Part I

Présentation de l'unité

Chapter 1

Équipes de recherche

1.1 Sémantique et Interprétation Abstraite

- Responsable : Patrick Cousot (Professeur à lÉcole Normale Supérieure & Université de Paris 9) ;
- Chercheurs :

  Éric Goubault (Ingénieur des mines détaché au CNRS),

  Bruno Monsuez (Chercheur contractuel Esprit BRA 8130 LOMAPS) ;

– Doctorants : Régis Cridlig, Ingénieur de l'armement-recherche,

Laurent Mauborgne, AMN, École Polytechnique,

Franck Védrine, Élève de l'École Normale Supérieure ;

- Invités :

  Christopher Colby (CMU, mai-août 1995),

  Pierpaolo Degano (Dipartimento di Informatica, Università di Pisa, janvier 1997),

  Peter Lee (CMU, mai 1995),

  Leslie Lamport (DEC/research/SRC, septembre 1995).

Part II

Activité scientifique des équipes

Chapter 2

Sémantique et Interprétation Abstraite

L'équipe s'est constituée en octobre 1991 à l'occasion du recrutement comme professeur de Patrick Cousot. L'axe principal de recherche de l'équipe est l'*interprétation abstraite* [CC77, Cou96a] dans ses aspects théoriques concernant la conception et l'approximation de sémantiques [Cou97c] et ses aspects appliqués concernant l'analyse sémantique et la vérification de programmes [Cou96g]. Au travers d'un séminaire hebdomadaire, le groupe conserve des liens privilégiés avec l'équipe ¡¡Sémantique, preuve et interprétation abstraite¿¿ du LIX, que Patrick Cousot a créée et dirigée jusqu'au 1er octobre 1991.

2.1 Introduction

Les buts de recherche dans cette équipe sont de développer les modèles de sémantiques, les fondements de l'interprétation abstraite et l'analyse sémantique des programmes.

Les modèles de sémantiques : ce travail concerne principalement les méthodes de présentation des sémantiques, l'étude des relations entre sémantiques permettant de construire des hiérarchies de descriptions obtenues par interprétation abstraite et les modèles du parallélisme permettant une distinction fine entre vrai parallélisme et non-déterminisme.

Les fondements de l'interprétation abstraite : il s'agit d'étudier les diverses modélisations de la notion d'approximation discrète en sémantique. Les applications concernent la sémantique (par exemple le non-déterminisme peut être compris comme une approximation du parallélisme) et l'analyse de pro-

gramme (par exemple l'analyse de nécessité pour les langages fonctionnels paresseux est traditionnelement comprise comme une approximation de leur sémantique dénotationnelle).

Les méthodes d'analyse et de manipulation de programmes par interprétation abstraite de sémantiques : il s'agit de spécifier et de construire des analyseurs sémantiques pour la détermination automatique, statique et correcte de propriétés dynamiques des programmes. L'analyse sémantique de programmes peut s'organiser en deux parties relativement indépendantes :

- La première concerne la représentation approchée des propriétés des objets manipulés par le programme, ce qui se formalise par des domaines abstraits relativement indépendants du langage de programmation (généralement par des ensembles partiellement ordonnés dont la relation d'ordre formalise l'implication) et des opérations abstraites (correspondant aux opérations élémentaires sur les objets manipulés par le programme) ;
- La deuxième partie concerne la représentation approchée des calculs effectués par le programme, ce qui se formalise de manière relativement dépendante du langage de programmation par des sémantiques abstraites (généralement sous forme d'équations de point fixe dont la résolution formalise l'inférence de propriétés de programmes).

Cette dichotomie sert de base à la présentation de notre travail. Les applications, qui concernent la mise au point symbolique de programmes, la compilation, l'évaluation partielle, la transformation de programmes comme la parallélisation, les preuves (génération automatique d'invariants, preuves de terminaison), la vérification automatique de modèles (*model-checking*) etc., sont évidemment liées aux deux aspects. Les efforts ont porté principalement sur les langages fonctionnels (analyse de nécessité, des temps de liaison, typage), les langages logiques, l'analyse de programmes parallèles et plus récemment l'analyse des langages objets.

2.2 Sémantiques

2.2.1 Présentation de sémantiques et construction de hiérarchies de sémantiques par interprétation abstraite

Si toutes les méthodes de définition de sémantiques des langages de programmation procèdent de manière compositionnelle par induction sur la syntaxe, elles diffèrent généralement dans leur style de présentation, que ce soit par point fixe, sous forme équationnelle, de contraintes, de conditions de fermeture, par un système formel à base de règles ou par un jeu. Nous avons étendu tous ces modes de présentation aux comportements infinis [Cou95e, Cou97c], ce

qui permet alors de montrer l'équivalence de ces divers modes de présentation [CC95a]. Ces présentations de sémantiques étant conservées par interprétation abstraite [CC95a], il devient possible de comprendre toutes les sémantiques d'un langage de programmation comme des approximations les unes des autres. Ceci permet de construire des hiérarchies de sémantiques [Cou95e, Cou97c] qui diffèrent seulement par la précision de la description de l'exécution et le mode de présentation. Par exemple [Cou95e] une sémantique opérationnelle du parallélisme peut s'approcher par le non-déterminisme. On obtient une sémantique de traces finies ou infinies. Ignorant les états intermédiaires, on obtient une sémantique dénotationnelle. Ignorant la non-terminaison, on obtient une sémantique naturelle. Considérant des ensembles d'états, on obtient les transformateurs de prédicats. Décrivant ces ensembles par des langages formels, on obtient des sémantiques axiomatiques sur lesquelles sont basées les méthodes de preuve [Cou97c]. Il s'agit à terme de trouver un modèle de sémantique et une présentation indépendants d'un langage de programmation particulier, rôle que jouent les systèmes de transition dans le cas des sémantiques opérationnelles de petits pas.

2.2.2 Modèles du vrai parallélisme par automates de dimension supérieure

Si les modèles mathématiques du calcul sont bien connus pour les langages séquentiels, il n'en va pas de même dans le cas du parallélisme où il faut prendre en considération l'exécution simultanée d'actions concurrentes, synchrones ou asynchrones. C'est pourquoi nous étudions les modèles opérationnels du ¡¡ vrai ¿¿ parallélisme basés sur la notion d'automate de dimension supérieure issue des idées de Vaughan Pratt et Rob van Glabbeek (Stanford University). Ces modèles sont extrêmement flexibles puisqu'ils permettent de définir de façon dénotationnelle ou bien opérationnelle des langages parallèles aussi divers que Linda, Parallel Pascal (avec des variables partagées) ou Concurrent ML (communications via des messages). Par interprétation abstraite, ils permettent de construire effectivement un automate approché qui modélise les comportements dynamiques possibles du programme considéré, pour faire, par exemple, des analyses de systèmes infinis par model-checking abstrait.

Un automate de dimension supérieure est similaire aux automates classiques, dans le sens où l'on a toujours des états, et des transitions reliant les états. Pour exprimer le parallélisme, et même le vrai parallélisme (qui se distingue des traductions par entrelacements, identifiant non-déterminisme et parallélisme), on rajoute des transitions de dimension supérieure exhibant le degré d'asynchronie que l'on peut avoir dans l'exécution d'un programme. Ces transitions peuvent se représenter géométriquement, tout comme les transitions ¡¡ ordinaires ¿¿ (de dimension 1, car ce sont des segments) et les états (ou transitions de dimension 0, car ce sont des points). Par exemple, l'entrelacement $a.b + b.a$ (notation CCS) a pour représentation comme automate quatre transitions de dimension

1 formant le bord d'un carré. Remplir le carré consiste à rajouter une transition de dimension 2, et donc à indiquer l'indépendance des actions a et b. Les dimensions d'ordre supérieur décrivent donc le degré de parallélisme effectif lors de l'exécution d'opérations concurrentes. Cette information supplémentaire est importante pour effectuer certaines analyses de programmes comme l'exclusion mutuelle ou l'indépendance entre actions et valider des propriétés liées à la géométrie des systèmes de transitions (blocages, branchements, etc.).

On voit déjà dans l'exemple ci-dessus que certaines propriétés géométriques ont une signification en terme de propriétés des exécutions possibles. Ici le trou (carré vide) indique une exclusion mutuelle, alors que le carré plein dénote le vrai parallélisme. D'autres propriétés ont un caractère géométrique également comme les blocages (deadlocks), les branchements et confluences, et donc les équivalences sémantiques ¡¡ branching-time ¿¿, l'ordonnancement des actions, etc. Tout ceci peut se traiter algébriquement de façon élégante en utilisant des outils de topologie algébrique et d'algèbre homologique [Gou95b]. Le modèle lui-même des automates de dimension supérieure peut se formaliser à travers la notion de complexe double de modules (algèbre homologique). Des critères purement algébriques ont été développés pour définir et calculer les propriétés précédemment citées. Par interprétation abstraite, il en découle naturellement des algorithmes d'analyse d'ordonnancement [Gou95c]. À terme, on devrait pouvoir disposer d'un prototype d'analyseur permettant de prouver des propriétés de correction partielle concernant les accès à des objets partagés ou l'ordonnancement des synchronisations et des messages.

