Rapport INRIA 1994 — Programme 2

Programmation typée, modularité et compilation

PROJET CRISTAL

3 mai 1995

PROJET CRISTAL

Programmation typée, modularité et compilation

Localisation : *Rocquencourt*

Mots-clés : compilation (1), compilation séparée (1), environnement de programmation (1), génération de code (1), génie logiciel (1), lambda-calcul (1), programmation fonctionnelle (1), typage (1).

1 Composition de l'équipe

Responsable scientifique

Michel Mauny, Directeur de recherche, Inria

Responsable permanent

Pierre Weis, Chargé de recherche, Inria

Secrétariat (commun avec COQ et PARA)

Ghislaine Lecorre
Sylvie Loubressac

Conseiller scientifique

Guy Cousineau, Professeur, Univ. Paris 7 et ENS

Personnel Inria

Xavier Leroy, Chargé de recherche
Daniel de Rauglaudre, Ingénieur de recherche
Didier Rémy, Chargé de recherche
François Rouaix, Chargé de recherche

Chercheur invité

Benjamin Goldberg, New-York Univ., depuis 07/94

Chercheurs doctorants

Valérie Ménissier-Morain, ATER Évry
Patrick Parot, Bourse MESR, Univ. Orléans
François Pottier, ENS, (stagiaire DEA à partir de 06/94, doctorant depuis 10/94)
Christian Rinderknecht, Bourse MESR, Univ. Paris 6
Emilie Sayag, Bourse MESR, Univ. Paris 7

Stagiaires

Rowan Davies, Carnegie-Mellon Univ., de 06 à 09/94
François Pessaux, Stage de Maîtrise, Univ. Paris 6, de 05 à 07/94
Jérôme Vouillon, Stage de 1ère année, ENS

2 Présentation du projet

Le projet Cristal s'intéresse à différents formalismes de typage statique des langages de programmation et à leur utilisation dans la conception et la mise en œuvre d'outils de programmation typée robustes et efficaces.

Le typage statique accroît la sécurité de programmation, la rapidité de développement d'applications et facilite leur maintenance. Les systèmes de types sont aussi parmi les formalismes de base de recherches de preuves de programmes où les types sont vus comme des spécifications. Il s'agit là d'autant d'arguments montrant la nécessité d'environnement de programmation typée alliant fiabilité, sécurité et efficacité.

Le typage statique des langages de programmation impose aux programmes de satisfaire un certain nombre de propriétés. L'objectif de nos travaux est de tirer parti de ce fait dans la recherche d'une compilation efficace, dans la conception d'extensions de ces langages et dans la conception d'environnements de programmation typée. Nous nous intéressons aussi aux liens et aux interactions entre systèmes de preuves de programmes et environnements de programmation. Nos travaux se situent donc au carrefour de la théorie des types, de la conception et la mise en œuvre de langages de programmation et de la programmation proprement dite.

Caml et Caml Light, deux implantations d'un langage fonctionnel typé et puissant de la famille ML développées au sein de notre équipe, sont utilisés à la fois comme contexte d'étude, comme outils d'implantation et comme banc d'essai des outils que nous développons.

Le projet Cristal dirige actuellement ses efforts vers l'amélioration de l'environnement de programmation. Cette notion d'environnement doit être comprise au sens large, c'est-à-dire incluant :

- les systèmes de modules, permettant la réalisation d'applications de taille réelle tout en conservant la sécurité de programmation propre aux langages typés ;
- des outils de mise au point pour langages statiquement typés ;
- des outils graphiques destinés à la programmation d'interfaces ;
- des facilités de manipulation de syntaxes concrètes, programmables par l'utilisateur, permettant d'écrire des programmes complexes de façon concise.

Cette année 1994 a été marquée par différentes réalisations concrétisant nos efforts en direction d'un environnement de programmation pour Caml-Light. A noter particulièrement la réalisation d'une première version de débogueur pour Caml Light.

3 Actions de recherche

3.1 Environnement Caml Light

3.1.1 Compilateur Caml-Light

Participant : Xavier Leroy

Xavier Leroy a mis sur pied la version 0.7 du système Caml Light, rendue publique fin décembre 1994. Les principales nouveautés de cette version, le débogueur source et l'interface avec la boîte à outils graphique Tk, sont décrites séparément dans la suite de ce rapport. Cette version s'enrichit également d'une détection exacte des filtrages partiels ou superflus (contribution de Luc Maranget, projet Para) et d'un mécanisme de gardes dans les clauses des filtrages. Elle améliore également la compatibilité avec les processeurs 64 bits et le système MS-Windows, et corrige un certain nombre de problèmes mineurs de la version précédente.

3.1.2 Le débogueur de Caml Light

Participants : Xavier Leroy, Jérôme Vouillon

Jérôme Vouillon (élève ENS en stage de première année) a réalisé un débogueur source pour le système Caml Light, sur des idées de Damien Doligez (projet Para). Ce débogueur facilite considérablement la mise au point des programmes Caml par rapport aux outils de "trace" fournis jusqu'ici par le système.

Ce débogueur permet un contrôle très précis de l'exécution du programme en cours de mise au point: pose de points d'arrêt, exécution pas-à-pas, examen des variables locales et globales. La principale originalité est l'exécution inverse du programme (*replay debugging*), c'est-à-dire la possibilité de reconstituer l'état du programme à n'importe quel instant de son exécution, permettant ainsi à l'utilisateur de revenir par exemple "un pas en arrière", ou jusqu'au précédent point d'arrêt.

