Rapport INRIA 1994 — Programme 2

Interprétation, Compilation et Sémantique des Langages Applicatifs

PROJET ICSLA

3 mai 1995

PROJET ICSLA

Interprétation, Compilation et Sémantique des Langages Applicatifs

Localisation : *Rocquencourt, École Polytechnique*

Mots-clés : allocation de registre (1), analyse de programme (1), analyse statique (1), approximation (1), compilation (1), génération de code (1), gestion de mémoire (1), interprétation abstraite (1), interprétation de langage fonctionnel (1), langage à objets (1), langage fonctionnel (1), langage intermédiaire (1), Lisp (1), Lisp parallèle (1), Lisp réparti (1), optimisation de code (1), parallélisme (1), pipe-line (1), ramasse-miettes (1), ramasse-miettes réparti (1), réseau informatique (1), Scheme (1), sémantique (1), vérification de programme (1).

1 Composition de l'équipe

Responsable scientifique

Christian Queinnec, maître de conférence, X

Responsable permanent

Bernard Serpette, chargé de recherche, INRIA

Secrétariat

Josy Baron, INRIA (en commun avec Chloé)

Personnel INRIA

Alain Deutsch, chargé de recherche, INRIA

Chercheur extérieur

Francis Dupont, DGA

Chercheurs doctorants

Jean-Marie Geffroy, boursier MESR
Manuel Serrano, boursier MESR

Stagiaire

Thierry Saura, École Supérieure d'Informatique, d'Électronique et d'Automatique du 1er juin au 30 septembre

2 Présentation du projet

Le projet ICSLA est né à l'automne 1989. Son but est l'étude des langages applicatifs (et principalement des dialectes dérivés de Lisp ou Scheme) sous trois formes principales :

implantation : y sont principalement étudiés des techniques de glanage de cellules (conservatif, parallèle, distribué et compilé), de nouveaux langages intermédiaires communs aux principaux langages applicatifs et la compilation vers le langage C.

sémantique : afin de rendre les compilateurs plus intelligents, diverses analyses sont effectuées comme la durée de vie et le partage des objets, la réduction statique et la forme des continuations, la taille des données intermédiaires et la compilation du filtrage. Y sont aussi étudiés de nouveaux traits linguistiques comme les continuations partielles, les modules, les objets ou les macros.

parallélisme : ces études portent sur les techniques de distribution de tâches ou de données, sur les protocoles de gestion de cohérence de variables distribuées partagées ainsi que sur les traits linguistiques procurant ces possibilités.

3 Actions de recherche

3.1 Langages intermédiaires

Participants : Bernard Serpette, Pierre Weis

L'activité sur les langages intermédiaires s'est portée sur les points suivants :

Spécification de langages intermédiaires : Deux langages nommés, respectivement **H** et **L**, ont été spécifiés. Le premier peut être vu comme un langage *à la* **C** (variables locales à portée statique et à durée de vie dynamique, i.e. pas de fermetures) où l'allocation de la mémoire est rendue explicite par des constructeurs attachés aux définitions de type. Le second est un langage d'assemblage où le protocole d'appel reste non spécifié et où l'allocation des registres n'est pas encore effectuée. La traduction, consistant à transformer les expressions de **H** en instructions de **L** et à rendre explicite l'organisation de la mémoire, reste commune à tous les compilateurs.

Entiers marqués : Dans un langage typé dynamiquement les entiers ne peuvent coïncider avec les entiers de la machine. Les entiers sont donc marqués. Certaines techniques de marquages (marque dans les bits de poids faible de l'entier) n'apportent qu'un faible surcoût. Ce surcoût peut être encore diminué par le compilateur. Après analyse, ce problème d'optimisation se trouve être équivalent au problème de minimisation de polynômes à valeurs dans $\{0, 1\}$. Bien que NP-complet, ce problème semble se comporter convenablement dans la plupart des expériences menées.

