Rapport INRIA 1994 — Programme 2

Conception et réalisation d'outils d'aide à la programmation

PROJET CROAP

3 mai 1995

PROJET CROAP

Conception et réalisation d'outils d'aide à la programmation

Localisation : *Sophia Antipolis*

Mots-clés : architecture client-serveur (9), architecture répartie (1), attribut sémantique (4), Centaur (1), ChLoÉ (4), COQ (8, 9, 13), CRISTAL (16), démonstration automatique (13), environnement de programmation (1, 16–18), évaluation partielle (5), fiabilité du logiciel (9), génie logiciel (1, 9), grammaire attribuée (4), interface homme-machine (13), Lambda-Prolog (6), langage à objets (15), langage fonctionnel (15), langage naturel (13), langage parallèle (15), MALI (6), production de document (17), programmation (1), Prolog (1, 6), qualité du logiciel (9), sémantique (1), sémantique naturelle (1, 6, 8, 9), spécification (6), spécification formelle (6), syntaxe abstraite (6), théorie des types (8, 13), transformation de programme (9).

1 Composition de l'équipe

Responsable scientifique

Gilles Kahn, jusqu'à mars 1994
Isabelle Attali, Yves Bertot, de avril 1994 à octobre 1994
Yves Bertot, à partir d'octobre 1994

Responsable permanent

Thierry Despeyroux, jusqu'à octobre 1994

Secrétaire

Lydia Vergamini, jusqu'en septembre 1994
Sandrine Chevris, à partir d'octobre 1994

Personnel Inria

Isabelle Attali, chargé de recherche
Yves Bertot, chargé de recherche
Joëlle Despeyroux, chargé de recherche
Thierry Despeyroux, chargé de recherche
Gilles Kahn, directeur de recherche
Francis Montagnac, ingénieur
Valérie Pascual, chargé de recherche
Laurence Rideau, chargé de recherche
Laurent Théry, chargé de recherche, à partir du 1/10/94, précédemment post-doc, ATT Bell Labs, Murray Hill

Chercheurs invités

Konstantin Chebotar, Académie des Sciences de Moldavie, Kichiniev, 10/94

Ingénieurs experts

Janet Bertot
Ian Jacobs, jusqu'au 31/1/94

Chercheurs extérieurs

Arnaud Le Hors, Bull S.A.
Vincent Prunet, SEMA Group, jusqu'au 31/1/94

Collaborateurs extérieurs

Denis Caromel, maître de conférence, I3S-CNRS – UNSA
Anne-Marie Déry, maître de conférence, ESSI
André Hirschowitz, directeur de recherche, CNRS

Chercheurs post-doctorants

Penny Anderson, jusqu'au 18/10/94
Healfdene Goguen, depuis le 1/4/94

Chercheurs doctorants

Sylvan Dissoubray, boursier Inria
Sidi Ould Ehmety, boursier CIES
Ranan Fraer, boursier MRE, à partir du 1/10/94
Jean-Michel Léon, Bull S.A.
Renaud Marlet, SIMULOG, Toulouse

Delphine Terrasse, boursier MRE, ENPC

Stagiaires

Guillaume Courjaret, ESSI, 3ème année, du 1/1/94 au 31/3/94
Yann Coscoy, école polytechnique, du 1/4/94 au 30/6/94
Benoît Desaute, ESSI, 3ème année, du 1/1/94 au 31/3/94
Éric Eschenbrenner, DEA, UNSA, du 1/3/94 au 30/6/94
Ali Jaber, DEA, UNSA, du 1/12/93 au 30/4/94
Nicolas Trotignon, ENSAE, du 15/8/94 au 15/10/94

2 Présentation du projet

L'objectif du projet CROAP est d'étudier les outils nécessaires à la programmation de systèmes complexes et fiables et la coopération de ces outils au sein d'environnements de travail conviviaux et efficaces. Un certain nombre de ces outils peut être obtenu par une étude constructive des langages de programmation. L'environnements CENTAUR que nous développons fournit les moyens de cette étude.

Pour étudier les langages de programmation, nous utilisons notre propre méthode, dite *Sémantique Naturelle*, pour laquelle nous développons un laboratoire d'expérimentation très complet autour du formalisme TYPOL. Nous nous efforçons d'avoir des implantations réalistes pour ce formalisme grâce à l'utilisation de moteurs d'évaluation efficaces, développés dans d'autres équipes de recherche : PROLOG compilé, évaluateur d'attributs sémantiques moderne. Ces implantations nous permettent de décrire formellement des outils de programmation tels que des compilateurs ou des interprètes pour des langages de programmation très variés, puisque notre étude va des langages parallèles à objets aux langages parallèles à flux de données en passant par des langages de production de documents. Cette activité centrée sur l'exécution des définitions sémantiques s'accompagne naturellement de recherches sur les preuves associées à ces définitions formelles. Nous nous donnons donc les moyens de prouver formellement des propriétés des langages de programmation, de vérifier la correction de compilateurs, ou de prouver la validité de transformations sur des programmes. Ces activités de preuve sont directement reliées à notre objectif de fiabilité.

Un environnement de programmation est en soi un système complexe, qui doit fournir des outils d'édition de programme, de test, de mise au

point, de validation vis-à-vis de spécifications, d'optimisation, de maintenance. Les environnements de programmation peuvent inclure des outils développés indépendamment, par exemple des outils de preuve. Notre volonté de construire des environnements de programmation complets nous impose donc d'étudier les moyens d'intégrer des outils hétérogènes dans un système interactif. L'approche que nous avons choisie, qui consiste à faire coopérer les différents outils dans un système multi-processus, éventuellement réparti sur plusieurs machines et communiquant par des protocoles que nous mettons au point, nous permet également d'apporter une contribution au problème de la maîtrise de l'informatique répartie.