La réalisation de ce débogueur fait appel, pour garder trace des états successifs du programme, une technique de *checkpointing* reposant sur les appels système Unix de gestion des processus. Cette technique, quoiqu'extrêmement simple, se révèle étonnamment efficace. Le débogueur opère directement sur le code compilé ; aucune instrumentation du code source n'est donc nécessaire, ce qui garantit de bonnes vitesses de compilation et d'exécution.

3.1.3 Interface Tcl/Tk

Participants : François Rouaix, François Pessaux

François Rouaix et François Pessaux ont réalisé une interface Tcl/Tk pour Caml [32]. Tcl (*Tool Command Language*) est un langage interprété, destiné à étendre et contrôler des applications. Tk est une bibliothèque Tcl implantant une toolkit X-Windows puissante et simple à programmer, c'est-à-dire une bibliothèque pour développer des interfaces homme-machine comportant fenêtres, boutons, menus et autres éléments graphiques. La bibliothèque CamlTk donne accès à la programmation simple et sûre d'interfaces homme-machine en Caml. Le premier prototype obtenu pendant le stage de François Pessaux a été repris, étendu, et amené à un stade approprié à la distribution à la communauté Caml. Des prototypes d'outils interactifs d'aide au dévelop-

pement (*browser, toplevel*), écrits grâce à cette interface, ont également été diffusés.

3.1.4 Profileur

Participant : François Rouaix

En collaboration avec Damien Doligez (projet Para), François Rouaix a réalisé un profileur pour Caml Light. Un profileur mesure le nombre de passages d'un programme à certains points de contrôle dans le source (appels de fonctions, branches d'un pattern-matching ou d'un `if ... then ... else`, boucles). Après exécution du programme, l'utilisateur peut voir une version des sources de son programme annotée avec le nombre de passage à chaque point de contrôle lors de l'exécution.

3.1.5 Indenteur

Participant : Michel Mauny

Michel Mauny a expérimenté une façon originale de réindenter et d'imprimer en Latex les programmes Caml. L'impression Latex de programmes les rend beaucoup plus lisibles, car elle exhibe mieux leur structure (mots-clés en gras, par exemple). Un tel traitement fait, en général, perdre les commentaires internes au programmes original. C'est le défaut habituel des *pretty-printers*.

Michel Mauny a réalisé une application nommée Camlp à la base de laquelle se trouve un réindenteur de programmes. Préserver les commentaires étant l'objectif initial de Camlp, cette application réalise l'analyse syntaxique du programme, attachant les espaces, sauts de lignes et commentaires aux lexèmes. Elle construit ensuite une arborescence de "boîtes" à partir de laquelle un moteur d'impression produit ou bien le programme initial simplement réindenté ou alors du code Latex.

Cette application permet par ailleurs le traitement de programmes Caml contenus dans un document Latex, traitant de manière distincte programmes d'entrée, les réponses et messages d'erreur de Caml-Light. Enfin, Camlp reconnaît les programmes incomplets (c'est-à-dire contenant des holophrastes) et le traitement de commentaires spéciaux permettant par exemple, de rédiger dans le même fichier Caml un programme et sa documentation sous forme de commentaires.

Camlp et sa documentation sont distribués par ftp depuis décembre 94.

3.2 Réutilisabilité

Participants : Patrick Parot, François Rouaix

L'expérience Alcool, entamée depuis plusieurs années par François Rouaix, promeut une forme de réutilisabilité basée sur la surcharge et les types abstraits. Le prototype Alcool, extension de Caml Light, a été utilisé par François Barthélémy (projet Chloe) dans le cadre de son travail sur une approche uniforme de l'analyse syntaxique avec contraintes. Ce travail s'est concrétisé par la réalisation du prototype Apoc écrit en Alcool. Grâce à une architecture uniforme et un ensemble d'environ 40 modules, il est possible de produire une vingtaine d'analyseurs syntaxiques implantant des stratégies différentes. Cette implémentation a servi de banc d'essai pour le langage Alcool, et le travail a donné lieu à deux communications [11, 12].

Patrick Parot a poursuivi son travail de thèse sur la conception et la réalisation d'algorithmes pour la recherche dans une bibliothèque de composants logiciels réutilisables. Cette approche est une généralisation des mécanismes de réutilisation esquissés dans le système Alcool, et tente de définir une architecture générale pour la recherche de composants, se basant plus sur la syntaxe des spécifications que sur leur sémantique. Patrick Parot s'est plus particulièrement intéressé cette année aux composants d'ordre supérieur. Ceux-ci imposent une approche différente de celle utilisée pour les composants d'ordre un. Ces derniers peuvent en effet être modélisés assez simplement en programmation logique (clauses de Horn), alors que l'ordre supérieur demanderait une logique d'ordre supérieur pour laquelle les outils de résolution ne sont pas disponibles. Toutefois, les particularités des spécifications des composants étudiés par Patrick Parot permettent de restreindre le problème de la recherche et donc de développer un algorithme pratique. Une formalisation de cet algorithme (pour des spécifications utilisant des termes du premier ordre) a été définie et soumise à publication [31]. Un prototype a également été développé et est en cours de test. Les travaux de Patrick Parot sont à la base de la coopération de l'Inria dans le projet Eureka SCI.