Portage de la pico-machine : La pico-machine est le langage intermédiaire mis au point dans le projet Cristal par Pierre Weis. Ce langage correspond au langage **L** après spécification du protocole d'appel et allocation des registres. Nous avons écrit la traduction de la pico-machine vers les assembleurs des machines Mips, Sparc, HP-Snake et DEC-Alpha.

GC n'utilisant pas de pile : Nous avions écrit, pour la pico-machine, avec Pierre Weis, un glaneur de cellules, en *Stop&Copy*. Cette version du GC, écrite en **C** et de manière récursive effectuait des débordements de pile sur un banc d'essai. Ce GC a été réécrit, toujours en **C**, mais de manière itérative (en conservant les continuations dans les structures de données). Les deux graphiques ci-dessus révèlent que pour le sparc comme pour le mips, la deuxième version est plus rapide que la première. L'énorme différence de pente (un rapport de 12) montre bien l'inefficacité des fenêtres de registres du sparc.

3.2 Interprétation abstraite et analyse interprocédurale d'alias existentiels pour langages à pointeurs

Participants : Alain Deutsch

A. Deutsch a continué ses travaux sur l'analyse d'alias par interprétation abstraite, pour langages impératifs avec pointeurs. Ces travaux sont basés sur les résultats théoriques qu'il a développés pendant sa thèse.

Les méthodes existantes pour l'analyse d'alias de structures de données inductives avec pointeurs sont toutes basées sur deux techniques d'approximation: d'une part le *k-limiting*, qui ne peut distinguer deux sous-objets situés à la profondeur k ou au delà; et d'autre part les approximations *store-based*, qui ne peuvent distinguer entre les différents éléments d'une même structure de donnée inductive. Ces deux techniques ont été proposées par N.D. Jones et S. Muchnick au début des années 80 et prévalent toujours aujourd'hui. Bien que des progrès notables aient été accomplis récemment en analyse d'alias dans ses aspects interprocéduraux (voir conférences POPL et PLDI de 90 à 94), très peu de progrès ont été effectués depuis les travaux de Jones et Muchnick en ce qui concerne la précision de l'analyse des structures inductives avec pointeurs. En conséquence, l'optimisation, la vérification et la parallélisation de programmes avec pointeurs sont restées difficiles.

A. Deutsch a developpé une nouvelle interprétation abstraite interprocédurale pour l'analyse de structures de données inductives avec pointeurs. Cette interprétation abstraite permet de découvrir une nouvelle classe de propriétés d'alias qui est hors de portée des méthodes déjà existantes.

3.3 Evaluation partielle et compilation

Participants : Jean-Marie Geffroy

J.M. Geffroy a entamé le travail de rédaction de sa thèse d'Université présentant, outre un nouveau schéma d'évaluateur partiel, une étude de son intégration dans une chaîne de compilation classique.

Evaluation partielle personnalisable L'*évaluation partielle*, ou *spécialisation de programmes*, est une technique visant à effectuer pendant la compilation autant de calculs que possible, en s'autorisant à engendrer des versions spécialisées de certaines fonctions, à chaque fois que

l'on rencontre un site d'appel où la fonction invoquée est connue et où certaines propriétés des arguments fournis sont statiquement connues.

Partant du travail réalisé précédemment sur la compilation du filtrage avec retour arrière intelligent par évaluation partielle, l'élaboration d'un évaluateur partiel original a été entreprise. Notre système (Jimix) se démarque des autres existants, tels que Similix, Mix, Schism ou Fuse, par les traits suivants:

- un contrôle fin fourni à l'utilisateur sur la spécialisation, par le biais d'un module de paramétrage de la propagation des informations statiques.
- une spécialisation élaborée due à la propagation des résultats des tests dans les deux branches d'une conditionnelle, ainsi qu'à la signature de notre spécialiseur qui, à l'instar de Fuse, ne se contente pas de renvoyer du code résiduel, mais également une *description* des valeurs que peut fournir l'évaluation de ce code résiduel.
- la présence dans le langage manipulé de la forme spéciale `set!`, absente des autres systèmes fort conservatifs dans le traitement des effets de bords: ceux-ci obligent à expliciter la mémoire, en définissant toutes les opérations sur celle-ci comme *dynamiques*, i.e. comme devant subsister dans le code résiduel. Jimix n'impose pas cette contrainte, et délimite correctement les effets de bords dangereux (i.e. qui doivent rendre la variable affectée dynamique).
- Le traitement de l'égalité physique.