On peut définir diverses sortes de systèmes de transitions de dimension supérieure. Les systèmes totaux (HTS) décrivent toutes les transitions de dimension supérieure qui vérifient une propriété de confluence locale. Ils permettent de modéliser la confluence et le branchement à tout niveau de parallélisme. Les systèmes partiels (PHTS) permettent de plus de modéliser l'interférence entre plusieurs actions parallèles, comme par exemple la lecture et l'écriture par deux processeurs d'une même location mémoire.

Le caractère géométrique du modèle des automates de dimension supérieure incite à des généralisations intéressantes telles les automates décrivant les systèmes temps réel et les systèmes probabilistes. Nous avons montré comment obtenir des sémantiques opérationnelles vraiment parallèles, temps réel (éventuellement continu) [Gou96]. Les constructions catégoriques permettent d'ailleurs de donner des lois pour le temps très raisonnables. Enfin les méthodes géométriques développées dans le cas sans temps se transportent aisément au cas avec temps.

2.2.3 Hiérarchie de modèles du vrai parallélisme

Pour comparer les nombreux autres modèles du vrai parallélisme, nous avons développé une nouvelle sémantique opérationnelle du calcul de processus CCS enrichie avec des étiquettes permettant d'encoder l'arbre de dérivation des transitions. Cette information permet de retrouver de nombreux modèles sémanti-

ques classiques sans entrelacement comme la causalité. Nous avons montré que l'approximation de la semantique enrichie par la sémantique de causalité est une interprétation abstraite modélisée par une correspondance de Galois entre les deux modèles. Des définitions similaires devraient permettre de retrouver de nombreux modèles sémantiques présentés dans la littérature [BP97].

2.3 Interprétation abstraite

2.3.1 Fondements de l'interprétation abstraite

L'interprétation abstraite consiste à approcher des propriétés concrètes des programmes par des propriétés abstraites. Au cas où toute propriété concrète a une meilleure approximation dans le domaine de propriétés abstraites, la correspondance entre les propriétés concrètes et abstraites s'établit traditionnellement par une correspondance de Galois. Cette propriété est très puissante car la sémantique concrète et l'approximation définie par la correspondance de Galois définit entièrement la sémantique abstraite qui se déduit constructivement de la sémantique concrète sans avoir à faire de nouvelles approximations. Cette hypothèse est souvent forte quand une propriété concrète peut s'approcher par plusieurs propriétés abstraites sans pour autant qu'il y en ait une plus précise que les autres. Pour prendre un exemple simple, c'est le cas quand il s'agit de décrire un langage sous contexte par une grammaire algébrique. Nous avons étudié les correspondances possibles dans ce cas qu'elles soient basées sur une relation d'approximation, une fonction de concrétisation, une fonction d'abstraction, étudié les situations duales et leurs cas particuliers où interviennent les correspondances de Galois. Ceci conduit à définir des notions de complétude relatives à divers degrés d'abstraction [Cou95d].

Quand les propriétés d'un programme sont exprimées comme solutions d'un système d'équations sur un domaine sémantique infini, l'utilisation d'une correspondance de Galois et d'équations abstraites formalise l'idée de simplification des équations par approximations discrètes. Une autre façon traditionnelle de procéder pour résoudre les équations consiste à utiliser un élargissement/ rétrécissement (widening/narrowing) ce qui formalise l'idée de résolution itérative avec accélération de la convergence par extrapolation. Cette seconde méthode est souvent méconnue ou mal employée (par exemple en utilisant un domaine fini pour effectuer les élargissements). Nous avons montré qu'elle est plus puissante que l'utilisation d'une correspondance de Galois dans un domaine satisfaisant garantissant la terminaison des itérés (satisfaisant par exemple la condition de chaßne). Plus précisément une telle correspondance peut être trouvée pour chaque programme mais il n'en existe en général aucune permettant de trouver pour tous les programmes d'un langage de programmation les mêmes résultats d'analyse de programmes que ceux obtenus en utilisant les techniques élargissement/ rétrécissement dans un domaine abstrait infini. Ceci a

été utilisé pour résoudre des problèmes de convergence laissés ouverts ou résolus grossièrement [CC95c].

Comme évoqué précédemment, les techniques d'analyse sémantique de programmes ont été principalement présentées sous forme de points fixes ou équationnelles. Depuis peu, la présentation sous forme de règles est plus courante. Nous avons introduit G$^\infty$SOS, une généralisation de SOS pour décrire des comportements infinis [Cou95f] et montré que les présentations sous forme de point fixe, équationnelle, de contraintes, de conditions de fermeture, par un système formel à base de règles ou par un jeu sont équivalentes et conservées par approximation basée sur une correspondance de Galois [CC95a].

2.3.2 Domaines abstraits pour l'analyse sémantique de programmes

L'étude d'un domaine abstrait consiste à imaginer un modèle mathématique de propriétés de programmes. Pour être utilisable pour l'analyse automatique de programmes, il faut en concevoir une repésentation machine efficace et trouver des algorithmes performants donnant une version abstraite des opérations/fonctions élémentaires que l'on trouve dans la plupart des langages de programmation.

Utilisation des BDDs en interprétation abstraite

Il s'agit d'utiliser une représentation symbolique des fonctions booléennes pour construire des interprétations abstraites plus performantes quand les propriétés abstraites sont exprimées sous forme de propositions logiques. Les représentations utilisées sont les TDGs (Typed Decision Graphs), une variante des BDDs (Boolean Decision Diagrams) plus compacte. Les TDGs sont utilisés pour représenter aussi bien le domaine abstrait au cours de l'interprétation abstraite que les fonctions abstraites (qui servent à calculer la sémantique particulière d'un programme). Dans cette optique, nous avons donné une méthode générale pour étendre des représentations en TDG à des ordres supérieurs c'est-à-dire des représentations des fonctions sur les domaines de départ. Cette méthode peut permettre de trouver un codage des fonctions abstraites étant donné un codage des domaines abstraits. Il existe des fonctions pour lesquelles la représentation sous forme de TDG n'est pas compacte, ce qui peut consommer énormément d'espace mémoire, mais aussi de temps, car la plupart des opérations sur les TDGs dépendent de leur taille. La solution spécifique à l'interprétation abstraite est d'approximer ces fonctions par d'autres prenant moins de place. Cette approximation peut servir dans le cadre d'un opérateur d'élargissement dynamique si le domaine abstrait est codé, ou encore elle peut servir à approximer dès le départ les fonctions abstraites elles-mêmes, accélérant chaque itération. Comme exemple d'utilisation de ces différents procédés, on a codé l'analyse de nécessité,

pour aboutir à un algorithme très performant, qui permet de trouver des résultats là où aucune analyse à la Mycroft exacte ne peut aboutir faute de place mémoire [Mau94].

Utilisation des langages formels, grammaires et contraintes ensemblistes en interprétation abstraite

Il existe une classe de méthodes d'analyse de programmes basées sur l'usage de grammaires (introduite par N. Jones et S. Muchnick) ou sur celui de contraintes ensemblistes introduite par N. Heinze pour laquelle on a longtemps cru qu'il s'agissait de méthodes radicalement différentes de celles utilisées en interprétation abstraite. N. Heinze n'affirme-t-il pas, page 18 de sa thèse, que : ¡¡ The finitary nature of abstract interpretation implies that there is a fundamental limitation on the accuracy of this approach to program analysis. There are decidable kinds of analysis that *cannot* be computed using abstract interpretation (even with widening and narrowing). The set-based analysis considered in this thesis is one example ¿¿. Au contraire, nous avons montré que ces analyses basées sur l'approximation de languages formels par des grammaires ou des contraintes ensemblistes sont bien des interprétations abstraites isomorphes, avec calcul itératif de point fixe pour lequel on peut utiliser un élargissement, un transformateur de grammaires/contraintes finitaire ou même comme le fait implicitement N. Heinze de manière encore plus simple, un domaine abstrait qui est fini pour chaque programme particulier [CC95c]. Cette nouvelle compréhension des analyses par grammaires ou contraintes a quelques avantages. En particulier le processus d'approximation est formalisé de manière rigoureuse et non plus expliqué en prenant quelques exemples. On met en évidence un domaine abstrait d'usage général qui peut être combiné avec d'autres domaines abstraites (polyhèdres, congruences linéaires, etc.) pour analyser les valeurs numériques voire exprimer des conditions de contexte. Enfin, on montre que quelques affirmations sur les mérites respectifs de ces méthodes d'analyse et de l'interprétation abstraite sont pour le moins non justifiées [CC95c].

Représentation des grammaires et contraintes ensemblistes en interprétation abstraite

Comme nous venons de le dire, une classe importante de méthodes d'analyse de programme est basée sur l'approximation de langages formels par des grammaires et/ou des contraintes ensemblistes. Malgré un usage largement répandu de ces objets, même en dehors du cadre de l'analyse de programmes, on ne sait pas manipuler efficacement les grammaires avec contraintes ensemblistes. Il s'agit de trouver de nouvelles représentations de tels objets, tout en développant des algorithmes de point fixe utilisant ces objets qui soient suffisamment efficaces pour être utilisables en pratique. L'utilisation d'approximations valides permet éventuellement de restreindre la taille des représentations au cours des

calculs.