Les similitudes entre programmation et démonstration nous ont également conduit à étudier l'application des concepts développés pour les environnements de programmation aux environnements de développement de preuves en construisant des interfaces conviviales pour des systèmes de preuve présents dans le monde de la recherche, en particulier HOL et Coq.

Comme points marquants de l'année 1994 nous pouvons relever:

- La fin du projet Esprit GIPE II en janvier 1994.
- La mise à disposition de la version 2.0 de Centaur en septembre 1994.
- La définition d'une nouvelle méthode de description de la sémantique des languages de programmation. Cette méthode est à notre connaissance la première qui permette d'utiliser conjointement la syntaxe abstraite d'ordre supérieure et les types inductifs.

3 Actions de recherche

Les recherches effectuées dans le projet recouvrent un spectre assez large depuis la recherche fondamentale jusqu'à des applications pratiques.

3.1 Syntaxe, sémantique et preuves

Nous nous intéressons ici à la description précise des langages de programmation pour générer des outils adaptés à ces langages et pour raisonner formellement sur leurs propriétés. Dans nos travaux sur la *sémantique naturelle* et son langage d'implémentation, Typol, nous

voulons étudier les implantations efficaces de ce formalisme et élargir son champ d'application par une prise en compte des syntaxes d'ordre supérieur et par une mécanisation des preuves sémantiques.

3.1.1 Implantations optimisées du formalisme Typol

Participante : Isabelle Attali

Nous avons défini des conditions pour qu'une description sémantique en Typol puisse être évaluée par une grammaire attribuée. Ces vérifications sont utiles pour la mise au point et la détection d'erreurs, même si la stratégie envisagée pour l'évaluation repose sur Prolog. En collaboration avec Didier Parigot de l'équipe CHLOE, nous avons spécifié en Typol la traduction de Typol vers Olga, langage d'entrée de FNC-2. Des tests réalisés avec l'évaluateur ainsi obtenu ont montré des performances (en temps et en mémoire) tout à fait acceptables sur des exemples de taille réaliste [11].

Notre objectif est de générer des évaluateurs autonomes, indépendants de Centaur, mais pouvant éventuellement communiquer avec le système à l'aide des protocoles développés dans le projet (voir section 3.7) pour une évaluation interactive et incrémentale. Nous obtiendrons ainsi un véritable atelier de production d'évaluateurs et de compilateurs.

3.1.2 Évaluation partielle

Participants : Renaud Marlet, Gilles Kahn

L'évaluation partielle est un outil de programmation puissant et complexe, qui permet, par exemple, de systématiser l'optimisation des programmes. Nous proposons une formalisation de cette technique et une panoplie d'outils pour la construction d'évaluateurs partiels. Les résultats de ces travaux sont regroupés dans une thèse [2].

Notre étude est motivée par le constat de lacunes dans les fondements théoriques et dans les mises en œuvre pratiques : on dispose de peu d'outils pour déterminer les programmes qui sont à la fois équivalents et plus performants qu'un programme donné.

Afin de modéliser précisément la performance d'un programme, nous proposons tout d'abord un cadre général pour représenter le processus d'exécution (par opposition à la sémantique pure) et attribuer un coût

à une exécution. Cette présentation aboutit à une définition formelle de l'évaluation partielle en termes d'optimisation et met en évidence plusieurs notions d'évaluation partielle optimale.

Nous employons ces notions pour étudier formellement la correction des transformations de pliage/dépliage et l'algorithme de redémarrage généralisé. En particulier, nous établissons quelques lemmes de commutativité sur les systèmes de réécriture et nous examinons quelques bons critères d'arrêt pour l'algorithme de redémarrage généralisé.

Nous concluons notre étude sur une analyse des limites de l'optimisation par évaluation partielle.

3.1.3 Utilisation d'une syntaxe d'ordre supérieur dans les spécifications sémantiques

Participant : Thierry Despeyroux

La conception d'un nouveau TYPOL d'ordre supérieur fait apparaître la nécessité de concevoir un nouveau langage de types pour les arbres de syntaxe abstraite manipulés. Un formalisme de définition de types syntaxiques d'ordre supérieur a donc été défini. Ces définitions de types sont compilées en λ-PROLOG (dans sa version compilée PM développée par le projet MALI) et en PROLOG dans l'hypothèse où toute la puissance de l'unification supérieure n'est pas requise. Par ailleurs, ce nouveau formalisme de définition apporte, par rapport à METAL qui était utilisé précédemment, une facilité de définition modulaire de syntaxes abstraites.

3.1.4 Définition catégorique d'une syntaxe abstraite

Participants : Thierry Despeyroux, André Hirschowitz

Nous nous attachons à donner une définition catégorique de la notion de syntaxe abstraite, tant pour les syntaxes de premier ordre que pour celles d'ordre supérieur. Le but de ces travaux, outre celui de formaliser une notion maintenant largement utilisée, est d'une part de mettre en lumière les avantages de l'utilisation d'une syntaxe abstraite d'ordre supérieur pour la manipulation syntaxique de programmes, d'autre part de montrer l'adéquation des implémentations actuellement utilisées.

3.1.5 Syntaxe abstraite d'ordre supérieur et types inductifs

Participants : Joëlle Despeyroux, André Hirschowitz

À partir de la donnée d'un type inductif, le système Coq génère un principe de récurrence, et un principe d'induction, principes nécessaires pour faire la plupart des preuves en sémantique. La syntaxe abstraite d'ordre supérieur permet de formaliser les lieurs et les abstractions d'un langage fonctionnel, ainsi que les "sides conditions" et les changements de contextes dans les règles d'inférence. Le problème est qu'une syntaxe abstraite d'ordre supérieur, définie de manière usuelle, ne forme pas un type inductif.

Nous avons défini deux notions nouvelles: la *syntaxe abstraite d'ordre supérieur restreint* [1] et la *sémantique d'ordre supérieur* [9], qui permettent toutes deux d'utiliser à la fois la syntaxe d'ordre supérieur et les types inductifs.