Evaluation partielle et compilation Dès l'origine, Jimix n'a pas été conçu pour être auto-applicable, i.e. pour pouvoir se spécialiser lui-même par rapport à un interprète afin d'engendrer un compilateur grâce aux classiques projections de Futamura. Le but de Jimix est d'être inséré dans une chaîne de compilation. Les motivations de cette approche sont multiples:

- restreindre autant que possible le champ d'action du spécialiseur, et profiter de toutes les optimisations classiques que savent très bien faire les compilateurs Scheme actuels.
- déterminer lesquelles de ces optimisations peuvent se *synthétiser* par évaluation partielle: typage, analyse de flot de contrôle, ...

L'architecture personnalisable de notre spécialiseur nous permet de mesurer aisément l'impact de différentes stratégies de propagation d'in-

formation statique. Notre système est encore à l'état de prototype mais sera bientôt disponible pour expérimentation.

3.4 Bigloo: compilateur de langages fonctionnels vers C

Participants : Manuel Serrano, Pierre Weis

Manuel Serrano a réalisé un nouveau compilateur Caml qui a été obtenu par une approche originale: un simple pont entre deux compilateurs existants, chacun prenant en charge une partie du processus de compilation. Le premier compilateur est un compilateur Caml qui s'occupe de la partie amont et assure la compatibilité. Le deuxième compilateur est un compilateur Scheme optimisant, il s'occupe de la partie aval du nouveau compilateur et assure l'efficacité. Le premier compilateur Caml est celui de la version 0.6 du système Caml Light du projet Cristal, et le second est le compilateur Scheme Bigloo. Le nouveau compilateur est autogène (en anglais *bootstrapped*), complètement compatible avec le compilateur de la version 0.6 de Caml Light, et présente d'intéressantes optimisations pour les fonctions currifiées. Il produit du code dont l'efficacicité est comparable avec celle du code produit par les meilleurs compilateurs ML.

Notre compilateur se compare donc très favorablement.

Le compilateur Caml [6] a quitté le stade de l'expérimentation. Il a déjà compilé avec succès de gros programmes ML comme par exemple le système Coq. Il est disponible sur `ftp.inria.fr` et est déjà couramment utilisé de par le monde.

3.5 Interface entre langages

Participant : Manuel Serrano

Manuel Serrano a également poursuivi ses recherches dans le domaine de l'interface entre langages. Bigloo, capable de compiler Scheme et ML, sert de laboratoire d'expérimentation très précieux. Avec la version actuellement distribuée, il est possible de concevoir des applications mariant les trois langages: Scheme, ML et C.

Une partie du programme peut être écrite en Scheme, une partie en ML et enfin une partie en C. Ces trois composantes étant réunies lors de l'édition de liens pour former un unique exécutable. Il est ainsi possible

de tirer partie des possibilités de chaque langage en écrivant chaque module dans le langage le mieux adapté.

3.6 Continuations

Participants : Luc Moreau, Christian Queinnec

La notion de continuation partielle a fait l'objet d'une nouvelle définition où elles apparaissent par différences de continuations [3]. Cette définition mise au point par Luc Moreau, pendant le trimestre qu'il a passé à Rocquencourt en 1993, a permis d'affiner la terminologie relative à la notion de durée de vie des continuations et lui a permis de présenter de nouveaux exemples d'emploi de continuations partielles. Deux définitions équivalentes, par réécriture ou transformation de programme, caractérisent ces nouvelles continuations partielles.