L'approche la plus générale du problème consiste à considérer ces grammaires comme des ensembles (presque) quelconques, éventuellement infinis, d'arbres éventuellement infinis. Pour que la représentation de ces arbres soit efficace, il faut d'une part qu'elle soit canonique (les tests d'égalités, très utilisés en recherche de point fixe doivent être rapides) et d'autre part compacte. Cela implique la recherche des cycles de l'arbre et leur repliage maximum. Une méthode a été développée qui permet de représenter des structures récursives selon leur pliage maximum, et cela au cours même de leur construction. Cette propriété est maintenue au cours des opérations récursives sur ces structures à un faible coût.

Une façon de représenter certains ensembles infinis d'arbres, et en particulier les grammaires, consiste à partager les branches d'arbres communes. On passe ainsi des ensembles aux relations. Or un arbre dont les feuilles sont des ensembles est une relation carrée, propriété qui ne se conserve pas lors des opérations d'union. Gérer directement des relations dans toute leur généralité, même en permettant des approximations, semble beaucoup trop complexe. Une approche semble donner un compromis acceptable : l'utilisation de la décomposition d'une relation en parties exactes. Une représentation des relations exactes a été mise au point dans le cadre des ensembles d'arbre. Elle permet facilement lorsque la relation devient trop complexe à gérer de l'approximer par au-dessus et par en-dessous par des relations plus simples. La décomposition en relations exactes n'étant pas unique, elle pose les problème de canonicité d'une part, et d'autre part de recherche de la décomposition optimale pour la représentation.

Un fois ces problèmes résolus, il faudra encore vérifier expérimentalement l'intérêt pratique de ces résultats jusqu'à présent entièrement théoriques.

2.4 Typage par interprétation abstraite

Traditionnellement, le typage est une phase conceptuelle distincte de la définition de la sémantique dynamique du langage. Un système d'inférence de types peut se décomposer globalement comme suit : tout d'abord une algèbre de types, ensuite un ensemble de règles logiques affectant un type à chaque expression du langage, enfin un algorithme de reconstruction des types calculant le type le plus général d'une expression quelconque. Classiquement cet algorithme est basé sur l'algorithme d'unification de Robinson. Nous avons développé une autre approche qui consiste à comprendre le typage statique comme une interprétation abstraite d'une sémantique avec typage dynamique, que cette sémantique soit opérationnelle [Mon94b] ou dénotationnelle [Cou97f].

2.4.1 Élargissements pour le typage

Ayant formulé les systèmes de type comme des interprétations abstraites [Mon94b], nous avons amélioré les algorithmes de typage classiques en utilisant les techniques d'élargissement de l'interprétation abstraite. Cette approche permet de relâcher les restrictions arbitraires des algorithmes classiques de typage comme la restriction monomorphe des constantes définies récursivement en ML. Pour illustrer le propos, considérons le système ML+ d'Alan Mycroft. Pour le système de types ML+, le problème d'inférence de types est indécidable. Dans le formalisme de l'interprétation abstraite, cela se traduit par un calcul du plus grand point fixe qui ne termine pas systématiquement [Cou97f, Mon94b]. La solution classique à ce type de problèmes en interprétation abstraite consiste à introduire un opérateur d'élargissement pour accélérer la convergence. Nous avons exhibé toute une famille d'opérateurs d'élargissement qui modélisent le comportement de systèmes d'inférences déjà existants comme ceux de ML ou de Miranda, ou qui améliorent de façon substantielle les systèmes de types basés sur l'algèbre d'Hindley/Milner [Mon94b].

2.4.2 Combinaison de l'analyse statique et du typage

Un deuxième axe de recherche a consisté à combiner l'analyse de types avec d'autres types d'analyse. La combinaison de notre analyse avec les interprétations abstraites d'ensemble de valeurs numériques nous a permis :

- de développer un système de types pour un langage ML parallèle autorisant l'existence de canaux non typés. Ce système de types est basé sur la combinaison de l'algorithme d'inférence de types par interprétation asbtraite et d'un algorithme d'analyse des communications. L'analyse de communication nous permet de déterminer de manière approchée l'ensemble des valeurs (ou plus exactement de leurs types) transitant en un point d'émission et un point de réception du programme. Ceci nous permet d'inférer les divers types des valeurs reçues à partir des types des valeurs envoyées ;

- de développer un système de types pour les listes hétérogènes, c'est-à-dire des listes comprenant des valeurs de types différents [Mon94a].

Comme les types sont définis comme des valeurs abstraites, il est possible d'utiliser les types afin de représenter d'autres propriétés que celles habituellement utilisées en typage. Ainsi, nous nous sommes intéressés aux propriétés de l'analyse par nécessité pour les fonctions d'ordre supérieur en utilisant les méthodes initialement conçues pour les systèmes de types [Mon95b]. En fait, l'algorithme d'inférence obtenu fonctionne de manière satisfaisante, les résultats obtenus étant meilleurs que ceux de Kuo et Mishra.

2.4.3 Construction d'une hiérarchie de types par interprétation abstraite

Un troisième axe de recherche consiste à unifier les différents systèmes de types. Nous avons montré que notre modèle général permettait de modéliser les systèmes de types d'ordre supérieur. Dans un premier temps, nous nous sommes attaqués au système F (introduit par Girard et Reynolds) et il nous a été possible de montrer que ce système est une interprétation abstraite de la sémantique opérationnelle [Mon95a].

Très récemment, nous avons réussi à reconstruire la hiérarchie des systèmes de types du cube de Barendregt (à l'exception du calcul des constructions). Dans un autre domaine, il a été possible de définir une présentation générale uniforme des systèmes de types avec contraintes. Plus généralement, ces derniers systèmes sont très utiles pour la définition des systèmes de types pour des langages objets, pusiqu'ils autorisent la prise en compte de dépendances dynamiques, ce que bien entendu les systèmes de types classiques ne sont pas en mesure d'appréhender. Un intérêt de la présentation par interprétation abstraite est de pouvoir exhiber les différentes restrictions syntaxiques qui ont été proposées par Fuh et Mishra, par Mitchell et par Henglein comme des opérateurs d'élargissement. En conlusion, nous montrons comment définir des opérateurs d'élargissement basés sur des critères sémantiques en lieu et place des critères syntaxiques, ce qui autorise la définition d'algorithmes de résolution de contraintes beaucoup moins grossiers.

L'intérêt théorique de ce résultat est d'avoir enfin une définition uniforme des systèmes de types du premier ordre et des systèmes de types d'ordre supérieur. Il permet de rompre la distinction entre typage ad hoc et typage paramétrique et permet de :

- comparer les systèmes de types d'ordre 1 et d'ordre supérieur,
- de définir un treillis des systèmes de types,
- de composer les différents systèmes de types. Certaines propriétés pouvant être modélisées par des termes d'ordre 1, d'autres par des termes d'ordre supérieur.

L'intérêt pratique de ce résultat est qu'il permet, grâce à une formalisation claire de l'abstraction, de définir des approximations décidables (c'est-à-dire non seulement l'algorithme termine mais trouve la solution de manière efficace) des systèmes de types d'ordre supérieur à l'aide des opérateurs d'élargissement. Des expérimentations sont en cours afin de déterminer les performances respectives des divers opérateurs actuellement étudiés pour le système F.

2.5 Sémantiques abstraites

Une sémantique concrète décrit les comportements possibles d'un programme. Une sémantique abstraite fait de même en considérant des propriétés approchées du contrôle de l'exécution. Cette sémantique est généralement paramétrée par un domaine abstrait formalisant les propriétés considérées pour les objets manipulés par le programme.

2.5.1 Analyse sémantique de programmes fonctionnels d'ordre supérieur par interprétation abstraite

L'analyse de nécessité (strictness analysis) est traditionnellement basée sur une sémantique dénotationnelle dans laquelle les propriétés des fonctions $\phi \in D_1 \mapsto D_2$ s'expriment par un élément de $\wp(D_1) \mapsto \wp(D_2)$. Cette approche ne permet pas de rendre compte de nombreuses autres méthodes d'analyse (par exemple l'analyse de projection de J. Hughes et P. Wadler) comme étant des interprétations abstraites. Une première raison est que certaines propriétés (comme l'absence traduisant la non-utilisation d'un paramètre) ne s'expriment pas comme un élément de $\wp(D_1) \mapsto \wp(D_2)$ mais plutôt comme un élément de $\wp(D_1 \mapsto D_2)$. Une seconde raison vient du fait que les sémantiques dénotationnelles correspondent à des approximations où des informations essentielles sur le déroulement des calculs sont perdus, ce qui n'est pas le cas des sémantiques relationnelles [CC94a] et opérationnelles étendues à des comportements infinis [Cou95f]. Pour traiter ces problèmes nous avons généralisé l'approche des correspondances de Galois aux interprétations abstraites d'ordre supérieur en distinguant ordre partiel d'approximation et pré-ordre de calcul [CC94c].

2.5.2 Analyse de comportement par interprétation abstraite

L'analyse de comportement, introduite dans [CC94c], combine les analyses de programmes fonctionnels ne dépendant pas des valeurs manipulées par le programme. Ceci recouvre l'analyse de nécessité, l'analyse de terminaison à la Mycroft, l'analyse de projection (J. Hughes et P. Wadler), l'analyse d'équivalence partiel (PER analysis de S. Hunt), l'analyse de temps de liaison (binding time analysis), etc. Ce type d'analyse est utilisé pour la compilation de langages paresseux (pour transformer des appels par nécessité/par nom par des appels par valeurs qui sont nettement plus efficaces) et l'évaluation partielle (pour effectuer des calculs dès la compilation).