3.3 Typage

Nos travaux sur le typage ont pout but d'accroître l'expressivité des langages fortement typés, de fournir de meilleurs fondements à leurs sys-

tèmes de types, et de mieux comprendre le lien entre typage et exécution des programmes.

3.3.1 Polymorphisme extensionnel

Participants : François Rouaix, Pierre Weis

Pierre Weis et François Rouaix ont travaillé en collaboration avec Catherine Dubois (Université d'Évry) sur le polymorphisme extensionnel (précédemment intitulé "typage des implicites"). Ce formalisme introduit la notion de fonctions génériques, qui sont des fonctions polymorphes dont la valeur dépend de leurs types d'usage. Contrairement aux fonctions polymorphes paramétriques habituelles de ML, ces fonctions sont définies par cas sur un ensemble de schémas de types, et requièrent le passage implicite de types en argument lors de l'exécution.

Cette forme de polymorphisme fournit une solution simple aux problèmes de la surcharge, des fonctions d'impression ou d'entrée/sortie, et ouvre la voie à un nouveau style de programmation où le calcul combine types et données. Nous avons étudié une extension simple du système de type de ML, assurant le typage fort des programmes contenant des génériques. Ce travail a donné lieu à une communication [14].

Une implémentation de ce formalisme, basée sur le compilateur Caml Light, a été réalisée. Les implications du polymorphisme extensionnel de ce formalisme sont encore à explorer: relations avec les langages Alcool et Haskell, programmation par objets, types dépendants, types dynamiques. François Rouaix et Pierre Weis ont déjà abordé la programmation par objets dans ce contexte, et les premiers résultats sont encourageants.

3.3.2 Programmation à objets dans un langage fortement typé

Participants : Didier Rémy, François Pottier, Rowan Davies

Didier Rémy a fait porter l'essentiel de ses activités de recherche sur l'ajout d'objets dans le langage ML. La composante principale de ce travail repose sur le langage expérimental ML-ART, une extension de ML avec des types existentiels, des enregistrements extensibles et des types récursifs dans laquelle il est possible de programmer ses propres

objets [22]. Les résultats plus récents sont d'une part l'ajout de conversion des objets d'une classe vers leur contre-partie dans une classe parente, d'autre part la complétude de la synthèse de types dans le langage ML-ART obtenue en montrant que l'unification dans les algèbres d'enregistrements s'étend aux termes rationnels.

Didier Rémy a encadré le stage d'été de Rowan Davies de l'université Carnegie Mellon (USA). Rowan Davies a étudié et réalisé une extension de ML où les objets et les classes sont primitifs en conservant les idées essentielles de ML-ART, notamment le fait de pouvoir paramétrer les classes par des classes parentes. Cette nouvelle fonctionnalité, originale, semble très utile en pratique.

François Pottier commence actuellement une thèse, sous la direction de Didier Rémy. Il s'agit d'étudier les liens entre modules et objets, dans le cadre des langages à synthèse de types statique.

3.3.3 Typage des constructions impératives

Participant : Didier Rémy

Dans la continuation de ses travaux sur la comparaison des sémantiques opérationnelles à réduction et de la sémantique naturelle, Didier Rémy a collaboré avec Jon Riecke (Bell Labs) et Carl Gunter (Université de Pennsylvanie) sur le typage des continuations [25]. Cette étude a abouti à une simplification des opérations de contrôle et à une preuve de correction forte (le type du résultat est relié au type du programme, ce qui n'est pas le cas dans une preuve de correction faible) pour les opérateurs de contrôle. Les nouveaux opérateurs font apparaître une analogie avec les exceptions de ML: déclaration d'un nouveau point de contrôle, dépose d'un point de contrôle, capture du contrôle jusqu'au point de contrôle le plus proche.

3.3.4 Reconstruction dynamique de types

Participants : Michel Mauny, Émilie Sayag

Le problème de la reconstruction dynamique de types, composante essentielle dans la construction d'outils de mise au point fiables, n'a jusqu'alors reçu comme solution que celle consistant à passer les types en

argument des fonctions polymorphes. Même si Andrew Tolmach a montré que le surcoût entraîné à l'exécution était raisonnable, cette solution repose sur une restriction du polymorphisme de ML, et sa correction n'a pas été prouvée. Le problème de cette correction est l'absence de "référence sémantique" fiable.

Au cours de sa première année de thèse, Emilie Sayag a étudié plusieurs sémantiques dans ce cadre. Les différentes sémantiques proposées jusqu'alors se sont toutes révélées insatisfaisantes : soit à cause de leur complexité, soit à cause du caractère partiel de la reconstruction. Émilie Sayag s'attaque donc à la définition d'une telle référence sémantique, qui devrait fournir une meilleure compréhension du comportement dynamique des types durant l'exécution de programmes polymorphes statiquement typés.

3.3.5 Typage et programmation distribuée

Participants : Michel Mauny, Christian Rinderknecht

Dans le cadre de sa première année de thèse, Christian Rinderknecht s'est intéressé aux langages de spécification de protocoles d'application. Il a ainsi entrepris, en collaboration avec le CNET de Lannion, une étude approfondie du langage ASN.1 '90 (*Abstract Syntax Notation One*), qui est un langage normalisé par l'ISO utilisé fréquemment dans les télécommunications afin de spécifier le type des données transmises sur le réseau. Il a mis en évidence les difficultés syntaxiques qui s'opposaient jusqu'alors à la réalisation d'outils automatiques d'analyse des spécifications, sans compromis avec la norme internationale, et a fourni une grammaire LL(1) du langage. Un analyseur syntaxique complet (macros ASN.1 exceptées) en Caml Light en a été dérivé, laissant la voie ouverte et bien dégagée pour un analyseur sémantique (vérificateur de types). La puissance d'expression de la pleine fonctionnalité et la sécurité du typage statique fort ont été des atouts majeurs pour cette réalisation. Un rapport technique décrivant ce travail est en préparation.