La première notion considère les fonctions d'un langage L comme des fonctions de type $var \rightarrow L$, pour un certain ensemble var, sur lequel on ne met aucun axiome, hormis le fait qu'il est infini. Nous avons appelé ce type de syntaxe *syntaxe abstraite d'ordre supérieur restreint*. Une telle syntaxe décrivant trop de termes, il est nécessaire de définir les termes légaux, qui correspondent aux arbres de syntaxe abstraite.

En collaboration avec Amy Felty (Bell Labs, Etats Unis), nous avons décrit le système de types et les règles d'évaluation d'un lambda-calcul simplement typé, dans ce style. Nous avons prouvé l'adéquation de notre langage par rapport à une description du même langage en LF (Edinburgh Logical Framework). Enfin nous avons implémenté une preuve du théorème de 'subject reduction' dans Coq. La prochaine étape est de généraliser la preuve d'adéquation de la syntaxe à toute syntaxe d'ordre 2.

La seconde notion considère un terme ouvert, dépendant de variables, comme une fonction d'une liste de variables. Nous avons appelé ce nouveau style: *sémantique d'ordre supérieur*. Bien entendu, il faut ici aussi caractériser les termes légaux. Cette définition est plus naturelle, et plus concise que dans l'approche précédente. Par contre, les preuves sont plus

[1] Ce travail est décrit dans "Higher-order abstract syntax in Coq" de J. Despeyroux, A. Felty et A. Hirschowitz, article accepté à la conférence TLCA'95

délicates. Nous avons prouvé l'adéquation de la syntaxe d'un λ-calcul simplement typé, défini dans ce style, par rapport à une description du même calcul à l'ordre 1. Cette preuve a été réalisée dans le système Coq. Elle devrait pouvoir être généralisée à toute syntaxe d'ordre 2. Nous avons aussi prouvé l'adéquation de toute syntaxe d'ordre 2 définie dans ce style, par rapport à une description du même langage en LF. Ces travaux sont décrits dans [9].

Une troisième approche est étudiée par Joëlle Despeyroux en collaboration avec Frank Pfenning (Carnegie Mellon University, Etats-Unis). Il s'agit ici d'étendre le système LF, bien adapté à l'utilisation des syntaxes abstraites d'ordre supérieur, avec des types inductifs. Le système envisagé sera basé sur une extension de LF par des opérateurs de la logique modale. L'opérateur $\Box$ de la logique modale permet en effet de décomposer les fonctions de type récursives $A \Rightarrow B$ en $\Box A \rightarrow B$ où $\rightarrow$ représente les fonctions "paramétriques".

3.1.6 Traduction de Typol en Coq

Participantes : Delphine Terrasse, Joëlle Despeyroux

Les spécifications sémantiques écrites en Typol peuvent être traduites en Prolog ou en grammaires attribuées pour générer des outils exécutables, évaluateurs, compilateurs etc. Ces spécifications peuvent également être traduites dans des définitions pour un système de preuve comme Coq, ce qui permet de raisonner formellement sur les propriétés des langages et sur les outils générés.

Les années précédentes nous avons étudié cette traduction et avons dégagé deux méthodes possibles. La première méthode utilise peu la théorie des types et est facile à implémenter, alors que la deuxième permet d'utiliser plus efficacement le système Coq lors des preuves. En revanche, cette deuxième méthode n'était pas implémentable jusqu'alors, en raison de l'absence de types mutuellement inductifs dans le système Coq.

Cette année nous avons formalisé ces deux techniques de traduction et prouvé qu'elles sont correctes en montrant que ce sont des fonctions bijectives. Enfin, nous avons pu implémenter la deuxième méthode grâce aux types mutuellement récursifs nouvellement introduits dans Coq.

À terme, ce travail doit permettre de fournir un atelier pour l'étude formelle des propriétés de langages de programmation.

3.1.7 Propriétés syntaxiques de la théorie des types

Participant : Healfdene Goguen

Dans sa thèse à l'université d'Edimbourg, dont la rédaction s'est achevée au sein de notre équipe[1], Healfdene Goguen a introduit une nouvelle technique d'étude des langages typés basée sur une sémantique opérationnelle typée[2]. Cette technique fournit des méthodes plus élégantes que les méthodes traditionnelles pour l'étude des langages typés et est notablement plus efficace dès que l'on étudie la règle de réduction η. En intégrant cette technique parmi nos compétences, nous avons un nouvel outil pour aborder des problèmes difficiles, comme la preuve de la propriété Church-Rosser du calcul UTT (*unifying theory of dependent types*) que nous sommes les premiers à établir. Cette technique est également adaptée à une implémentation en Typol.

Dans l'avenir, cette nouvelle technique devrait faciliter notre étude des langages de programmation typés.

3.1.8 Transformation de programmes et validation

Participants : Ranan Fraer, Yves Bertot

Nous avons continué l'étude de la cohérence des spécifications sémantiques associées à un petit langage impératif en utilisant les outils de traduction de Typol vers Coq (cf. 3.1.6) et CtCoq (cf. 3.4.1). Les spécifications envisagées sont deux formes de sémantique dynamique (sémantique naturelle et sémantique opérationnelle structurelle "petits pas"), des spécifications de typage et des spécifications de transformations globales non-triviales (propagation de constantes).

Ces expériences [16] visent la construction d'environnements de programmation contenant des outils de manipulation dont la validité est garantie.

3.2 Construction de systèmes hétérogènes

Pour concevoir des environnements de programmation complets, il est nécessaire d'aborder la coopération des outils qui les composent et

[2]Ce travail est décrit dans "Typed Operational Semantics" de H. Goguen, article accepté à la conférence TLCA'95

leur interaction avec l'utilisateur humain. Pour faciliter la coopération inter-outils, nous développons une panoplie d'outils de connexion et de communication : SOPHTALK. Ce travail s'accompagne d'une recherche plus fondamentale sur la forme que devrait avoir un langage adapté à l'intégration d'outils.

