARANGET, D. RÉMY, «Implicit typing à la ML for the join-calculus», *rapport de recherche*, INRIA, Bologna, Rocquencourt, 1996, soumis à publication.

[603] S. HUET, «Sécurité et processus mobiles», *rapport de recherche*, DEA, Ecole polytechnique, Palaiseau, 1996.

8 Abstract

The PARA project traditionally studies interactions between concurrency and functional programming. Since 1995, our focus is on languages for mobile applications. We defined a higher-order distributed language with mobility and error detection primitives based on the *join-calculus*. In 1996, we mainly developed its implementation, which will be released in January 1997. Theoretical results have also been expressed on explicit substitutions, incremental computations, and abstract reduction systems.