3.4 Modularité

Participants : Benjamin Goldberg, Xavier Leroy, François Pottier

Xavier Leroy a poursuivi son travail de conception et de formalisation de systèmes de modules. Ce travail a pour but pratique de doter Caml

Light d'un système de modules expressif mais qui se prête cependant à la compilation séparée ; il apporte également une nouvelle approche formelle pour décrire les systèmes de modules, qui fait le lien avec les systèmes de types classiques. Le système de modules résultant de ce travail [17] atteint la puissance de celui de Standard ML, la référence actuelle en la matière, mais se révèle plus simple à formaliser et plus pratique à utiliser. En particulier, ce système de modules se prête à la compilation séparée à la manière de Modula-2, avec détection immédiate des erreurs de types inter-fragments, au contraire des modules de Standard ML, qui repoussent la détection de ces erreurs au moment de l'édition de liens.

Cette année, Xavier Leroy a complété l'étude théorique de ce système de modules en prouvant formellement que sa restriction au premier ordre offre la même expressivité que le système de modules de Standard ML [18]. Pour étendre ce résultat à l'ordre supérieur, Xavier Leroy a conçu une extension de son système qui offre des foncteurs d'ordre supérieur d'une expressivité comparable (en fait, légèrement supérieure) aux foncteurs "complètement transparents" de New Jersey ML [19]. Cette extension repose sur une sémantique non standard des foncteurs, la sémantique applicative (deux applications d'un foncteur au même type abstrait renvoient le même type abstrait), déjà proposée par François Rouaix dans le contexte du langage Alcool.

Dans le cadre de son stage de DEA, François Pottier a réalisé une implémentation "grandeur nature" de ce système de modules dans le compilateur Caml Light, permettant ainsi l'expérimentation de ce système sur des programmes réalistes. Un soin particulier a été apporté à l'efficacité du typage et des vérifications de cohérence entre modules ; l'implémentation n'est donc pas un décalque de la définition formelle du système de modules, mais utilise des structures de données et des algorithmes plus complexes. François Pottier a également conçu et réalisé la compilation du langage de modules vers le langage intermédiaire de Caml Light. Ce prototype va être rendu public pour recueillir les réactions des utilisateurs, et a déjà identifié des parties du compilateur Caml Light qui devront être refondues pour mieux s'adapter au nouveau système de modules.

Benjamin Goldberg est professeur à New York University, et consacre son séjour à l'Inria à l'étude de l'intégration de systèmes de modules dans les systèmes de types de langages comme ML. Les fondements

d'une telle intégration existent dans divers lambda-calculs d'ordre supérieur, tels F_ω, où des systèmes de types fournissent vérification et inférence. Ce domaine connaît des avancées régulières, mais il reste cependant un certain nombre de problèmes ouverts. Benjamin Goldberg s'intéresse à ces problèmes, ainsi qu'à l'application de résultats du domaine de la théorie des types dans la conception de langages réalistes. Par exemple, il a été montré que l'inférence de types était décidable pour plusieurs de ces systèmes, mais il est probable qu'une inférence partielle soit suffisante dans la pratique, et puisse servir de base à la conception d'un environnement de programmation modulaire. Seules la conception et l'expérimentation de tels environnements permettent une évaluation réaliste de ces résultats de recherche théorique.

3.5 Arithmétique des grands nombres

Participants : Valérie Ménissier-Morain, Pierre Weis

Valérie Ménissier-Morain a rédigé et soutenu sa thèse [4], dans laquelle elle a mis en évidence la nécessité d'une arithmétique exacte dans un langage de programmation moderne. L'insuffisance des solutions apportées jusqu'à présent ont conduit à l'implémentation d'une arithmétique rationnelle exacte et efficace en Caml [7] et à la description d'une arithmétique réelle complète en précision arbitraire. Valérie Ménissier-Morain a prouvé la correction de cette description et un prototype en a été implémenté.

3.6 Compilation

3.6.1 Langages intermédiaires

Participant : Pierre Weis

En collaboration avec Bernard Serpette du projet Icsla, Pierre Weis a poursuivi ses recherches sur les langages intermédiaires, dans l'optique d'obtenir des compilateurs performants facilement portables. Ce travail a conduit à l'écriture d'une maquette de compilateur Caml. Ce compilateur produit un assembleur virtuel, dérivé de la pico-machine de [3]. Pour démontrer le bien-fondé de l'approche, l'assembleur a été porté sur plusieurs processeurs : Mips, Sparc, HP-Snake et DEC-Alpha. Cette expérience est très encourageante puisque les portages ont été réalisés rapidement et sans difficultés majeures. Pour compléter la gestion mémoire

de la pico-machine, un GC *Stop&Copy* a été ajouté à la maquette. Pierre Weis envisage d'étendre le compilateur au langage Caml tout entier.