3.2.1 Sophtalk

Participants : Janet Bertot, Ian Jacobs, Francis Montagnac

SOPHTALK est un système autonome permettant l'intégration de composants logiciels appartenant ou non à un même processus grâce à une architecture asynchrone dans laquelle les différents composants communiquent par envoi de messages sur un réseau logiciel. De nouvelles primitives ont été ajoutées au système pour améliorer les possibilités de paramétrisation et l'intégration dans un environnement fonctionnel.

3.2.2 Expression de relations réactives synchrones entre objets

Participants : Sylvan Dissoubray, Gilles Kahn

Notre but est de définir un langage pour spécifier le contrôle et l'intégration d'outils dans un environnement de programmation. Nous nous basons pour cela sur l'expérience acquise en utilisant SOPHTALK et le langage ESTEREL. Suite aux travaux réalisés l'an dernier sur l'ajout d'une instruction de reconfiguration au langage ESTEREL, nous avons conçu une interface objet adaptée au cas où les reconfigurations interviennent. Cette interface objet consiste en un mécanisme pour référencer les nouveaux objets ajoutés au système par reconfiguration. La particularité de ce mécanisme est de réaliser les appels de méthodes sur les objets en respectant l'ordre partiel induit par l'analyse causale du programme ESTEREL, et non dans un ordre arbitraire. Ceci nous donne une machine d'exécution "causale", partiellement déterministe du point de vue des objets de l'environnement. Cette approche reste compatible avec l'exigence de déterminisme car il est possible de compléter l'ordre partiel obtenu en un ordre total en ajoutant des sous-modules qui expriment des dépendances causales, sans modifier la structure modulaire du programme.

L'avantage de cette approche est que l'ajout de nouveaux objets est plus modulaire car on évite des problèmes de partage de variables. De plus, la causalité exprime directement les dépendances significatives du problème sous forme d'un ordre partiel.

Ces travaux font l'objet d'une thèse dont la rédaction est en cours et dont la soutenance est envisagée début 1995.

3.3 Outils interactifs

La mise en œuvre pratique des outils générés à partir de définitions formelles de langages nous impose de fournir également des outils d'interaction avec l'utilisateur. Ces outils ont la particularité originale de manipuler principalement des objets structurés.

3.3.1 Édition syntaxique guidée par menus

Participante : Valérie Pascual

CENTAUR offre plusieurs possibilités d'édition complémentaires : édition textuelle (en utilisant un éditeur en ligne, un éditeur externe ou des possibilités de copier/coller X) ou édition syntaxique guidée par menus. Deux améliorations de l'édition syntaxique ont été réalisées, pour la rendre plus efficace et plus agréable à utiliser : la possibilité de définir et d'utiliser des menus hiérarchiques et la possibilité d'utiliser le clavier pour choisir une entrée dans un menu.

Un menu hiérarchique contient à la fois des entrées correspondant à des schémas ou à des règles de transformation et des entrées correspondant à des sous-menus. Le langage de spécification de menus, TransForm, a été étendu afin de permettre la définition de sous-menus.

Le mécanisme d'accélérateur clavier permet de choisir une entrée dans un menu en tapant un préfixe non ambigu. Ce mécanisme, combiné à l'édition en place, rend l'édition de programme sous CENTAUR plus rapide et agréable.

Des expériences ont été faites avec les panneaux (voir section 3.3.2) pour regrouper une fenêtre d'édition, son menu et son "presse-papier" associés. Ces panneaux ont le grand avantage de distinguer les outils d'édition (menu, clipboard) associés à chaque fenêtres d'édition.

3.3.2 Objets graphiques

Participante : Janet Bertot

Les objets graphiques de CENTAUR ont été enrichis de panneaux qui permettent une meilleure présentation pour les environnements complexes. Des primitives de modifications dynamiques de ces panneaux sont également fournies. Ces nouveaux panneaux trouvent des applications fréquentes dans nos outils (voir les sections 3.3.1 et 3.4.1).

3.3.3 Formateur incrémental graphique FIGUE

Participante : Valérie Pascual

Le système FIGUE (Formateur Incrémental GraphiUE), initialement développé par Laurent Hascoët, est un système de formatage d'objets structurés bi-dimensionnel, au style de formatage paramétrable et incrémental. Ce système est essentiel pour les applications mathématiques de CENTAUR.

Nous envisageons d'utiliser ILOGVIEWS pour réaliser l'affichage des arbres d'objets graphiques utilisés par FIGUE, afin de dissocier formatage et affichage. Ce découplage doit permettre de faciliter la réutilisation de cet outil de formatage par des applications distinctes de CENTAUR.

3.3.4 Un serveur de graphe

Participants : Arnaud Le Hors, Jean-Michel Léon

Nous développons un outil d'affichage et d'édition de graphes qui peut servir à visualiser des données obtenues lors d'analyses de programmes. Cet outil est en fait conçu comme un serveur indépendant, communiquant avec le reste du système par un protocole `rpc`.

Ce serveur d'affichage et d'édition de graphes est en cours de réécriture. Toujours en C++, la nouvelle version est construite à l'aide de la librairie graphique IlogViews qui remplace avantageusement la InterViews utilisée jusque là. En effet, après une évaluation approfondie d'IlogViews nous pensons que cette nouvelle boîte à outils, tout en ayant les mêmes fondements, fournit un niveau de fonctionnalités supérieur ainsi qu'une meilleure intégration au système de fenêtrage X. Ceci, associé à l'aspect commercial de ce produit qui implique un certain support, permettra

d'étendre les fonctionnalités du serveur de graphes tout en augmentant sa stabilité et sa longévité.