3.7 Sémantiques de Lisp

Participant : Christian Queinnec

Christian Queinnec a passé une partie importante de l'année 1994 à finir d'écrire un livre fondé sur le cours qu'il donne au DEA ITCP de Paris 6 [1]. On y trouve successivement les techniques d'interprétation classiques (`eval` *expression environnement*) puis (`eval` *expression environnement continuation*) pour finir par (`eval` *expression environnement continuation mémoire*). La sémantique dénotationnelle est alors simplement obtenue à partir de ce dernier interprète. De celle-ci est ensuite dérivé un interprète rapide et efficace duquel est extrait un compilateur vers du code-octets. Ce compilateur permet alors d'appréhender et de discuter les problèmes d'évaluation dynamique et d'environnement de première classe mis en œuvre par `eval` ainsi que les aspects réflexifs de Lisp. Les macros et la compilation vers le langage C sont abordés dans les derniers chapitres ainsi que l'essence de l'implantation d'un système à objets. L'ouvrage utilise Scheme comme langage d'expression d'algorithmes mais, sur de nombreux points, discute des mérites respectifs de Bigloo, COMMON LISP, Dylan, EULISP, IS-Lisp, Le-Lisp, ML et Scheme. Quelques chapitres sont déjà enseignés à l'université d'Orléans (Jean-Jacques Lacrampe), de Nantes (Pierre Cointe), de Liège (Daniel Ribbens et Luc Moreau) et de Montréal (Marc Feeley).

Deux contributions originales de recherche apparaissent dans cet ouvrage : une proposition pour des environnements de première classe en Scheme assurant une bonne compilabilité ainsi qu'une proposition de sémantique de macros sous forme d'une tour d'évaluateurs imbriqués.

Pour la réalisation de cet ouvrage, a été développé un outil de fusion de code Scheme et TEX. Cet outil est nommé LiSP2TEX, est disponible depuis trois ans sur `ftp.inria.fr`, et a été récemment étendu après utilisation par Sophie Anglade et Jean-Jacques Lacrampe de l'université d'Orléans. Le principe de l'outil est de rechercher dans le texte en TEX des directives demandant l'inclusion de telle ou telle définition de fonction telle qu'elle apparaît dans le fichier où elle est définie. On peut ainsi séparer le code de sa documentation et fusionner les deux facilement.

3.8 Scheme distribué et parallèle

Participants : Christian Queinnec, Thierry Saura

L'écriture de programmes répartis sur un très grand nombre de sites de par le monde est un problème que l'extension récente d'Internet rend de plus en plus crucial. Un grand nombre de services présents sur les réseaux consistent en l'élaboration, l'enrichissement et la propagation d'informations comme par exemple `news`, `finger`, `netfind` ou `archie`. La conception et l'écriture de tels programmes sont notoirement délicates et sujettes à erreurs, ce que l'on peut aisément constater notamment dans les `news` où il n'est point rare de lire des réponses avant les questions auxquelles elles sont liées.

Nous avons étudié un nouveau langage, intitulé ICSLAS, et dont le but est d'être un langage d'extension, une sorte de glu permettant de mettre en œuvre de façon cohérente un ensemble de programmes partageant une même mémoire. Loin des excès des anciens Lisp qui exigeaient que tout soit écrit en Lisp même, notre dialecte ne fait que procurer un espace commun mais réparti à travers lequel dialoguent de multiples tâches. Cet espace commun est géré causalement, c'est-à-dire comme s'il était constitué d'une unique mémoire physique. Les techniques d'implantation supportent des communications intermittentes entre sites pourvu qu'aucun message ne soit perdu.

Deux résultats ont été obtenus : le premier concerne un protocole d'invalidation paresseux [4] permettant de gérer des données modifiables et réparties. Lorsqu'une telle donnée est modifiée sur le site qui la gère,

les informations invalidant les copies qui peuvent en exister ailleurs sont propagées paresseusement au gré des migrations calculatoires. Ce mécanisme assure que tout site voit la mémoire partagée suivant une coupe cohérente.