3.6.2 Compilateur optimisant pour Caml-Light

Participant : Pierre Weis

En collaboration avec Manuel Serrano du projet Icsla, Pierre Weis 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 (en anglais *front-end*) 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, et le second est le compilateur Scheme Bigloo, réalisé par Manuel Serrano. Le nouveau compilateur [23] 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 curryfiées. Il produit du code dont l'efficacité est comparable avec celle du code produit par les meilleurs compilateurs ML.

Ce compilateur Caml optimisant 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 par ftp et est déjà couramment utilisé.

3.7 Syntaxes extensibles

Participants : Michel Mauny, Daniel de Rauglaudre

Les types structurés définissables et la pleine fonctionnalité confèrent aux langages ML une grande expressivité. Afin d'exploiter au mieux cette expressivité, on ressent souvent le besoin de disposer de constructions syntaxiques définissables par l'utilisateur afin d'exprimer de façon plus concise et plus lisibles certaines expressions. Les constructions et destructions de valeurs structurées complexes peuvent être traitées de façon élégante par une syntaxe concrète appropriée utilisée entre "guillemets". Michel Mauny et Daniel de Rauglaudre ont poursuivi et publié leur travail sur ce sujet [21], obtenant un langage extensible par des

constructions programmables représentant des données arbitrairement complexes. La mise en œuvre de Daniel de Rauglaudre allie efficacité et compatibilité avec la modularité et la compilation séparée.

L'expressivité de ML ne repose pas sur les seuls types structurés définissables, mais aussi sur l'utilisation de combinaisons fonctionnelles arbitrairement complexes (la "pleine fonctionnalité") fournissant une grande richesse en terme de styles de programmation. Même si le typage statique est d'une grande utilité dans ce cadre, le programmeur peut être limité par la complexité des combinaisons qu'il désire utiliser, ses programmes pouvant devenir trop difficiles à lire et donc à maintenir. Il s'agit donc de disposer d'un support syntaxique programmable plus spécialement destiné au contrôle. Le seul support disponible jusqu'à présent dans les langages ML est la possibilité de définir ses propres opérateurs infixes, et s'avère insuffisant dans bien des cas. Il apparaît nécessaire de disposer de nouvelles constructions syntaxiques définissables permettant de donner une syntaxe particulière à telle ou telle construction complexe correspondant à un style de programmation particulier.

S'inspirant du travail d'Anika Aasa (Göteborg), qui proposait dans sa thèse une syntaxe extensible par des opérateurs dits *distfixes*, Michel Mauny et Daniel de Rauglaudre ont conçu et mis en œuvre une généralisation de tels opérateurs, suffisamment puissante pour, par exemple, rendre définissable toute la syntaxe concrète des expressions ML (à l'exception des atomes). Cette forme d'extensibilité, nouvelle en ML et très facile d'usage, est à même d'être employée pour programmer un support syntaxique rendant utilisable tel ou tel style de programmation dans toute sa puissance, et le rendre disponible sous forme de bibliothèque.

4 Action industrielles

Projet Eureka SCI

Participants : François Rouaix, Patrick Parot

Le Projet Eureka SCI (*Software Components for the Industry*), dont le thème est la réutilisabilité au travers de bibliothèques de composants, a commencé courant 94; notre partenaire principal, dans ce projet, est la société Verilog. François Rouaix et Patrick Parot travaillent à la concep-

tion et la réalisation d'algorithmes pour une recherche automatique de composants. Notre outil principal est le système Alcool.

5 Actions nationales et internationales

Pôle Programmation Fonctionnelle du GDR

Michel Mauny est responsable de l'équipe "Typage et Programmation en ML" (INRIA, LRI, LIENS, LITP), du Pôle "Programmation Fonctionnelle" et membre de l'équipe de direction du GDR "Programmation".

Nous avons organisé nos troisièmes "Journées de Pôle" à Lille en septembre.

Projet "Méthodes formelles et langages de haut niveau"

Nous avons soumis à la DRED une proposition de projet intitulée "Méthodes formelles et langages de haut niveau", impliquant essentiellement les équipes ML, TAF et TEACUP du GDR, et avons reçu un soutien financier de la part de cet organisme.

ML 2000

Xavier Leroy est membre invité du comité ML2000, qui réunit un certain nombre de spécialistes travaillant sur ML et la sémantique, dans le but de concevoir une version modernisée du langage ML intégrant les avancées récentes dans ce domaine. Xavier Leroy a participé à la réunion de janvier 1994 de ce comité.

Projet d'Ingéniérie Logicielle

Guy Cousineau participe à une action de recherche en coopération avec Dassault-Aviation et le projet Coq visant à la mise au point d'une chaîne de production de logiciels utilisant le système Coq et le compilateur CeML développé au LIENS. Ce projet a été accepté dans le cadre d'une réponse à un appel d'offre du MRES.

Comités de lecture et de programmes

Michel Mauny est membre du comité de lecture de la revue *Fifth Generation Computing*.

Didier Rémy a organisé le programme du séminaire *ACM Sigplan Workshop on ML and its applications* qui a eu lieu en Juin à Orlando (USA). Guy Cousineau était membre de ce comité de programme. Le projet Cristal en a publié les actes sous la forme d'un rapport de recherche [2].

Xavier Leroy a été membre du comité de sélection du symposium *Static Analysis Symposium* 1994.

François Rouaix était membre du comité de programme du second *Workshop on Configurable Distributed Systems*, auquel il a assisté (Pittsburgh, Pennsylvanie, 21–23 mars 1994).