Un problème récurrent de l'"analyse de nécessité est celui de l'efficacité. Comme exemple d'utilisation des TDGs, nous avons codé l'analyse de nécessité, pour aboutir à un algorithme très performant, qui permet de trouver des résultats là où aucune analyse à la Mycroft exacte ne peut aboutir faute de place mémoire [Mau94].

L'analyse de nécessité traditionnelle comme ci-dessus est basée sur des domaines booléens simples pour lesquels les ordres abstraits d'approximation et de calcul coßncident. Ce n'est pas le cas pour les analyses de comportement [CC94c] dans laquelle on sépare les éléments statiques (pour lesquelles la terminaison est certaine) des éléments dynamiques (pour lesquels la non-terminaison est possible). La situation est alors compliquée par le fait qu'un pré-ordre de calcul conduit à un ensemble de solutions abstraites possibles dont il s'agit d'extraire les plus précises.

Pour étudier ce phénomène, nous avons étudié la combinaison de l'analyse de nécessité (strictness analysis) et des temps de liaison (binding time analysis) [V94, V95] pour un petit langage fonctionnel paresseux. Cette combinaison permet de trouver des résultats significatifs alors qu'elles échoueraient séparement. Le produit précision $\times$ vitesse d'analyse est à l'avantage de l'analyse combinée : le treillis abstrait est beaucoup plus grand que la somme disjointe des deux treillis utilisés pour l'analyse de nécessité et des temps de liaison, et les premières itérations de ces deux analyses sont communes. L'usage de correspondances de Galois relatives à une sémantique opérationnelle permet de caractériser exactement la perte d'information dans l'analyse.

Il est facile d'intégrer des analyses abstraites inférant des propriétés de la logique du premier ordre (pour exprimer terminaison sure, la non-terminaison sure, la dependance par rapport à des variables dont nous ne savions rien, etc.). Le fait de mettre la non-terminaison au même niveau que la terminaison dans le treillis abstrait nous empêche d'abstraire directement une fonction définie récursivement (donc par un plus petit point fixe pour un ordre où non-terminaison $\sqsubseteq$ terminaison). Ce problème a été résolu en définissant des itérations abstraites qui contiennent toujours l'abstraction des itérations concrètes de notre fonction (du moins pour un nombre fini d'itérations). Le passage à la limite se justifie par le fait que nous inférons des propriètés de la logique du premier ordre, et qu'ainsi il existe un sous-modèle fini sur lequel les itérés abstraits sont également valables.

Ensuite, ce travail a été élargi de manière à pouvoir énoncer des propriétés générales concernant le comportement d'un programme, et en particulier celui de ses fonctions (terminaison, valeur, structures de données, types). Le langage initial a été élargi à l'ordre supérieur pour des structures de données arbitraires, avec la possibilité de distinguer des structures arbitrairement grandes, mais finies, des structures infinies. Cela a nécessité l'introduction de preuves de terminaison totales dans le cadre de l'interprétation abstraite, et l'introduction d'une fonction d'abstraction correspondant à la preuve du théorème de Tarski.

L'analyse doit maintenant être généralisé de manière à pouvoir traiter des langages de programmation réels : en pratique, cela consiste à donner le comportement d'un programme impératif avec tous les effets de bords qu'il est susceptible de générer. L'intérêt serait d'analyser des programmes écrits dans un petit langage orienté-objet. Les propriétés dégagées peuvent servir soit au compilateur, soit être délivrées en l'état au programmeur, ce qui lui fournit un

metteur au point abstrait.

2.5.3 Analyse sémantique de programmes logiques par interprétation abstraite

Ce travail est basé sur une sémantique opérationnelle des petits pas de programmes PROLOG (résolution SLD). Les principales contributions concernent l'usage de domaines abstraits infinis (les analyses classiques utilisent des treillis finis), en particulier des langages formels, grammaires et contraintes ensemblistes [Cou95g] et la combinaison des analyses descendantes et ascendantes (les analyses classiques utilisent l'une ou l'autre mais pas les deux) [Cou95c].

2.5.4 Analyse sémantique de programmes parallèles par interprétation abstraite

Certaines analyses fines de programmes parallèles comme l'exclusion mutuelle ou l'indépendance et la validation de propriétés liées à la géométrie des systèmes de transitions (blocages, branchements, etc.) requièrent l'usage de sémantiques qui décrivent véritablement le parallélisme entre processus concurrents à l'aide d'automates munis de transitions de dimension supérieure. Ces automates nous permettent de définir des sémantique approchées, calculables par interprétation abstraite en vue de l'analyse statique des programmes parallèles considérés. Cela a été réalisé pour différents types de langages (Linda, Parallel Pascal, CML) et une propriété de correction sémantique de l'interprétation abstraite a été prouvée. Deux techniques différentes permettent d'atteindre cet objectif :

- L'approche dénotationnelle vise à calculer l'automate abstrait de manière compositionnelle, en donnant des équations sémantiques structurées sur la syntaxe. Cela fonctionne pour des algèbres de processus comme CCS ou le Linda-calcul que nous avons défini. Mais les équations se compliquent lorsqu'un état partagé est à prendre en compte.

- C'est pourquoi une approche opérationnelle a été préférée pour des langages plus complexes mais plus réalistes comme CML ou Parallel Pascal. Il suffit alors de donner les règles définissant les transitions à grain fin autorisées entre deux configurations. Les transitions de dimension supérieure se construisent aisément à l'aide des transitions de dimension plus basse et de conditions d'indépendance.

Une caractérisation des interprétations abstraites entre deux automates de dimension supérieure a été donnée en terme d'adjonctions. La *troncation* est une adjonction qui permet de modéliser la sémantique d'un pool de n processeurs qui entrent en compétition afin d'exécuter un programme. Les *repliements* d'états permettent de simplifier les configurations abstraites et de rendre la sémantique approchée calculable statiquement.

L'implantation d'un analyseur de Parallel Pascal basé sur une telle sémantique a été effectuée et est disponible sur le serveur WEB du DMI. Certaines techniques d'implémentation comme l'utilisation de transitions d'un grain plus grossier que dans les définitions sémantiques (Virtual Coarsening) permettent l'interprétation abstraite précise de programmes de taille raisonnable car elles diminuent fortement (au moins d'un facteur 4) le nombre d'états à considérer. Pour des algorithmes classiques d'exclusion mutuelle comme ceux de Dekker ou de Peterson, la preuve d'invariance est ainsi réalisée par l'interprète abstrait en quelques secondes de calcul. L'automate abstrait qui en résulte compte un millier d'états alors que le nombre d'états réels de ce programme est de l'ordre de cent mille. On gagne donc un facteur 100 sur la vérification de la propriété d'exclusion mutuelle dans cet exemple significatif [Cri96a].

D'autres programmes (avec des sémaphores par exemple) ont été analysés avec succès. L'analyseur adjoint au source du programme des invariants calculés à chaque point de contrôle. Il produit aussi l'automate qu'on peut visualiser ou bien fournir comme input à une deuxième passe d'analyse.

2.5.5 Combinaison de l'interprétation abstraite et du model-checking

La vérification de modèle (model-checking) est une technique séduisante puisque, contrairement à l'interprétation abstraite, elle fournit des résultats exacts. Malheureusement, et contrairement à l'interprétation abstraite, elle n'aboutit pas toujours, faute de temps, d'espace mémoire ou parce qu'il faut explorer exhaustivement un espace d'états infini. La vérification de modèle de modèle et l'analyse de programme par interprétation abstraite apparaissent donc comme complémentaires. Nous avons montré comment les combiner dans un calcul parallèle [CC97b], l'analyse du système, convenablement conduite en parallèle avec la vérification du modèle permettant de réduire l'espace des états devant être effectivement explorés. Dans cette combinaison l'analyse est approchée et le model-checking est exact de sorte que la terminaison n'est toujours pas assurée. Une autre approche, celle du model-checking abstrait est également explorée, pour garantir la terminaison au prix d'une perte de précision.

2.5.6 Logique temporelle et model-checking abstrait

L'idée du model-checking abstrait est de valider les formules de logique modale qui caractérisent certaines propriétés comme les blocages (deadlocks), l'équité ou l'exclusion mutuelle sur les systèmes de transitions approchés obtenus par interprétation abstraite, et d'en tirer une information sur la sémantique concrète. En effet, contrairement au model-checking traditionnel, le système de transitions d'un programme n'est pas calculable en général [Cri95, Cri96b].

On peut quand même tirer parti du fait qu'il existe d'excellents algorithmes de model-checking sur les automates. La difficulté principale est de réussir à

transférer la validité d'une formule de logique temporelle suffisamment expressive de la sémantique abstraite vers la sémantique standard. Ce résultat a été obtenu pour le μ-calcul complet grâce à une interprétation abstraite qui allie à la fois une approximation supérieure et une approximation inférieure des états concrets.

Un évaluateur de formules logiques du μ-calcul est en ce moment testé en sortie de l'interprète abstrait pour réaliser un outil plus général d'analyse de programmes parallèles.