3.4 Conception d'interfaces pour les systèmes de calcul symboliques

L'essor des méthodes formelles en programmation implique l'intégration d'outils de preuves dans les environnements de programmation et nous étudions ce problème en utilisant CENTAUR pour développer des environnements de travail adaptés à des systèmes de preuves variés, avec un accent particulier sur une interface graphique pour le système de preuve COQ ainsi que pour HOL. Nous abordons aussi les problèmes de connexion avec les systèmes de calcul formel.

3.4.1 Interface au système de preuve Coq

Participants : Janet Bertot, Yves Bertot

L'adaptation de CTCOQ aux versions successives du système COQ pose des problèmes scientifiques nouveaux, dûs à l'apparition de techniques et d'outils novateurs dans la dernière version. Nous utilisons désormais l'analyseur syntaxique interne au système de preuve plutôt qu'un analyseur généré par CENTAUR. Pour cela un outil de traduction d'arbres dirigé par des tables a été mis au point et l'analyseur syntaxique inclus dans COQ a été isolé. Le dernière version de Coq fournit également une possibilité de retour-arrière hiérarchique meilleur que le retour-arrière historique fourni par la version précédente. Nous avons étudié les moyens d'étendre le prototype d'interface CTCOQ pour utiliser cette fonctionnalité.

Nous avons mené une étude sur une nouvelle conception de l'algorithme de preuve par sélection dont le but est de permettre un guidage des tâche de preuve à l'aide de la souris. Le nouvel algorithme, construit sur des tables de règles de comportement, permet à l'utilisateur d'ajouter de nouvelles règles afin d'adapter l'algorithme au domaine mathématique abordé.

3.4.2 Présentation textuelle de preuves

Participants : Yann Coscoy, Gilles Kahn

Ce travail a pour objectif de produire automatiquement des textes de preuves mathématiques, dans un langage compréhensible par un ingénieur non spécialiste des outils de preuves formelles. Les preuves considérées sont celles effectuées à l'aide du système Coq. Ce système représente les démonstrations à l'aide de λ-termes typés extraordinairement riches d'information, mais obscurs. Nous cherchons à présenter ces termes en langage naturel.

Initialisé l'an dernier lors du DEA d'Anne Massol, ce travail a été repris par Yann Coscoy [13] sous la direction de Gilles Kahn et a fait des progrès en généralité, optimisation et en qualité d'implémentation. La généralité de l'outil a été améliorée et il s'adapte maintenant à tous les symboles définis inductivement par l'utilisateur. En particulier, les preuves par récurrence sont maintenant compréhensibles. Le texte produit est mieux optimisé car un grand nombre de références inutiles pour le raisonnement sont détectées et éliminées. Enfin, le code a été entièrement refondu pour être plus facilement extensible.

Un article a été rédigé, en collaboration avec Laurent Théry, alors à Bell Labs[3]. Une version de ce logiciel est distribuée avec le système CtCoq.

Pour continuer ce travail, nous devons passer à la nouvelle version de Coq où la structure des preuves a changé et doit permettre de nouveaux développements comme la génération incrémentale du texte au fur et à mesure qu'une preuve est construite. Nous devons également améliorer la qualité et la densité du texte produit, en tenant mieux compte des différents styles de preuves, comme les preuves par réécriture. Ce travail peut nécessiter une interaction avec des linguistes (F. Pereira, A. Ranta).

Dans le cadre du projet GENIE, des preuves de taille significative seront construites avec le système Coq, constituant ainsi un matériau privilégié d'expérimentation. Un résultat solide sur la notion d'audit de preuves formelles n'est cependant pas à attendre avant 1996.

[3] "Extracting Text from Proof" de Yann Coscoy, Gilles Kahn et Laurent Thery, article accepté à la conférence TLCA'95

3.4.3 Interface graphique pour le système Elf

Participante : Penny Anderson

En collaboration avec Frank Pfenning (Carnegie Mellon University, États-Unis) nous avons initialisé la construction d'un environnement CENTAUR pour le système ELF qui peut servir également comme un système d'assistance à la preuve.

3.4.4 Utilisation de GAIA pour la génération d'exemples

Participante : Laurence Rideau

Dans le but de construire des jeux de tests pour nos éditeurs syntaxiques et nos décompilateurs, nous avons fait une expérience d'utilisation du système GAIA, développé en MAPLE dans le projet EURECA, qui permet la génération aléatoire d'objects structurés. Nous utilisons donc GAIA dans une connexion client-serveur pour générer des arbres et les transférer vers CENTAUR.

Cette expérience a montré que les arbres syntaxiques générés par GAIA ne ressemblent pas du tout à de *vrais* programmes : les *vrais* programmes sont larges et peu profonds alors que les arbres produits par GAIA sont étroits et profonds. En collaboration avec les développeurs de GAIA, il faut trouver le moyen de paramétrer ce système afin qu'il produise des arbres plus proches de la réalité.

3.5 Étude formelle de formalismes

Nous étudions des langages de programmation choisis pour l'aspect porteur des paradigmes qu'ils mettent en jeu. L'étude formelle des langages contenant du parallélisme nous permet d'adapter nos compétences à l'évolution du matériel informatique vers des machines parallèles.

3.5.1 Sémantique d'un langage à objets parallèle

Participants : Isabelle Attali, Denis Caromel, Sidi Ould Ehmety

Nous souhaitons étendre la définition de la sémantique dynamique du langage EIFFEL, au langage EIFFEL//, défini par Denis Caromel. Sidi Ould Ehmety, au cours de sa première année de thèse, a étudié les différents formalismes permettant de modéliser des systèmes concurrents

et a choisi une sémantique de type SOS "little step" pour définir la sémantique des systèmes EIFFEL//.