Le langage ICSLAS a d'autres propriétés comme une détection automatique de fin de calcul d'expression qui permet de procurer une détection automatique de terminaison distribuée. Cette caractéristique provient de la similarité entre terminaison distribuée et Glaneur de Cellules (GC) ainsi que de l'existence d'un GC distribué sûr. ICSLAS procure aussi la possibilité de manipuler explicitement l'ensemble des ressources informatiques associées à un calcul. Ce qui permet notamment de lancer plusieurs calculs en parallèle qui s'interrompront dès que l'un s'achèvera. L'intérêt d'ICSLAS est de pouvoir programmer explicitement le type de coopération souhaitée entre les tâches. Ce protocole de gestion de groupe de tâches est décrit en [5].

3.9 Sémantique de Scheme

Participants : Sophie Anglade, Jean-Jacques Lacrampe, Christian Queinnec

Ce travail [8] mené en collaboration avec Sophie Anglade et Jean-Jacques Lacrampe du LIFO (Université d'Orléans) a porté sur la spécification dénotationnelle de l'appel fonctionnel en Scheme qui a la propriété de ne pas préciser l'ordre d'évaluation de ses termes. Le travail a conduit à une spécification précise de cette imprécision et a consisté à caractériser tous les ordres possibles d'évaluation et rien que ceux-là.

3.10 Previsia

Francis Dupont a contribué à la mise en place et à l'administration de la plate-forme d'expérimentation du GIE PREVISIA pour:

- l'installation et l'exploitation de nouveaux systèmes d'exploitation (Solaris, Windows NT, ...)
- la configuration de réseaux non-standards (X.25, CLNS, ...)
- l'introduction des outils distribués DCE (Distributed Computing Environment).

4 Actions industrielles

C. Queinnec a été consultant auprès de la société Sagem durant l'année 94 en matière de machines pour l'intelligence artificielle et le temps réel.

Un accord de collaboration d'un an a été établi entre l'INRIA (A. Deutsch) et le centre de recherche IBM T.J Watson (Yorktown Heights, USA) sur l'analyse d'alias.

5 Actions nationales et internationales

Bernard Serpette organise, tous les deux mois environ et conjointement au club des porteurs de Le-Lisp, les réunions "(entre parenthèses)". Ces *(réunions)* tentent de faire le lien entre chercheurs et utilisateurs de langages basés sur Lisp.

Bernard Serpette a fait partie du comité de programme (et d'organisation) des Journées Francophones des Langages Applicatifs qui se sont déroulé à Noirmoutiers les 31 janvier et 1er février 1994.

A. Deutsch et C. Queinnec ont été membres du comité de programme de ICCL '94, *IEEE Computer Society 1994 International Conference on Computer Languages* qui s'est tenue à Toulouse en mai 1994.

C. Queinnec dirige le pôle "parallélisme et distribution" du GDR-PRC de Programmation.

Christian Queinnec anime le groupe Afnor de normalisation de Lisp ainsi que le groupe ISO sur ce même thème. Ce dernier groupe a notamment produit un premier projet de document normatif ISLisp (ISO-IEC/JTC1/SC22/WG16 CD13816). C. Queinnec participe aussi aux travaux du groupe EuLisp pour la définition d'un Lisp européen, les travaux du groupe y sont régulièrement exposés.

C. Queinnec est membre du comité de rédaction de TSI.

Le projet participe au réseau d'excellence "Parallel Virtual MultiComputer" (VIM) du programme HCM de la CEE. Ce réseau a été créé en 1993 et est animé par l'université de Bath.

Le projet a reçu une subvention dans le cadre de l'accord Tournesol d'échanges scientifiques avec la Belgique et, plus précisément dans notre cas, l'université de Liège.