2.5.7 Ordonnancement de programmes parallèles

Dans le modèle des automates de dimension supérieure, on a la possibilité de distinguer de façon très fine les ordonnanceurs possibles d'un programme parallèle tournant sur une machine contrainte (nombre de processeurs limité, topologie de communication imposée, exclusion mutuelle obligatoire pour certains objets...). Toutes ces conditions contraignent les exécutions possibles et on peut donner des critères géométriques (homotopie) pour déterminer s'il y en a encore (spécification/vérification) et même pour déterminer les meilleures possibles. A contrario, on peut aussi essayer de paralléliser au maximum un programme séquentiel afin d'occuper au maximum un type de machine donné. Ces deux problèmes se posent en des termes géométriques similaires, et sont naturellement exposés dans le cadre de l'interprétation abstraite [Gou95c].

Nous avons commencé à nous intéresser aux applications des sémantiques du vrai parallèlisme par automates de dimensions supérieures aux protocoles de systèmes distribués tolérants aux pannes. Un premier travail a permis de lier les considérations géométriques des automates de dimension supérieure à celles de M. Herlihy, N. Shavit et S. Rajsbaum concernant les systèmes distribués t-résilients (résistants à t pannes). Cela a permis de prouver à nouveau de façon sémantique un résultat de calculabilité sur les systèmes t-résilients à mémoire partagée (avec lecture et écriture atomique). Cette nouvelle vision des choses a rendu possible l'amélioration de l'algorithme d'Herlihy pour la génération automatique de protocoles sans attente [Gou97b], ainsi que la généralisation à d'autres systèmes (avec test & set etc.).

2.5.8 Analyse de programmes parallèles stochastiques

Nous avons étendu le π-calcul par des informations stochastiques qui donnent un modèle uniforme avec lequel il est possible d'étudier des propriétés qualitatives (comportements) et quantitatice (performances) des systèmes parallèles et distribués. À titre d'analyse de cas, nous avons étudié les procédures de transmission dans les réseaux de téléphonie mobile [Pri97].

2.5.9 Analyse sémantique de langages orientés objets par interprétation abstraite

L'objet de cette recherche, commencée récemment, est de définir un cadre sémantique pour l'interprétation abstraite des langages orientés objets. Les applications envisagées concernent le typage, l'optimisation dans les compilateurs (par exemple pour transformer des appels à des fonctions virtuelles en appels statiques) et la mise au point abstraite (où les valeurs utilisées pour la mise au point interactive sont abstraites c'est-à-dire à des ensembles de valeurs concrètes qu'il faudrait considérer individuellement dans les approches classiques).

2.6 Perspectives

Les perspectives doivent être situées dans le contexte fixé par le DMI. La croissance des surfaces occupées et du nombre de chercheurs est nulle, ce qui pose à la fois le problème de masse critique et d'accueil des post-doctorants (programme européen CHM), années sabbatiques, etc. L'équipe est largement constituée de jeunes chercheurs thésards dont le renouvellement est rapide et qui quittent l'équipe quand ils atteignent leur maturité scientifique. En dehors du responsable, il n'y a pas de chercheurs confirmés et d'ingénieurs de recherche affectés à des projets scientifiques à moyen terme, ce qui limite considérablement les possibilités de développements pratiques et du suivi de ces développements. La durée de vie de l'équipe, comme toutes celles du DMI, est limitée dans le temps, etc. Ces conditions de travail nous placent dans une position de relative faiblesse par rapport à nos partenaires et concurrents européens.

Malgré ces conditions difficiles, l'objectif essentiel est de fournir un excellent cadre de travail aux jeunes thésards. Nos atouts sont la qualité des chercheurs, des thèmes de recherche, des indispensables partenaires extérieurs et de l'animation scientifique que crèent le séminaire et les visiteurs extérieurs.

En ce qui concerne nos idées sur la sémantique et l'interprétation abstraite, elles se répandent doucement mais sûrement, notamment au travers d'écoles pour jeunes chercheurs :

- Tutorial sur l'interprétation abstraite, ICCL'94, Toulouse, mai 1994 [Cou94] ;
- Dagstuhl-Seminar on Abstract Interpretation, 28 août – 1er septembre 1995 [CCME95] ;
- École jeunes chercheurs en programmation, Bordeaux, 25 mars – 5 avril 1996 [Cou96e]
- ESSLLI'96, Eighth European Summer School in Logic, Language, and Information, 12 – 23 août 1996 [Cou96f] ;

- Dottorato di Ricerca Corso di Interpretazione Astratta, Universit di Pisa, 9 – 20 septembre 1996 [Cou96c] ;
- École jeunes chercheurs en programmation, INRIA Sophia Antipolis, 17 – 28 mars 1997 [Cou97a] ;
- Indo-French School on Abstract Interpretation, 14 – 20 April 1997, JNCPAR, Bangalore, Inde, organisée par R.K. Shyamasundar [Cou97b].
- NATO International Summer School 1998 on Calculational System Design. Marktoberdorf, Germany, 28 july–9 august 1998, organized by F.L. Bauer, M. Broy, E.W. Dijkstra, D. Gries and C.A.R. Hoare.

Le succès peut également se mesurer aux nombreuses conférences invitées[1] dans des congrès ICCL'94 [Cou94], GULP-PRODE'95 [Cou95d], CAV'95 [Cou95a], POPL'97 [Cou97f], MFPS XIII [Cou97c], SAS'97 ou réunions thématiques (workshops) WAILL'95 [Cou95c], 4th Compulog [Cou95g], Venice workshop'95 [Cou96d], F.N.R.S. meeting [Cou97e].

Un certain nombre d'éléments prospectifs ont déjà été évoqués dans le compte-rendu du travail des quatre dernières années. Nous comptons nous concentrer sur quelques thèmes, comme par exemple :

- Le développement de méthodes de définition de sémantiques (parallélisme véritable, langages objets, etc.) que ce soit pour les domaines de calcul, les méthodes de présentation des sémantiques, l'étude des relations entre sémantiques différentes d'un même langage ;
- La conception de domaines abstraits pour l'analyse symbolique relationnelle de structures de données non numériques ;
- L'interprétation abstraite de structures de programmes et de contrôle sous forme de sémantiques abstraites génériques pour les diverses formes de parallélisme (processus mobiles), de modules, les langages objets, etc. ;
- L'étude des méthodes d'approximation effective de points fixes ;
- Les méthodes formelles de conception d'analyseurs de programmes basées sur l'interprétation abstraite de sémantiques pour mettre en évidence une structure modulaire facilement réutilisable voire en envisager la construction semi-automatique ;
- La combinaison de l'analyse sémantique avec d'autres méthodes de spécification et de vérification de programmes.

[1] Contrairement à d'autres disciplines ces invitations par le comité de programme sont beuacoup plus prestigieuses que les soumissions normales.

Part III

Eléments d'appréciation de l'activité du laboratoire

Chapter 3

Collaborations

- Contrat ¡¡ Action Informatique 92 ¿¿ du MRT, *Techniques symbolique pour l'analyse statique et la vérification des systèmes*, BULL (J. Goubault & D, Bolignano), IMAG (N. Halbwachs), LIENS (P. Cousot), novembre 1992 à novembre 1996 [CGH96].

- Projet ¡¡ Sémantique, interprétation abstraite et parallélisme ¿¿, thème 2.1 : ¡¡ Spécification et vérification ¿¿ du GDR 1169 du CNRS ¡¡ Parallélisme, Réseaux et Systèmes ¿¿, depuis octobre 1994 ;

- Groupe de travail 6809 ¡¡ SÉMANTIQUE ¿¿ du programme européen ¡¡ ESPRIT II Basic Research Action ¿¿ *Semantics-based program manipulation techniques* liant l'École Normale Supérieure (P. Cousot), l'École Polytechnique (R. Cousot), l'Imperial College (C. Hankin, coordinateur), l'Université de Chalmers (J. Hugues), l'Université de Copenhagen (N. Jones), l'Université d'Århus (F. Nielson) et l'Université de Glasgow (P. Wadler), de juillet 1992 à septembre 1995 [Cou95f, CC95b].

- Projet 6809 ¡¡ LOMAPS ¿¿ du programme européen 'ESPRIT III Basic Research Action' *Logical and operational methods in the analysis of programs and systems* liant l'École Normale Supérieure (P. Cousot), l'École Polytechnique (R. Cousot), le Swedish Institute of Computer Science (M. Dam), l'Université de Pise (P. Degano), l'École Des Mines de Paris (P. Jouvelot), l'Université de Cambridge (A. Mycroft), l'Université d'Århus (F. Nielson, coordinateur), le European Computer-Industry Research Center (B. Thomsen), de décembre 1993 à juin 1997 [NCD$^+$95, NCD$^+$96b, NCD$^+$96a].

- Projet NSF-ESPRIT ¡¡ Atlantique ¿¿ liant l'École Normale Supérieure (P. Cousot), l'École Polytechnique (R. Cousot), l'Université d'Århus (F. Nielson & H. Nielson), l'Université de Boston (M. Wand), l'Université de

Carnegie Mellon (P. Lee), l'Université de Chalmers (J. Hughes), l'Université de Copenhague (N. Jones), l'Université de Glasgow (P. Wadler), l'Université du Kansas (D. Schmidt), l'Université de New York (A. Turchin), l'Université d'Oregon (J. Launchburry), l'Université de Stanford (C. Talcott) et l'Université de Yale (P. Hudak) de janvier 1993 à juin 1997 [CC96b].