Nos objectifs sont de détecter automatiquement le parallélisme latent de systèmes à objets séquentiels (EIFFEL) et de définir et prouver des transformations formelles de systèmes EIFFEL séquentiels en systèmes EIFFEL//.

3.5.2 Sémantique d'un langage fonctionnel parallèle

Participants : Isabelle Attali, Denis Caromel, Ali Jaber

Le langage SISAL 2.0, défini conjointement par les équipes de recherche de Colorado State University et Lawrence Livermore National Laboratory, est un langage fonctionnel, ayant, entre autres objectifs, une bonne adéquation aux architectures "dataflows". L'étude du langage à travers son manuel de référence nous a montré un manque de rigueur, des ambiguïtés, voire des contradictions dans la sémantique informelle du langage décrite en anglais. Après de nombreux échanges de vue avec les auteurs du langage, nous avons défini formellement la sémantique dynamique de SISAL en TYPOL [10].

3.6 Environnements de programmation et de conception

Le développement d'environnements de programmation pour des langages en vraie grandeur nous permet de vérifier l'applicabilité de nos méthodes à des problèmes pratiques. Au même titre que le travail sur les environnements de calcul symbolique, ce domaine de recherche motive notre recherche sur l'intégration d'outils hétérogènes, puisque les environnements que nous développons doivent coopérer avec des outils déjà existant sur le marché pour les langages considérés.

3.6.1 Environnement de programmation pour ML

Participants : Laurence Rideau, Laurent Théry

Le développement d'un environnement de programmation CENTAUR pour le langage SML/NJ (développé par ATT Bell Laboratories) a continué cette année en collaboration avec Laurent Théry alors aux Bell Labs. En parallèle, nous avons repris notre collaboration avec le projet CRISTAL pour le langage CAML.

Cette année nous nous sommes interessés plus particulièrement à la vérification de types dans les programmes SML : l'utilisateur a accès au type de n'importe quelle sous-expression au moyen de l'environnement graphique de CENTAUR. Pour implémenter ces facilités pour l'environnement SML/NJ, nous avons dû modifier le vérificateur de type de SML, mal adapté à cette utilisation interactive. À terme nous pensons utiliser la vérification de types au cours de la construction [20] des programmes, avec une édition syntaxique guidée par les types.

3.6.2 Environnement de programmation pour C

Participants : Guillaume Courjaret, Benoît Desaute, Thierry Despeyroux

L'éditeur syntaxique développé précédemment par Thierry Despeyroux a été enrichi par l'utilisation de CENTAUR comme interface au compilateur GCC. Les messages d'erreur émis par GCC sont affichés dans une fenêtre de CENTAUR qui permet la liaison hypertext entre messages et occurrences. Les deux difficultés principales de ce travail étaient d'une part de prendre en compte la possibilité offerte par C d'inclusion de fichiers (pour lesquels il faut offrir la possibilité de pointage hypertext en cas d'erreur), et d'autre part de permettre l'interprétation précise, sous forme d'occurrence dans un arbre, des messages d'erreur émis par GCC qui utilisent une notion de numéro de ligne.

3.6.3 Analyseur hors contexte pour C++

Participants : Nicolas Trotignon, Thierry Despeyroux

De même que cela a été fait l'an passé pour C, la construction d'un environnement pour C++ a été initiée. Une partie d'un analyseur syntaxique hors contexte pour C++ a été réalisée.

3.6.4 Manipulation de documents structurés

Participante : Isabelle Attali

La collaboration avec Dennis Arnon (XEROX PARC) et Paul Franchi-Zannettacci (Université de Nice) sur un environnement de programmation CENTAUR dédié à la manipulation de documents s'est poursuivie

(définition et génération de nouveaux outils sémantiques). Ces outils manipulent un ou plusieurs documents pour en faire des synthèses, des extractions, des vérifications, et plus généralement tout outil fondé sur une analyse dépendant du contexte. Ces travaux font l'objet d'une publication commune dans "Mathematical and Computer Modelling Journal" [3].

3.6.5 Environnement générique pour la synthèse architecturale

Participantes : Valérie Pascual, Laurence Rideau

Notre participation au projet ASAR [7, 6] s'organise autour de 3 axes : transformations de programmes, visualisation de graphes, construction d'un atelier hétérogène.

Pour les transformations de programmes, nous expérimentons d'une part avec l'interface homme-machine pour déclencher les transformations, construire des menus proposant les transformations valables et modifier dynamiquement ces menus. D'autre part nous étudions la connexion avec des processus externes pour effectuer certaines transformations spécifiques.

La visualisation de graphes utilise des algorithmes de "layout" externes qui peuvent être développés indépendamment de CENTAUR tel celui-ci développé par ailleurs dans le projet.

Enfin, nous intervenons dans la création d'un Atelier Hétérogène par les tâches suivantes: (1) assurer que les composants peuvent fonctionner seuls ou dans l'environnement ; (2) définir un protocole pour accueillir des outils du commerce dans l'environnement (par exemple SPEEDCHART) ; (3) assurer l'intégration de l'ensemble des logiciels développés dans le projet.

3.7 Centaur 2.0

Participants : Isabelle Attali, Janet Bertot, Thierry Despeyroux, Ian Jacobs, Francis Montagnac, Valérie Pascual, Laurence Rideau, Laurent Théry

Le groupe a distribué en septembre une nouvelle version de CENTAUR, la version 2.0, qui intègre de nombreux développements basés sur des recherches effectuées dans le cadre du projet Esprit GIPE II.

Notons en particulier des outils divers d'édition textuelle et structurée, des librairies de gestion de messages d'erreur et de génération d'environnements, des protocoles de transfert d'arbre, une nouvelle implémentation de TYPOL basée sur un processus PROLOG externe et une nouvelle batterie d'outils d'analyse statique sur les programmes TYPOL.