Trois sites de l'INRIA situés à Rocquencourt, Grenoble et Sophia-Antipolis participent à l'expérimentation du service pilote ATM IRLE (*Asynchronous Transfer Mode* pour commutation de cellules, Interconnexion de Réseaux Locaux d'Entreprise). Ce service fournit une connexion niveau réseau à très haut débit. Francis Dupont a participé à la mise en place du réseau d'accès (FDDI à 100Mbits/s) du site de Rocquencourt, des services associés (IP multicast, etc...) et aux tests de performance. Pour exploiter au mieux ce réseau haut débit à longue distance, des systèmes d'exploitation de la famille Unix BSD 4.4 ont été installés sur des stations de travail.

Francis Dupont a porté les extensions TUBA sur ce type de systèmes et écrit un gestionnaire (*driver*) pour une carte FDDI PMD-TP d'origine Cisco (ex-Crescedo).

6 Diffusion des résultats

6.1 Diffusion de produits

Le projet diffuse quelques logiciels, principalement Bigloo, Meroon et LiSP2TEX, en `ftp` anonyme. Ces logiciels apparaissent aussi dans le *Scheme Repository*, vaste répertoire de documents relatifs au langage Scheme, qui se trouve sur la machine `cs.indiana.edu` ainsi que sur quelques autres serveurs (et notamment `ftp.inria.fr` sur lequel nous maintenons un miroir à l'intention des européens). Nous ne pouvons donc pas donner de statistiques planétaires mais seulement celles relatives au serveur `ftp` de l'INRIA : cette année, de janvier à septembre inclus, Bigloo a été pris environ 500 fois, Meroon 300 fois et LiSP2TEX 60 fois.

6.2 Actions d'enseignement

6.2.1 Enseignement universitaire

J-M Geffroy est moniteur à Marne la vallée, M. Serrano est moniteur à Paris 6.

ENSTA A. Deutsch a enseigné deux cours de 21h portant sur les langages fonctionnels et la compilation à l'École Nationale Supérieure des Techniques Avancées.

cours "sémantiques de Lisp" C. Queinnec a donné ce cours de 18 heures au DEA ITCP de l'université Paris 6, en janvier–mars 1994.

C. Queinnec a une demi-charge de maître de conférences à l'école polytechnique et y a dispensé les enseignements suivants :

travaux dirigés "ML" encadrement des travaux dirigés d'initiation algorithmique en CAML dans la majeure de première année sous la direction de J-M Steyaert.

cours "Compilation de Scheme" cours et travaux dirigés sur le langage Scheme et sa compilation vers le langage C.

6.2.2 Séminaires et formation permanente

A. Deutsch a présenté en 1994 ses travaux aux séminaires de :

- *Bull R&D*, à l'invitation de Jean-Éric Pin;
- *DIKU, université de Copenhague*, à l'invitation de S. Sagiv & Neil D. Jones;
- l'*École Normale Supérieure*, à l'invitation de Patrick Cousot;
- l'*École des Mines, Paris*, à l'invitation de François Irigoin;
- *Sun Microsystems, Mountain View, CA, USA*;
- *Rutgers University, NJ, USA* à l'invitation de Barbara Ryder;
- *Xerox PARC, Palo Alto, USA*, à l'invitation de Hans-J. Boehm;
- *IBM Watson, NY, USA* à l'invitation de Michael Burke;
- *Stanford University, USA* à l'invitation de Monica Lam;

C. Queinnec a présenté ses travaux [4] au LIFO (Orléans, janvier) et au LITP (Paris, janvier). M. Serrano a présenté Bigloo, les analyses statiques et les techniques de compilation au LITP (Paris, janvier) ainsi qu'à l'institut Pasteur.

M. Serrano et P. Weis du projet Cristal ont présenté la compilation optimisante de Bigloo pour Caml-Light en juin à Rocquencourt.