- Réseau ¡¡ ABILE ¿¿ du programme européen 'CHM (Capital Humain et Mobilité) *Network on abstract interpretation for declarative languages*, liant l'Université Catholique de Louvain (M. Bruynooghe), les Facultés Universitaires de Namur (B. Le Charlier, coordinateur), l'Université de Copenhagen (N. Jones), l'Université de Bordeaux (M. Corsini), l'École Normale Supérieure (P. Cousot), l'Université de Lille (P. Devienne), l'INRIA, Rocquencourt (P. Codognet), le Max-Planck-Institut für Informatik de Saarbrücken (M. Hanus), l'Université de Cambridge (A. Mycroft), l'Université de Padoue (G. Filé), l'Université de Pise (G. Levi), l'Université Polytechnique de Madrid (M. Hermenegildo) et l'Université de Linköping (J. Małuszyński), depuis janvier 1995.

Chapter 4

Missions, conférences et séminaires

- Participation à des conférences :
  - P. Cousot : ICCL'94 (Toulouse, mai 1994) [Cou94, CC94c], WAILL'95 (Eilat, juin 1995) [Cou95c], FPCA'95 (La Jolla, Californie, juin 1995) [CC95c], CAV'95 (Liège, juillet 1995) [Cou95a, CC95a], Dagsthul seminar 9535 (Sarrebruck, septembre 1995) [CCME95], GULP-PRODE (Salerno, Italie, septembre 1995) [Cou95d], WLDP'95 (Darmstadt, mai 1995) [Cou95e], École jeunes chercheurs en programmation (Bordeaux, avril 1996) [Cou96e], 4th Compulog (Marina del Vietri,Italie, mai 1995) [Cou95g], MOVEP'96 (Nantes, juin 1996) [Cou96h], Symposium on models of programming languages and computation (MIT, Boston, juin 1996) [Cou96a], ESSLLI'96 (Prague, août 1996) [Cou96f], Venice Workshop (Venise, septembre 1996) [Cou96d], Corso di Interpretazione Astratta, Dottorato di Ricerca, Univ. di Pisa (Pise, septembre 1996) [Cou96c], POPL'95 (Paris, janvier 1997) [Cou97f], AAS'97 (Paris, janvier 1997) [CC97b], École jeunes chercheurs en programmation (Sophia-Antipolis, mars 1997) [Cou97a], MFPS XIII (Pittsburgh, mars 1997) [Cou97c], Indo-french school on abstract interpretation (Bangalore, Inde, avril 1997) [Cou97b], Namur F,N.R.S. meeting (Namur, mai 1997) [Cou97e] ;
  - R. Cridlig : JFLA (Noirmoutier, janvier 1994) [Cri95], PEPM'95 (La Jolla, Californie, juin 1995) [Cri95], Lomaps Workshop, Stockholm, avril 1996 [CC96a, Cri96a], Infinity Workshop (Pise, août 1996) [Cri96b] ;
  - É. Goubault : PEPM'95 (La Jolla, Californie, juin 1995) [Gou95c], ESOP'96 (Linköping, Suède, avril 1996) [Gou96], 3^{rd} Theory and

Formal Methods Section Workshop (Londres, 1997) [Gou97c], 22^{nd} CAAP (Lille, 1997) [Gou97b] ;

- L. Mauborgne : SAS'94 (Namur, 1994) [Mau94] ;
- B. Monsuez : SAS'94 (Namur, septembre 1994), PEPM'95 (La Jolla, Californie, juin 1995) [Mon95b], SAS'95 (Glasgow, septembre 1995) [Mon95a] ;
- F. Védrine : SAS'95 (Glasgow, septembre 1995) [V95].

• Séminaires :

- P. Cousot : Carnegie Mellon University (septembre 1994) [CC94b], École Polytechnique (avril 1995) [Cou], ENSTB (janvier 1996) [Cou] Université de Pise (septembre 1996) [Cou96b] ;
- R. Cridlig : CRIN, Nancy (mars 1994), École Normale Supérieure (mars 1995, avril 1997) ;
- É. Goubault : CRIN, Nancy (1994), École Normale Supérieure (mai 1995) ;
- L. Mauborgne : École Normale Supérieure (1994) ;
- B. Monsuez : École Normale Supérieure (février 1995), LRI, Université Paris XI (avril 1994) ;
- F. Védrine : École Normale Supérieure (mars 1995, mai 1996) ;

Chapter 5

Accueil de chercheurs

Professeurs et directeurs de recherche invités

- Christopher Colby, Carnegie Mellon University, Pittsburg, USA, chercheur invité sur le projet Atlantique, mai/août 1995 ;
- Peter Lee, Carnegie Mellon University, Pittsburg, USA, un mois, comme professeur invité de l'École Normale Supérieure en mai/juin 1995 ;
- Leslie Lamport, DEC-SRC, Palo Alto, USA, un mois, comme professeur invité de l'École Normale Supérieure en septembre 1995 ;

  Pierpaolo Degano, Dipartimento di Informatica, Università di Pisa, un mois, comme professeur invité de l'École Normale Supérieure en janvier 1997.

Chapter 6

Diffusion de la connaissance

- Organisation à l'École Normale Supérieure, d'un séminaire régulier ¡¡ sémantique et interprétation abstraite ¿¿. Ce séminaire accueille occasionnellement des orateurs prestigieux mais a principalement pour objet de permettre à de jeunes chercheurs de présenter leurs travaux et de les voir discutés :
  - 1994/95 : N. Benton, F. Bourdoncle, C. Colby, R. Cridlig, V. Danos, A. Deutsch, F. Fages, É. Goubault, J. Goubault, Ph. Granger, R. Heckmann, N. Heintze, T. Jensen, B. Monsuez, B. Le Charlier, P. Lee, I. Mackie, A. Mycroft, F. Nielson, F. Védrine ;
  - 1995/96 : S. Abramsky, A. Banerjee, B. Blanchet, R. Cridlig, P. Degano, P. Emelianov, S. Gay, R. Giacobazzi, É. Goubault, J. Goubault, J. Gunawardena, A. Kennedy, L. Lamport, P. Lee, I. Mackie, D. Méry, V. Pratt, F. Védrine, A. Venet ;
  - 1996/97 : V. Benzaken, R. Cridlig, P. Degano, L. Fajstrup, A. Frey, J. Goubault, J. Gunawardena, F. Levi, I. Mackie, S.-O. Nyström, A. Podelski, C. Priami, M. Raussen, K. Rustan M. Leino, O. Shivers.
- Organisation de réunions de groupes de travail :
  - P. Cousot a organisé la réunion du groupe de travail 2.3 de l'IFIP à Obernai (septembre 1997) ;
  - É. Goubault et B. Monsuez ont organisé la réunion du projet Esprit BRA 6809 ¡¡ Lomaps ¿¿ à l'École Normale Supérieure du 6 au 9 janvier 1994, participation de R. Cridlig, P. Cousot, L. Mauborgne et et F. Védrine ;
  - P. Cousot a organisé la réunion du projet Esprit BRA 6809 ¡¡ Lomaps ¿¿ à l'École Normale Supérieure les 13 et 14 janvier 1997, participation de R. Cridlig, É. Goubault, L. Mauborgne, B. Monsuez et F. Védrine ;

- P. Cousot a organisé la réunion du projet Esprit ¡¡ Atlantique ¿¿ à l'École Normale Supérieure le 18 janvier 1997, participation de R. Cridlig.

• Participation à des séminaires et réunions de groupes de travail :

- P. Cousot, R. Cridlig, É. Goubault, B. Monsuez et F. Védrine ont participé aux réunions du groupe de travail Esprit BRA 6128 ¡¡ Sémantique ¿¿ à Århus (juillet 1994) et Glasgow (septembre 1995) [Cou95f, CC95b] ;
- R. Cridlig, É. Goubault, B. Monsuez, L. Mauborgne et F. Védrine ont participé à la réunion du projet Esprit BRA 6809 ¡¡ Lomaps ¿¿ à Fontainebleau du 16 au 18 mai 1994 et avec P. Cousot à celles de Cambridge du 3 au 5 juillet 1995, Pise du 12 au 15 décembre 1996 et Stockhom en juin 1996 ;
- P. Cousot a participé aux réunions du réseau Esprit-NSF ¡¡ Atlantique ¿¿ à Portland, Oregon (janvier 1994) et San Franscisco (janvier 1995) et avec R. Cridlig, É. Goubault et F. Védrine à celle d'Århus (juillet 1994) ;
- R. Cridlig a été invité par le pôle ¡¡ Programmation Fonctionnelle ¿¿ du GDR ¡¡ Programmation ¿¿ à la réunion de Noirmoutier en janvier 1994 ;
- R. Cridlig a participé à la réunion du groupe de travail SIP du GDR ¡¡ Parallélisme, réseaux et systèmes ¿¿ à Grenoble en décembre 1994 ;
- P. Cousot organise (avec R. Cousot & A. Mycroft) le Dagstuhl-Seminar on "Abstract Interpretation" du 28 août 1995 au 1er septembre 1995 ;
- P. Cousot participe au Dagstuhl-Seminar on "New trends in the integration of paradigms" (organisateurs C. Hankin & H. Riis Nielson) du 18 au 22 septembre 1995.