4 Actions industrielles

4.1 Projet Esprit GIPE II

La revue finale du projet Esprit GIPE II, dont l'Inria était le contractant principal pour les deux dernières années avec Janet Bertot comme directeur et Laurence Rideau comme site-leader, s'est tenue le 18 janvier 1994 au CWI à Amsterdam. Ce projet ESPRIT regroupait l'Inria, Bull S.A., CWI/University of Amsterdam, Gipsi S.A., PELAB (Linkoping University, Sweden), Planet S.A. (Grèce), PTT Research (Pays-Bas), SEMA Group et TUD (Technical University of Darmstadt).

4.2 Projet de coopération GENIE

Nous participons au projet de coopération entre Dassault Aviation, l'Inria et certaines de ses filiales intitulé *Sciences de l'information et ingéniérie concourante*. Les thèmes qui concernent notre projet sont la conception de logiciels critiques et la réingénierie. Nous intervenons avec le prototype CTCOQ (voir section 3.4.1).

4.3 Coopération avec SIMULOG

Nous maintenons des relations constantes avec la société SIMULOG qui développe et commercialise des environnements de programmation dédiés au calcul scientifique utilisant CENTAUR comme base d'intégration.

5 Actions nationales et internationales

5.1 Actions nationales

Isabelle Attali est membre du comité de direction du PRC Programmation et anime le pôle "Interfaces et Environnements" qui comprend 4 projets nationaux. Elle est consultant extérieur pour le groupe de travail SCOOP du PRC Communication Homme Machine dont le thème est la mise en œuvre des "Systèmes Coopératifs" et qui réunit des chercheurs de différents domaines (infrastructures matérielles et logicielles, aspects réseaux, métaphores d'interaction, facteurs humains, etc.).

Dans le cadre de l'action *Développement d'outils de démonstration pour la vérification de programmes de type industriel* du PRC Programmation, nous collaborons avec les équipes COQ (INRIA-Rocquencourt, LIP, ENS-Lyon), Demons (LRI), Euréca, Protheo (CRIN), et REVEAL (LIFO) sur les aspects interfaces graphiques et intégration d'outils.

Les travaux autour de l'implémentation de TYPOL utilisant les grammaires attribuées sont menés en collaboration avec les chercheurs de l'équipe CHLOE et s'inscrivent dans le cadre d'un projet soutenu par le PRC Programmation sur le thème "Intégration des systèmes Centaur et FNC-2".

Les travaux sur la sémantique de EIFFEL// s'inscrivent dans un projet commun avec l'équipe de Patrick Sallé (IRIT, Toulouse) sur le thème "Outils pour la Programmation en Langages d'Acteurs" soutenu par le PRC Programmation.

Dans le cadre des projets inter-PRC, nous participons au projet ASAR (*Atelier d'accueil générique pour la Synthèse ARchitecturale*). L'objectif du projet est de mettre en place en utilisant CENTAUR un environnement générique pour la CAO d'architectures. Nos partenaires sont : LIVE (Evry), IRISA (Rennes), CERT (Toulouse), LASTI (Lannion), I3S (Nice).

5.2 Actions internationales

Depuis le 1er juin 1994, Joëlle Despeyroux est *site leader* du projet Bra *Types for Proofs and Programs*. Gilles Kahn en est le directeur pour l'année universitaire 94-95.

Yves Bertot a rendu visite à l'université de Nimègue en Mars 1994 pour y présenter le prototype d'interface CtCoq et pour des discussions sur la présentation textuelle des preuves.

La définition de Sisal a donné lieu à une collaboration avec Andrew Wendelborn (University of Adelaide, Australie) et Jean-Luc Gaudiot (University of Southern California) pour les aspects concernant la gestion de tableaux dynamiques spécifiques au langage.

Gilles Kahn a visité les laboratoires d'ATT Bell Laboratories aux États-Unis pour une semaine.

Notre projet est associé, en collaboration avec le projet MEIJE, au réseau EuroFOCS (Fondations of Computer Science) soutenu par le programme Capital Humain et Mobilité.

6 Diffusion des résultats

6.1 Enseignement

Plusieurs membres de l'équipe participent aux cours du DEA informatique de l'université de Nice, de la 3ième année de l'ESSI (Ecole Supérieure en Sciences Informatiques) et du CERICS.

Par ailleurs nous participons à l'École Jeunes Chercheurs du PRC Programmation et collaborons avec le CNAM.

6.2 Participation à des conférences et colloques

Des membres du projets ont participé aux sessions de démonstration de la conférence STACS (Caen, février 1994), au colloque HISC (*Human Interaction for Symbolic Computation*, Amsterdam, mars 1994), à la conférence TACS (*Theoretical Aspects of Computer Science*, Sendai, Japon, avril 1994), au séminaire annuel Bra *Types for Proofs and Programs* (Baastad, Suède, juin 1994), à la conférence LICS (*Logic In Computer Science*, Paris, juillet 1994), à la conférence LPAR (*Logic Programming and Automated Reasoning* Kiev, Ukraine, juillet 1994), aux Journées du PRC Programmation (Lille, septembre 94), au colloque *Third International workshops on Hardware/Software Codesign* (CODE/CASHE, Grenoble, septembre 1994), à la conférence HOL (Malte, septembre 1994).

6.3 Organisation de colloques et de cours

Isabelle Attali a organisé la journée du pôle "Interfaces et Environnements" du GDR Programmation (Bordeaux, juin 1994).

6.4 Diffusion de produits

Une nouvelle version de SOPHTALK est disponible sur un serveur `ftp`.

La version 2.0 de CENTAUR ainsi que sa documentation est distribuée depuis septembre 1994 grâce à un serveur ftp accessible uniquement aux possesseurs d'une licence utilisateur. À ce jour 97 sites à travers le monde sont titulaires d'une licence CENTAUR.