Pierre Weis a été membre du comité de programme des Journées Francophones des Langages Applicatifs (JFLA 94). Il est aussi le modérateur de la tribune de discussion de Caml et Caml Light (250 abonnés).

Directions de stages, thèses

Michel Mauny encadre les thèses d'Emilie Sayag et de Christian Rinderknecht (en collaboration avec Bernard Lorho).

Didier Rémy encadre la thèse de François Pottier. Didier Rémy a encadré le stage d'été de Rowan Davies de l'université Carnegie Mellon (USA).

François Rouaix encadre la thèse de Patrick Parot et a encadré le stage de François Pessaux, étudiant de M.S.T. de l'Université Paris 6, portant sur la conception et la réalisation d'une interface Tcl/Tk pour Caml.

Xavier Leroy a encadré le stage de DEA de François Pottier ainsi que le stage de Jérôme Vouillon.

Séminaire Coq-Cristal-Para

Le séminaire Coq-Cristal-Para, organisé par Didier Rémy, a accueilli cette année une trentaine d'orateurs internationaux.

6 Diffusion des résultats

Les résultats de l'équipe sont largement diffusés, aussi bien en ce qui concerne les logiciels que les résultats scientifiques dans les colloques nationaux, internationaux et dans des publications.

6.1 Diffusion de produits

Nos logiciels (Caml, Caml Light, Alcool, Camloo) sont en accès libre par FTP anonyme sur la machine `ftp.inria.fr`, dans le répertoire `INRIA/Projects/cristal`. Les versions Macintosh et PC de Caml Light sont également distribuées sous forme de disquettes par le Centre de

Diffusion de l'Inria. La distribution de Caml Light est régulièrement en tête du *hit-parade* des logiciels Inria transférés par FTP (de l'ordre de 1500 transferts cette année).

Le langage Caml connaît un certain succès comme langage d'enseignement, aussi bien dans des cours d'initiation à la programmation que dans des cours de second et troisième cycle universitaire. Thérèse Hardin, Véronique Donzeau-Gouge et Pierre Weis ont fait un effort de publication [5, 15, 26] en direction des milieux éducatifs afin de démontrer l'apport méthodologique de Caml pour la pédagogie de l'Informatique.

En outre, les livres de Pierre Weis et Xavier Leroy *Le langage Caml* et *Manuel de référence du langage Caml*, parus l'an dernier chez InterÉditions, sont employés comme support de cours dans plusieurs universités.

François Rouaix et Xavier Leroy ont mis sur pied un serveur World Wide Web (`http://pauillac.inria.fr/`) qui donne accès à un certain nombre de réalisations du projet Cristal: les logiciels Caml, Caml Light, Alcool, leurs documentations, et certaines publications du projet. Le but est d'une part d'améliorer le service fourni à nos utilisateurs, et d'autre part d'expérimenter les moyens nouveaux de diffusion des résultats de la recherche qu'offre le Web.

6.2 Actions d'enseignement

6.2.1 Enseignement universitaire

Universités – 3ème cycle

Guy Cousineau, Michel Mauny et Didier Rémy donnent un cours dans le DEA d'Informatique Fondamentale de Paris 7 sur la programmation fonctionnelle typée, les propriétés du typage des langages ML et de ses extensions ainsi que sur les techniques de preuve y afférant.

École Polytechnique

Michel Mauny, Didier Rémy et Pierre Weis sont Chefs de Travaux Dirigés d'Informatique à temps partiel à l'École Polytechnique. À ce titre, ils enseignent dans le tronc commun de première année (52h). En outre, Michel Mauny et Pierre Weis assurent un TP de Programmation Fonctionnelle et Complexité de la Majeure Algèbre-Informatique (cours de Jean-Marc Steyaert).

François Rouaix a participé à l'enseignement de Systèmes et Réseaux de la Majeure 2 d'Informatique de l'Ecole Polytechnique (20h) ainsi qu'à l'enseignement d'informatique de tronc commun (52h).

CNAM

Pierre Weis est professeur associé au CNAM (contrat Barre) où il assure des cours de Caml en amphithéâtre avec 450 étudiants (2 heures hebdomadaires).

Écoles d'Ingénieurs

Michel Mauny a donné une semaine de cours de Programmation Fonctionnelle en septembre, à l'ISIA (École des Mines de Paris, Sophia-Antipolis).

Universités – 1er et 2ème cycle

Valérie Ménissier-Morain a assuré, pendant le second semestre de l'année universitaire 1993-1994, des TD/TP de Maple (36h) en DEUG MIAS à l'Université de Versailles Saint-Quentin en Yvelines. Elle occupe depuis septembre 1994 un demi-poste d'ATER à l'Université d'Évry Val d'Essonne (96h) où elle enseigne l'algorithmique à travers Caml en licence de mathématiques et la programmation fonctionnelle avec Caml en seconde année de MIAGE.

Christian Rinderknecht a un poste de Moniteur à l'Université Paris 7, dans le cadre duquel il assure des travaux dirigés d'Algèbre Linéaire en première année d'IUP d'Informatique.

6.2.2 Séminaires et formation permanente

Michel Mauny a donné un cours de Programmation Fonctionnelle à l'école de jeunes chercheurs du GDR Programmation, qui s'est tenue à Toulouse.

6.2.3 Jurys de thèse

Michel Mauny a été rapporteur de la thèse de Yang Mei-Tang (mars 94, Paris 6).