6.2.3 Jurys de thèse

C. Queinnec a fait partie des jurys de thèse de Luc Moreau (Liège, juin), David Plainfossé (Paris 6, juin), Manuel Serrano (Paris 6, décembre) et Éric Mattiachof (Paris 6, décembre).

6.3 Participation aux manifestations

JFLA 94 (Noirmoutiers, février)
F. Dupont, J-M. Geffroy, C. Queinnec, B. Serpette et M. Serrano ont assisté aux Journées Francophones des Langages Applicatifs.

Workshop ESPRIT BRA LOMAPS
A. Deutsch a présenté ses travaux à ce workshop ESPRIT qui s'est déroulé à l'École des Mines en mai 94.

ICCL 94 (Toulouse, mai)
C. Queinnec s'est rendu à la conférence ICCL 94 en tant que membre du comité de programme.

PLDI 94 (Orlando, USA, juin)
A. Deutsch y a présenté [2].

ML workshop 94 (Orlando, USA, juin)
M. Serrano et P. Weis du projet Cristal ont présenté leurs travaux de compilation autour de Caml-Light [6].

LFP 94 (Orlando, USA, juin)
C. Queinnec y a présenté [4].

INET/JENC 94 (Prague, juin)
F. Dupont a assisté à cette conférence.

SIGCOMM 94 (Londes, UK, août-septembre)
F. Dupont a assisté à cette conférence.

PLILP 94 (Madrid, Espagne, septembre)
M. Serrano y a effectué une démonstration de Bigloo [7] tandis que Luc Moreau y a présenté [3].

Journées GDR-PRC de Programmation (Lille, septembre)
C. Queinnec et M. Serrano ont participé aux journées du GDR-PRC de Programmation. M. Serrano y a présenté les travaux qu'il a effectué avec P. Weis (projet Cristal) [6] autour du compilateur optimisant pour Caml-Light.

Sécurité et réseaux
F. Dupont a présenté la sécurité d'IPng à la réunion GERET, au groupe de travail sécurité de l'AFUU ainsi qu'à NetWorld+InterOp 94 en septembre et octobre.

TPPP 94 (Sendai, Japon, novembre)
C. Queinnec y a présenté [5].

6.4 Activités extérieures

- Visite de l'université de Liège par C. Queinnec à l'invitation de D. Ribbens et L. Moreau dans le cadre de l'accord Tournesol.
- A. Deutsch a visité *DIKU, université de Copenhague* pendant une semaine en juin 94, à l'invitation de Neil D. Jones et S. Sagiv. Il y a donné un séminaire et a établi une collaboration avec S. Sagiv (IBM Haifa, DIKU, Copenhague & Wisconsin-Madison) qui est par ailleurs venu à deux reprises pour une semaine à Rocquencourt.
- A. Deutsch a visité *Carnegie Mellon University, Pittsburgh, USA* pendant un mois en juillet 94, à l'invitation de Peter Lee. Une collaboration a été entamée.
- Séjour à l'université de Montréal (Canada) du 20 juillet au 10 août de C. Queinnec à l'invitation de Marc Feeley, Guy Lapalme et Jacques Malenfant.
- Séjour à l'université de Pise (Italie) du 22 au 24 novembre de C. Queinnec à l'invitation de Giuseppe Attardi dans le cadre du réseau d'excellence VIM.

6.5 Réunions diverses

VIM Réunion plénière du réseau d'excellence VIM (pour *Virtual Multicomputer*) à Bonn (février).

Présentation du projet ICSLA dans le cadre de l'évaluation du LIX (Laboratoire d'Informatique de l'École Polytechnique) en mars.

RIPE F. Dupont, président du groupe de travail DNS, a participé à ces réunions en janvier, mai et septembre.

7 Publications

Livres et monographies

[1] C. Queinnec, *Lisp: implantation, sémantique, programmation*, InterÉditions, 1994.

Communications à des congrès, colloques, etc.