• Participation à des écoles de chercheurs :

- Tutorial sur l'interprétation abstraite, P. Cousot, ICCL'94, Toulouse, mai 1994 [Cou94] ;
- Dagstuhl-Seminar on Abstract Interpretation, P. Cousot, B. Monsuez, 28 août – 1er septembre 1995 [CCME95] ;
- École jeunes chercheurs en programmation, P. Cousot, Bordeaux, 25 mars – 5 avril 1996 [Cou96e]
- ESSLLI'96, Eighth European Summer School in Logic, Language, and Information, P. Cousot, 12 – 23 août 1996 [Cou96f] ;

  - Dottorato di Ricerca Corso di Interpretazione Astratta, Universit di Pisa, P. Cousot, 9 – 20 septembre 1996 [Cou96c] ;
  - École jeunes chercheurs en programmation, P. Cousot, INRIA Sophia Antipolis, 17 – 28 mars 1997 [Cou97a] ;
  - Indo-French School on Abstract Interpretation, P. Cousot, B. Monsuez, F. Védrine, 14 – 20 April 1997, JNCPAR, Bangalore, Inde, organisée par R.K. Shyamasundar [Cou97b].

- R. Cridlig est coéditeur du bulletin C^3R de veille technologique de la DRET : article sur le langage JAVA et la sécurité des applets.

Chapter 7

Réalisation et diffusion de logiciels, brevets

- R. Cridlig a implanté un analyseur de Parallel Pascal qui est disponible sur le serveur WEB du DMI.

Chapter 8

Participations à l'évaluation de la recherche

- P. Cousot :
  - membre du comité de rédaction de la revue ¡¡ Science of Computer programming ¿¿ ;
  - membre des comités de programme de ICCL'94, SAS'94, PLILP'94, PLILP'95, SAS'95, JFPLC'96 ;
  - membre du comité scientifique du laboratoire d'ingénierie des systèmes d'information (LISI) de l'Institut National des Sciences Appliquées de Lyon et de l'Université Claude Bernard – Lyon I ;
  - expert de recrutements (referee for academic promotion comitees) DAIMI (Århus, Danemark), DIKU (Copenhague, Danemark), Imperial College of Science, Technology and Medicine (Londres), INRIA, Universités de Cambridge, Loyola (Chicago), Melbourne, Sarrebrück, Ben-Gurion University of the Negev (Israël) ;
  - membre des commissions de spécialistes d'informatique, 27$^{\text{ème}}$ section de l'École Normale Supérieure et de l'Université de Paris 9, Dauphine ;
  - membre du conseil de laboratoire du LIENS ;
  - expertises de soumissions de projets de recherche : projets américains de la NSF, projets européens ESPRIT et LTR, Israel Science Foundation ;
  - expertise (reviewer) de projets européens ESPRIT II et III :
    - Esprit B.R.A. project 6021 REACT (P. Wolper (coordinateur), Liège, Belgique, 8 juillet 1994 et 20–22 novembre 1995) ;

    - ∗ Esprit B.R.A. project 7071 PROCOS II (C. Hoare (coordinateur), Bruxelles, 30 septembre 1994 et 13 octobre 1995) ;
  - évaluateur (referee) pour les revues Acta Informatica, Information Processing Letters, Science of Computer Programming, TOPLAS et les conférences ESOP'94, ICCL'94, ILPS'94, SAS'94, PEPM'94, POPL'95, PLILP'95, SAS'95, POPL'96, SAS'96 ;

- R. Cridlig :
  - évaluateur (referee) pour les conférences SAS'94, ICCL'94, SAS'95, ESOP'96, FMPPTA'96, SAS'97, FMPPTA'97, FME'97 ;

- É. Goubault :
  - évaluateur (referee) pour les revues Fundamenta Informaticae (1993), Information Processing Letters (1994), Theoretical Computer Science (1994), Mathematical Structures in Computer Science (1995) et les conférences SAS'94, STACS'95, PEPM'95, CONCUR'95, FMPPTA'96, SAS'97, FMPPTA'97 ;

- B. Monsuez :
  - membre du conseil de Laboratoire du LIENS.
  - évaluateur (referee) pour les conférences SAS'94, PLILP'95, SAS'95, SAS'97 ;

Chapter 9

Encadrement doctoral

9.1 Direction de thèses

Nous donnons uniquement la liste des thèses soutenues entre 1994 et 1996 et dirigées par un membre du laboratoire.

- Patrick Cousot
  - B. Monsuez, École Polytechnique, soutenue en février 1994
    Typage par interprétation abstraite [Mon94b]
  - É. Goubault, École Polytechnique, soutenue en février 1994
    Typage par interprétation abstraite [Gou95b]

9.2 Participation à d'autres jurys de thèses

Pour chaque chercheur concerné, on indique le nombre de jurys dans lequel il a siégé, non compris ceux des thèses qu'il a dirigées, et le nombre de rapports de thèse qu'il a rédigés.

- P. Cousot : 8 jurys (dont 2 habilitations), 1 présidence, 3 rapports.

Chapter 10

Enseignement

- P. Cousot : École Polytechnique, Magistère M.M.F.A.I. et DEA IMA ;
- R. Cridlig : École Polytechnique, ENSTA ;
- É. Goubault : École Polytechnique, ENSTA ;
- B. Monsuez : École Polytechnique, Magistère M.M.F.A.I. ;
- L. Mauborgne : École Polytechnique ;
- F. Védrine : ENSTA, Magistère M.M.F.A.I..

Bibliography

Articles invités

[CC94c] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *Proceedings of the 1994 International Conference on Computer Languages*, Toulouse, France, pages 95–112. IEEE Computer Society Press, Los Alamitos, CA, 16–19 May 1994.

[CC95a] P. Cousot and R. Cousot. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form, invited paper. In P. Wolper, editor, *Computer Aided Verification, Proceedings of the 7^{th} International Conference, CAV'95*, Liège, Belgium, Lecture Notes in Computer Science 939, pages 293–308. Springer-Verlag, Berlin, July 1995.

[Cou97f] P. Cousot. Types as abstract interpretations, invited paper. In *Conference Record of the 24^{th} Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 316–331, Paris, France, 1997. ACM Press, New York.

Articles dans des revues internationales avec comité de lecture

[Cou96a] P. Cousot. Abstract interpretation. *Symposium on Models of Programming Languages and Computation, ACM Computing Surveys*, 28(2):324–328, June 1996.

[Cou96g] P. Cousot. Program analysis: The abstract interpretation perspective. *ACM Computing Surveys*, 28A(4), December 1996.

Conférences invitées

[Cou94] P. Cousot. A tutorial on abstract interpretation. In *1994 IEEE International Conference on Computer Languages*, Toulouse, France, 16–19 May 1994.

[Cou95a] P. Cousot. Abstract model checking, invited lecture. In *Computer Aided Verification, Proceedings of the 7^{th} International Conference, CAV'95*, Liège, Belgium, 5 July 1995.

[Cou95c] P. Cousot. Combining bottom-up and top-down in abstract interpretation of logic languages, invited lecture. In *Special Workshop on Abstract Interpretation of Logic Languages, WAILL'95*, Eilat, Israël, 18–19 June 1995.

[Cou95d] P. Cousot. Completeness in abstract interpretation, invited lecture. In M.I. Sessa and M. Alpuente, editors, *Proceedings of the GULP-PRODE'95 Joint Conference on Declarative Programming*, Marina di Vietri, Italy, pages 37–38. Poligraf Press, Salerno, Italy, 11–14 September 1995.

[Cou95g] P. Cousot. Set-constraint-based analysis of logic programs by abstract interpretation, invited lecture. In M. Gabbrielli, editor, *Proceedings of the Fourth Compulog-network subgroup meeting on Programming Languages*, Marina di Vietri, Italie, pages 1–2, 15 September 1995.

[Cou96d] P. Cousot. From semantics to classical proof methods by abstract interpretation, invited lecture. Workshop on "Program Correctness: Abstract Interpretation vs. Classical Verification Methods", Venise, Italie, 10–12 June 1996.

[Cou97c] P. Cousot. Design of semantics by abstract interpretation, invited address. In *Mathematical Foundations of Programming Semantics, Thirteenth Annual Conference (MFPS XIII)*, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 23–26 March 1997.

[Cou97e] P. Cousot. Types as abstract interpretations, exposé invité. Meeting on Validation and Verification of Formal Descriptions, Fundamental Computer Science F.N.R.S. Contact Group, F.U.N.D.P., Namur, Belgique, 6 May 1997.

Communications dans des conférences internationales avec comité de lecture

[CC77] ← P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4^{th} Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los AngeLes, California, 1977. ACM Press, New York.

[CC95c] P. Cousot and R. Cousot. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In *Proceedings of the 7^{th} ACM Conference on Functional Programming and Computer Architecture*, pages 170–181, La Jolla, California, 25–28 June 1995. ACM Press, New York.

[CC97b] P. Cousot and R. Cousot. Parallel combination of abstract interpretation and model-based automatic analysis of software. In R. Cleaveland and D. Jackson, editors, *Proceedings of the first ACM Sigplan workshop on Automatic Analysis of Software, AAS'97*, pages 91–98, 23-26 March 1997.

[Cri95] R. Cridlig. Semantic analysis of shared-memory concurrent languages using abstract model-checking. In *Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation, PEPM'95*, La Jolla, California, 21–23 June 1995. ACM Press, New York, June 1995.

[Cri96a] R. Cridlig. Implementing a static analyzer of concurrent programs: problems and perspectives. In M. Dam and F. Orava, editors, *Proceedings of the 5^{th} LOMAPS Workshop on Analysis and Verification of Multiple-agent Languages*, Lecture Notes in Computer Science 1192, pages 244–259. Springer-Verlag, Berlin, June 1996.