Didier Rémy a été rapporteur de la thèse de Guiseppe Castagna (janvier 94, Paris 7).

Michel Mauny a été membre du jury de la thèse de Manuel Serrano (décembre 94, Paris 6).

Pierre Weis était membre du jury de la thèse de Valérie Ménissier-Morain.

6.3 Participation aux manifestations

POPL

Xavier Leroy a participé au congrès *Principles of Programming Languages* (Portland, Oregon, 17–21 janvier 1994) et y a présenté son article sur les modules et la compilation séparée.

LFP, MLW

Xavier Leroy, Michel Mauny, Daniel de Rauglaudre, Didier Rémy, François Rouaix et Pierre Weis ont assisté à la conférence *Lisp and Functional Programming* ainsi qu'au *ML Workshop* qui se sont tenus à Orlando en juin 94. Didier Rémy était président du Comité de Programme du *ML Workshop*.

Xavier Leroy y a présenté son travail comparant l'expressivité des systèmes de modules, Michel Mauny et Daniel de Rauglaudre leur travail sur les guillemets (*quotations*), François Rouaix son travail commun avec François Barthélémy (Chloe) sur le système Apoc, et Pierre Weis et Manuel Serrano (Icsla) y ont présenté leur travail commun sur la compilation optimisée de Caml.

LICS, FOOL

Didier Rémy a participé à la conférence LICS (*Logic in computer Science*) en juillet (Paris) ainsi qu'au groupe de travail FOOL (*Foundations of Object-Oriented Languages*) qui a eu lieu à Paris début juillet où il a présenté une version plus avancée de ses travaux sur la programmation des objets dans ML-ART.

TACS

Didier Rémy a assisté à la conférence TACS (*Theoretical Aspects of Computer Science*) à Tokyo (Japon) en avril où il a présenté ses travaux récents sur la programmation d'objets dans le langage ML-ART [22]. À cette occasion, il a rendu visite à A. Ohori à l'Université de Kyoto.

CONFER

Didier Rémy a participé au séminaire et à la revue du BRA CONFER à Imperial College (GB) en septembre.

JFLA

La majeure partie de l'équipe a participé aux Journées Francophones des Langages Applicatifs qui se sont déroulées à Noirmoutiers en février, ainsi qu'à la réunion du Pôle "Programmation Fonctionnelle" du GDR Programmation organisée par Michel Mauny.

Journées du GDR Programmation

La rencontre annuelle du GDR Programmation a eu lieu en septembre à Lille, et Michel Mauny y a organisé une Journée du Pôle "Programmation Fonctionnelle". Didier Rémy, Émilie Sayag et Pierre Weis y ont participé. Pierre Weis et Manuel Serrano (Icsla) ont présenté leur travail commun aux journées plénières. Didier Rémy a présenté son travail sur ML-ART à la journée de Pôle.

7 Publications

Livres et monographies

[1] X. Leroy, P. Weis, *Manuel de référence du langage Caml*, InterÉditions, juillet 1993.

[2] D. Rémy (réd.), *Record of the 1994 ACM-SIGPLAN Workshop on ML and its Applications*, *Rapport de Recherche INRIA*, 2265, juin 1994.

[3] P. Weis, X. Leroy, *Le langage Caml*, InterÉditions, juillet 1993.

Thèses

[4] V. Ménissier-Morain, *Arithmétique exacte: conception, algorithmique et performances d'une implémentation informatique en précision arbitraire*, Thèse, Université Paris 7, décembre 1994.

Articles et chapitres de livre

[5] T. Hardin, V. Donzeau-Gouge, P. Weis, «Apprendre la programmation à l'aide d'un modèle Sémantique de Caml et Ada», *Technique et Science Informatique 12*, 4, 1993.

[6] X. Leroy, M. Mauny, «Dynamics in ML», *Journal of Functional Programming 3*, 4, octobre 1993, p. 431–463.

[7] V. Ménissier-Morain, P. Weis, «An exact arithmetic package for ML», *Science for Computer Programming*, 1994, À paraître.

[8] D. Rémy, «Type Inference for Records in a Natural Extension of ML», *in : Theoretical Aspects Of Object-Oriented Programming. Types, Semantics and Language Design*, C. A. Gunter et J. C. Mitchell (réd.), MIT Press, avril 1994.

[9] D. Rémy, «Typing Record Concatenation for Free», *in : Theoretical Aspects Of Object-Oriented Programming. Types, Semantics and Language Design*, C. A. Gunter et J. C. Mitchell (réd.), MIT Press, avril 1994.

Communications à des congrès, colloques, etc.

[10] M.-V. Aponte, X. Leroy, «Llamado de procedimientos a distancia y abstracción de tipos», *in : Actes du 20e congrès latino-américain d'informatique CLEI-PANEL*, p. 1281–1292, septembre 1994.

[11] F. Barthélemy, F. Rouaix, «Abstract Data-Types and Operators: an Experiment in Constraint-Based Parsing», *in : Record of the 1994 ACM-SIGPLAN Workshop on ML and its Applications*, p. 34–40, juin 1994.

[12] F. Barthélemy, F. Rouaix, «A Modular Architecture for Constraint-Based Parsing», *in : Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, p. 454–460, Kyoto, Japan, août 1994.

[13] D. Doligez, X. Leroy, «A concurrent, generational garbage collector for a multithreaded implementation of ML», *in : Actes du 20e symposium Principles of Programming Languages*, ACM Press, p. 113–123, janvier 1993.