Une nouvelle version du prototypes CTCOQ (interface pour le système COQ) a été distribuée en juillet 1994. Elle est disponible, ainsi que CHOL (interface pour le système HOL), sur un serveur `ftp` anonyme. Des pages de présentation `WWW` sont également mises en accès sur le réseau.

7 Publications

Thèses

[1] H. GOGUEN, *A Typed Operational Semantics for Type Theory*, thèse de doctorat, University of Edinburgh, août 1994.

[2] R. MARLET, *Vers une formalisation de l'évaluation partielle*, thèse de doctorat, Université de Nice - Sophia Antipolis, décembre 1994.

Articles et chapitres de livre

[3] D. ARNON, I. ATTALI, P. FRANCHI-ZANNETTACCI, «A Document Manipulation System based on Natural Semantics», *Mathematical and Computer Modelling Journal*, 1994, parution prévue fin 94.

Communications à des congrès, colloques, etc.

[4] P. ANDERSON, «Program Extraction in a Logical Framework Setting», *in : Proceedings of the 5th International Conference on Logic Programming and Automated Reasoning*, juillet 1994. Paru aussi en rapport de recherche Inria RR-2261.

[5] P. ANDERSON, «Representing Proof Transformations for Program Optimization», *in : Proceedings of the 12th International Conference on Automated Deduction*, Springer-Verlag, juin 1994. Paru aussi en rapport de recherche Inria RR-2229.

[6] P. ASAR, «Framework and Multi-Formalism: the ASAR Project», *in : Proceedings of the Fourth International IFIP 10.5 Working Conference on Electronic Design Automation Frameworks*, Gramado, Brésil, novembre 1994. (P. Asar est un nom générique pour le projet ASAR; dont L. Rideau).

[7] P. ASAR, «Towards a Multi-Formalism Framework for Architectural Synthesis: the ASAR Project», *in : Proceedings of the Third International Workshop on Hardware/Software Codesign (Codes/CASHE'94)*, Grenoble, septembre 1994. (P. Asar est un nom générique pour le projet ASAR; dont L. Rideau).

[8] Y. BERTOT, G. KAHN, L. THÉRY, «Proof by Pointing», *in : Theoretical Aspects of Computer Software*, 1994. Springer-Verlag LNCS 789.

[9] J. DESPEYROUX, A. HIRSCHOWITZ, «Higher-Order Syntax and Induction in Coq», *in : Actes de la 5ième conf. int. sur 'Logic Programming and Automated Reasoning' (LPAR 94), Kiev, Ukraine, 16–21 Juillet*, 1994. LNCS 822. Paru aussi en rapport Inria RR-2292, juin 1994.

Rapports de recherche et publications internes

[10] I. ATTALI, D. CAROMEL, A. WENDELBORN, «A Formal Dynamic Semantics of Sisal», *rapport de recherche n°RR94-49*, I3S, octobre 1994.

[11] I. ATTALI, D. PARIGOT, «Integrating Natural Semantics and Attribute Grammars: the Minotaur System», *rapport de recherche n°RR2339*, Inria, septembre 1994.

[12] *The Centaur 2.0 manual*, août 1994, I. Jacobs, réd., approx. 1000 pages.

[13] Y. COSCOY, *Traduction de preuves en langage naturel pour le système COQ*, Rapport de stage d'option, École Polytechnique, juillet 1994.

[14] A.-M. DÉRY, L. RIDEAU, «Distributed programming environments: an example of a message protocol», *rapport technique n°165*, Inria, août 1994.

[15] E. ESCHENBRENNER, *Preuve de Church-Rosser en sémantique d'ordre supérieur*, Rapport de stage de DEA, Université de Nice - Sophia Antipolis, juin 1994.

[16] R. FRAER, *Preuves en sémantique et preuves de programmes*, Rapport de stage de DEA, ENS Lyon, juin 1994.

Divers

[17] Y. BERTOT, R. FRAER, «Reasoning with Executable Specifications», septembre 1994, in proceedings des Journées du GDR Programmation, Lille.

[18] «proceedings of BRA technical workshop "On proving properties of programming languages", Sophia Antipolis, 20-21 septembre 1993», Inria, avril 1994, J. Despeyroux, réd.

[19] «Gipe II Sixth Review Report, Esprit project 2177», Inria, janvier 1994, J. Bertot, réd.

[20] L. RIDEAU, L. THÉRY, «Building Programming Environments for ML, a first step», septembre 1994, in proceedings des Journées du GDR Programmation, Lille.

[21] D. TERRASSE, «Encoding Natural Semantics in Coq», septembre 1994, in proceedings des journées du GDR de programmation, Lille.

[22] D. TERRASSE, «Translation From Typol to Coq», avril 1994, in proceedings of BRA technical workshop "On proving properties of programming languages", Inria, J. Despeyroux, réd.

8 Abstract

The aim of the CROAP project is to study what tools are necessary to program secure and complex systems and how to make them cooperate in user-friendly and efficient working environments. A large number of these tools can be obtained through a study of programming languages and, more generally, formal languages. We implement the programming environment generator CENTAUR, which provides the means for studying languages.

For programming languages semantics, we use a method called *Natural Semantics*, implemented by the specification language TYPOL. This language can be executed fairly efficiently, which makes it possible to extract practical tools from programming language specifications. We also provide ways to reason formally about programming language properties and about the validity of the extracted tools.

A programming environment is itself a complex system that must provide a wide range of tools concerned with the tasks of editing, testing, debugging, validating, etc. To be complete, programming environments may have to include tools developed independently. Thus, we must

also study the means of integrating heterogeneous tools in an interactive system. We developed the necessary tools and methodology to have several tools communicate in a multi-process system, possibly from several machines connected via a network.

The similarities between the activities of proving and programming have led us to study the application of our results to working environments for proof systems. The environments we obtain provide a notable added value to proof systems, with new ways of communicating between the user and the machine. This work gives us new insights in the domain of programming environments.

Table des matières