[Cri96b] R. Cridlig. Semantic analysis of Concurrent ML by abstract model-checking. In B. Steffen and T. Margaria, editors, *Proceedings of the International Workshop on Verification of Infinite State Systems*. Universität Passau, MIP-9614, August 1996. to be published in Electronic Notes on Theoretical Computer Science, 1997.

[Gou95c] É. Goubault. Schedulers as abstract interpretations of higher-dimensional automata. In *Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation, PEPM'95*, La Jolla, California, 21–23 June 1995, pages 134–145. ACM Press, New York, June 1995.

[Gou96] É. Goubault. Durations for truly-concurrent actions. In H. Riis Nielson, editor, *Proceedings of the 6^{th} ESOP'96, Linköping, Sweden*, number 1058 in Lecture Notes in Computer Science, pages 173–187. Springer Verlag, Berlin, April 22–26 1996.

[Gou97b] É. Goubault. Optimal implementation of wait-free binary relations. In *Proc. of the 22nd CAAP*, Lecture Notes in Computer Science. Springer Verlag, Berlin, 1997.

[Gou97c] É. Goubault. A semantic view on distributed computability and complexity. In *Proceedings of the 3rd Theory and Formal Methods Section Workshop*. Imperial College Press, 1997.

[Mau94] L. Mauborgne. Abstract interpretation using TDGs. In B. Le Charlier, editor, *Proceedings of the Static Analysis Symposium, SAS'94*, Namur, Belgium, 20–22 September 1994, Lecture Notes in Computer Science 864, pages 363–379. Springer-Verlag, Berlin, 1994.

[Mon95a] B. Monsuez. System *f* and abstract interpretation. In A. Mycroft, editor, *Proceedings of the Static Analysis Symposium, SAS'95*, Glasgow, Scotland, September 1995, Lecture Notes in Computer Science 983, pages 279–295. Springer-Verlag, Berlin, 1995.

[Mon95b] B. Monsuez. Using abstract insterpretation to define a strictness type inference system. In *Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation, PEPM'95*, La Jolla, California, pages 122–133. ACM Press, New York, 21–23 June 1995.

[NCD+96a] F. Nielson, P. Cousot, M.F. Dam, P. Degano, P. Jouvelot, A. Mycroft, and B. Thomsen. Logical and operational methods in the analysis of programs and systems. In M. Dam, editor, *Analysis and Verification of Multiple-Agent Languages, 5th LOMAPS Workshop*, Stockhlom, Sweden, 24–26 June 1996, Lecture Notes in Computer Science 1192, pages 1–21. Springer Verlag, Berlin, 1996.

[Vé95] F. Védrine. Binding-time analysis and strictness analysis by abstract interpretation. In A. Mycroft, editor, *Proceedings of the Static Analysis Symposium, SAS'95*, Glasgow, Scotland, September 1995, Lecture Notes in Computer Science 983, pages 400–417. Springer-Verlag, Berlin, 1995.

Autres conférences

[CC94b] P. Cousot and R. Cousot. Higher-order abstract interpretation. Seminar, Carnegie Mellon University, Pittsburgh, U.S.A., 14 September 1994.

[CC95b] P. Cousot and R. Cousot. Equivalent presentations of compositional inductive semantic definitions. In *Semantique II meeting*, Glasgow, Scotland, 28–29 September 1995.

[CC96a] P. Cousot and R. Cousot. Abstract symbolic model checking. LOMAPS meeting, Stockholm, Suède, 30 April 1996.

[Cou] P. Cousot. Sémantique, preuve et interprétation abstraite, une première introduction. Séminaire de l'École polytechnique, 12 avril 1995 & ENSTB, 25 janvier 1996.

[Cou95e] P. Cousot. Constructing a hierarchy of semantics by abstract interpretation. In *Workshop on Logic, Domains and Programming Languages*, Darmstadt, Germany, 24–27 May 1995.

[Cou95f] P. Cousot. Natural semantics trees in G^{∞}SOS style. In *Semantique II meeting*, Glasgow, Écosse, 28–29 September 1995.

[Cou96b] P. Cousot. Assigning types to programs by abstract interpretation. Seminario, Dip. di Informatica, Univ. di Pisa, Pise, Italie, 11 September 1996.

[Cou96c] P. Cousot. Corso di interpretazione astratta. Dottorato di Ricerca, Dip. di Informatica, Univ. di Pisa, Pise, Italie, 9–12 September 1996.

[Cou96e] P. Cousot. Interprétation abstraite. École jeunes chercheurs en programmation, GDR Programmation du CNRS, LaBRI, Bordeaux, 31 mars – 5 avril 1996, 4 April 1996.

[Cou96f] P. Cousot. Introductory course on abstract interpretation. Eighth European Summer School in Logic, Language, and Information ESSLLI'96, Charles University & Czech Technical University, Prague, République Tchéque, 12–26 August 1996.

[Cou96h] P. Cousot. Vérification et interprétation abstraite. MOdélisation et VErification des Processus Paralléles, MOVEP'96, École Centrale de Nantes, Nantes, 18–21 juin 1996, 21 June 1996.

[Cou97a] P. Cousot. Analyse sémantique de programmes par interprétation abstraite. École jeunes chercheurs en programmation, GDR Programmation du CNRS, INRIA, Sophia-Antipolis, 17 au 28 mars 1997, 20 March 1997.

Thèses

[Gou95b] É. Goubault. La géométrie du parallélisme. Thèse de doctorat de l'École Polytechnique en informatique, École Polytechnique, Palaiseau, France, 13 November 1995.

[Mon94b] B. Monsuez. Typage par interprétation abstraite. Thèse de doctorat de l'École Polytechnique en informatique, 'Ecole Polytechnique, Palaiseau, France, 13 February 1994.

Notes de cours

[Cou95b] P. Cousot. Calcul parallèle. Notes de cours, M.M.F.A.I., École Normale Supérieure, March 1995. 70 p.

[Cou95h] P. Cousot. Système d'exploitation Unix et réseaux d'ordinateurs. Notes de cours, École Polytechnique, January 1995. 306 p.

[Cou97b] P. Cousot. A course on abstract interpretation. Indo-French School on Abstract Interpretation, JNCASR, Bangalore, Inde, 14–19 April 1997.

[Cou97d] P. Cousot. Langages de programmation et compilation. Notes de cours, M.M.F.A.I., École Normale Supérieure, March 1997. 157 p.

Articles soumis ou en préparation

[BP97] C. Bodei and C. Priami. True concurrency via abstract interpretation. Soumis pour publication, 1997.

[Pri97] C. Priami. Stochastic analysis of mobile telephony networks. Soumis pour publication, 1997.

Rapports du DMI

[Gou95a] É. Goubault. Higher dimensional automata, part I – basic definitions. Research report, Laboratoire d'Informatique, École Normale Supérieure, Paris, France, 1995.

[Gou97a] É. Goubault. The dynamics of wait-free distributed computations. Technical report, École Normale Supérieure, 1997.

[Mon94a] B. Monsuez. Polymorphic typing of heterogeneous lists. LOMAPS Research Report LOMAPS-ENSX-5, École Normale Supérieure, June 1994.

Rapports de recherche publiés dans d'autres laboratoires

[CC94a] P. Cousot and R. Cousot. Forward and backward strictness analysis by abstract interpretation of a relational semantics. Research Report LIX/RR/94/05, Laboratoire d'Informatique, École Polytechnique, Palaiseau, France, February 1994.

[CCME95] P. Cousot, R. Cousot, Organizors Mycroft, A., and Editors. Report on Dagsthul seminar 9535 on abstract interpretation. Sarrebrücken, Germany, 28 August – 1 September 1995.

[V94] F. Védrine. Galois connection based abstract interpretation for binding time analysis. Rapport de stage, DEA "Informatique, Mathématiques et Applications", École Polytechnique, École Normale Supérieure, Universités de Paris 6, 7 & 11, France, 13 July 1994.

Rapports non publiés

[CC97a] P. Cousot and R. Cousot. Grammar analysis by abstract interpretation. Technical report, LIENS, June 1997.

Rapports de contrats

[CC96b] P. Cousot and R. Cousot. Atlantique ppr 3, ens-x, paris, sep. 95 – dec. 96. Technical report, DIKU, Copenhague, Danemark, 23 December 1996.

[CGH96] P. Cousot, J. Goubault, and N. Halbwachs. Techniques symboliques pour l'analyse statique et la vérification de systèmes,. Rapport final du contrat mrt "informatique 92", BULL/IMAG/LIENS, November 1993 – November1996, November 1996.

[NCD+95] F. Nielson, P. Cousot, M.F. Dam, P. Degano, P. Jouvelot, A. Mycroft, and B. Thomsen. Lomaps scientific overview and periodic progess report 8130-2, december 1994–november 1995. Technical report, DAIMI, Computer Science Department, Aarhus University, Aarhus, Danemark, December 1995.

[NCD+96b] F. Nielson, P. Cousot, M.F. Dam, P. Degano, P. Jouvelot, A. Mycroft, and B. Thomsen. Lomaps scientific overview and periodic progess report 8130-3, december 1995–november 1996. Technical report, DAIMI, Computer Science Department, Aarhus University, Aarhus, Danemark, December 1996.