[14] C. Dubois, F. Rouaix, P. Weis, «Extensional polymorphism», *in : Actes du 22e symposium Principles of Programming Languages*, ACM Press, janvier 1995. À paraître.

[15] T. Hardin, V. Donzeau-Gouge, P. Weis, «Débuter en programmation avec Caml puis Ada», *in : Specif numéro spécial 93*, avril 1993.

[16] X. Leroy, «Polymorphism by name for references and continuations», *in : Actes du 20e symposium Principles of Programming Languages*, ACM Press, p. 220–231, janvier 1993.

[17] X. Leroy, «Manifest types, modules, and separate compilation», *in : Actes du 21e symposium Principles of Programming Languages*, ACM Press, p. 109–122, janvier 1994.

[18] X. Leroy, « A syntactic approach to type generativity and sharing (extended abstract) », *in : Record of the 1994 ACM-SIGPLAN Workshop on ML and its Applications*, INRIA, p. 1–12, juin 1994.

[19] X. Leroy, « Applicative functors and fully transparent higher-order modules », *in : Actes du 22e symposium Principles of Programming Languages*, ACM Press, janvier 1995. À paraître.

[20] M. Mauny, D. de Rauglaudre, « Analyseurs lexico-syntaxiques en ML », *in : Actes des Jounées Francophones des Langages Applicatifs*, Annecy, 1993.

[21] M. Mauny, D. de Rauglaudre, « A complete and realistic implementation of quotations for ML », *in : Record of the 1994 ACM-SIGPLAN Workshop on ML and its Applications*, juin 1994.

[22] D. Rémy, « Programming Objects with ML-ART: An extension to ML with Abstract and Record Types », *in : International Symposium on Theoretical Aspects of Computer Software*, M. Hagiya, J. C. Mitchell (réd.), Springer-Verlag, p. 321–346, Sendaï, Japon, avril 1994.

[23] 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, juin 1994.

Rapports de recherche et publications internes

[24] M. Abadi, L. Cardelli, B. Pierce, D. Rémy, « Dynamic Typing in Polymorphic Languages », *Research Report n° 120*, Digital Equipment Coorporation, System Research Center, 130 Lytton Avenue, Palo ALto, CA 94301, USA, janvier 1994, À paraître dans *Journal of Functional Programming*.

[25] C. A. Gunter, R. Rémy, J. G. Riecke, « Prompting ML », Manuscrit non publié, août 1994.

[26] T. Hardin, V. Donzeau-Gouge, P. Weis, « Teaching Programming via a Semantic Model of Caml and Ada », Software Engeneering Education Workshop (ICSE-16), 1994.

[27] X. Leroy, *The Caml Light system, release 0.7 – Documentation and user's manual*, INRIA, décembre 1994, Documentation distribuée avec le système Caml Light.

[28] M. Mauny, F. Pottier, « An implementation of Caml Light with existential types », *Rapport de recherche n° 2183*, INRIA, octobre 1993.

[29] M. Mauny, *The Camlp user's manual*, INRIA, 1994, En préparation.

[30] M. Mauny, *Functional programming using Caml Light (version 0.7)*, INRIA, décembre 1994, Documentation distribuée avec le système Caml Light.

[31] P. Parot, « Automatisation d'une bibliothèque de modules », soumis à publication (JFLA 95), septembre 1994.

[32] F. Pessaux, «Conception et réalisation d'une interface entre Caml Light et Tcl/Tk », Rapport de stage de 1ère année de M.S.T, Université Paris 6, septembre 1994.

[33] F. Pottier, *Implémentation d'un système de modules évolué en Caml-Light*, Mémoire de DEA, ENS, juillet 1994.

[34] M. Serrano, P. Weis, « 1+1 = 1, an optimizing Caml compiler », *Rapport de recherche n° 2301*, INRIA, juillet 1994.

8 Abstract

The Cristal project is concerned with different formalisms of static typing for programming languages, and their use in the design and implementation of robust and efficient tools for typed programming.

Static typing improves programming security, allows for the fast development of applications and helps in maintaining them. Type systems are also amongst the basic formalisms used in the research on correctness proofs for programs, where types are seen as program specifications. Hence the need for environments dedicated to typed programming, combining reliability, programming safety and efficiency.

Static typing of programming languages imposes that programs satisfy some properties. The goal of our project is to take advantage of these properties in research for efficient compilation and in the design of language extensions and programming environments. We are also interested in the links and interactions between proof systems for programs and programming environments. Our research stands at the borderline between type theory, design and implementation of programming languages and programming by itself.

Caml and Caml Light, two implementations of a typed and powerful functional programming language belonging to the ML family, developed in our project, are used as context of study as well as programming tools and test bed for the the tools that we are developing.

We have scientific connections with other laboratories where related work is being achieved. In France, with the French Telecommunications Research Center (CNET, Lannion), the Computer Science Laboratory of École Normale Supérieure (LIENS, in Paris), with École des Mines in Paris and LAAS in Toulouse. The Cristal Project is member of the CNRS Research Group on Programming (GDR). Our main international connections are with AT&T Bell Laboratories (Murray-Hill, USA), DEC-SRC (Palo-Alto, USA), Carnegie-Mellon University (Pittsburgh, USA) and the Asian Institute of Technology (Bangkok, Thailand).

Table des matières