[2] A. Deutsch, «Interprocedural May-Alias Analysis for Pointers: Beyond k-Limiting», *in : SIGPLAN'94 Conf. on Programming Language Design and Implementation*, Association of Computing Machinery, p. 230–241, Orlando (Florida, USA), juin 1994. SIGPLAN Notices, 29(6).

[3] L. Moreau, C. Queinnec, «Partial Continuations as the Difference of Continuations, A Duumvirate of Control Operators», International Workshop PLILP '94 – Programming Language Implementation and Logic Programming, Springer-Verlag, Madrid (Spain), septembre 1994.

[4] C. Queinnec, «Locality, Causality and Continuations», *in : LFP '94 – ACM Symposium on Lisp and Functional Programming*, ACM Press, p. 91–102, Orlando (Florida, USA), juin 1994.

[5] C. Queinnec, «Sharing Mutable Objects and Controlling Groups of Tasks in a Concurrent and Distributed Language», *in : Proceedings of the Workshop on Theory and Practice of Parallel Programming (TPPP'94)*, T. Ito, A. Yonezawa (réd.), Springer-Verlag, Sendai (Japan), novembre 1994.

[6] M. Serrano, P. Weis, «1+1=1: An Optimizing Caml Compiler», *in : Record of the 1994 ACM SIGPLAN Workshop on ML and its Applications*, INRIA RR 2265, p. 101–111, Orlando (Florida, USA), juin 1994.

[7] M. Serrano, «Using Higher Order Control Flow Analysis When Compiling Functional Languages», *in : 6th International Symposium, PLILP'94*, M. Hermenegildo, J. Penjam (réd.), *Lecture Notes in Computer Science*, 844, p. 447–449, Madrid, Spain, septembre 1994.

Rapports de recherche et publications internes

[8] S. Anglade, J.-J. Lacrampe, C. Queinnec, «Semantics of Combinations in Scheme», à apparaître dans Lisp Pointers.

8 Abstract

The ICSLA team is a joint project of INRIA and École Polytechnique. As implied by its name which stands for *Interprétation, Compilation et Sémantique des Langages Applicatifs*, this project studies applicative languages and, chiefly, dialects of the Lisp family. Three axes can be identified:

Implementation: Various garbage collection techniques (conservative, concurrent, distributed or compile-time) are studied as well as

intermediate languages, separate compilation and compilation towards the C language.

Semantics: Object lifetime or sharing, closure analyses, partial evaluation applied to pattern matching are studied to improve compilation technology. New features such as partial continuation or modules are investigated as well.

Concurrency: Concurrency and distribution for networks of workstations are studied. This axis also analyses linguistic features and implementation techniques related to that goal such as coherency of shared mutable variables.

Most of the results (papers, tools and implementations) of the ICSLA project are accessible from Internet, via `ftp` or WWW, through
`file://ftp.inria.fr/INRIA/Projects/icsla/WWW`
(IP: 128.93.2.54).

Table des matières

1 Composition de l'équipe **1**

2 Présentation du projet **2**

3 Actions de recherche **2**

3.1 Langages intermédiaires 2

3.2 Interprétation abstraite et analyse interprocédurale d'alias existentiels pour langages à pointeurs 4

3.3 Evaluation partielle et compilation 4

3.4 Bigloo: compilateur de langages fonctionnels vers C 6

3.5 Interface entre langages 6

3.6 Continuations 7

3.7 Sémantiques de Lisp 7

3.8 Scheme distribué et parallèle 8

3.9 Sémantique de Scheme 9

3.10 Previsia 9

4 Actions industrielles **10**

5 Actions nationales et internationales **10**

6 Diffusion des résultats **11**

6.1 Diffusion de produits 11

6.2 Actions d'enseignement 11

6.2.1 Enseignement universitaire 11

6.2.2 Séminaires et formation permanente 12

6.2.3 Jurys de thèse 12

6.3 Participation aux manifestations 13

6.4 Activités extérieures 14

6.5 Réunions diverses 14

7 Publications **14**

8 Abstract **15**