0.1 Equipes de recherche

0.1.1 Équipe Sémantique et Interprétation Abstraite

- Responsable : Patrick Cousot (Professeur à l'École Normale Supérieure/Université de Paris 9) ;
- Chercheurs :

  Philippe Granger (Chercheur contractuel Esprit BRA 6809 Sémantique II, parti en septembre 1993 professeur à l'ENSTA),

  Klaus Havelund (Boursier HCM, parti en septembre 1995),

  Bruno Monsuez (Chercheur contractuel Esprit BRA 8130 LOMAPS) ;
- Doctorants :

  Régis Cridlig, Ingénieur de l'armement-recherche,

  Éric Goubault, Ingénieur des mines,

  Laurent Mauborgne, Élève de l'École Normale Supérieure,

  Franck Védrine, Élève de l'École Normale Supérieure.

Part I

Activité scientifique des équipes

Chapter 1

Équipe Sémantique et Interprétation Abstraite

L'équipe s'est constituée en octobre 1991 à l'occasion du recrutement comme professeur de Patrick Cousot. Elle conserve des liens privilégiés avec l'équipe "Sémantique, preuve et interprétation abstraite" du LIX, qu'il a créée et dirigée jusqu'au 1er octobre 1991, au travers d'un séminaire hebdomadaire.

1.1 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 approximation et les modèles géométriques 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 programme (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), 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.

1.2 Sémantiques

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

Si toutes les méthodes de définition de sémantiques des langages de programmtion 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 [19, 11, 14], ce qui permet alors de montrer l'équivalence de ces divers modes de présentation [50, 8]. Ces présentations de sémantiques étant conservées par interprétation abstraite [50, 8], 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 [19, 11, 14] qui diffèrent seulement par la précision de la description de l'exécution et le mode de présentation. Par exemple [38] 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 [9]. 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.

1.2.2 Méthodes géométriques pour le parallélisme

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. 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). 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.

On voit déjà dans cet exemple 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 deadlocks, les branchements et confluences, et donc les équivalences sémantiques "branching-time" [24], l'ordonnancement des actions... Tout ceci peut se traiter algébriquement de façon élégante en utilisant des outils de topologie algébrique et d'algèbre homologique [25]. 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. Il en découle naturellement des algorithmes qui devrons être expérimentés dans un avenir proche. D'autre part, les moyens de calculs inventés pour la topologie algébrique (suites exactes, suites spectrales...) prennent un tour tout particulier qu'il est intéressant d'examiner. Cela est d'autant plus vrai que les objets que nous étudions sont quelque peu différents de leurs homologues mathématiques classiques.

Tout ceci ne serait pas complet si on ne démontrait pas que l'on peut donner des sémantiques de languages dans les automates de dimension supérieure. Nous avons prouvé qu'il est possible de donner des sémantiques à la SOS ainsi que des sémantiques catégoriques compositionnelles [25, 21]. Les automates de dimension supérieure fournissent également un modèle de la logique linéaire intuitioniste. D'autres logiques (temporelles, ...) peuvent être utilisées en liaison avec ce modèle et devraient fournir des moyens de spécification/vérification de programmes parallèles (éventuellement par systèmes de types).

Enfin, le caractère géométrique du modèle des automates de dimension supérieure incite à des généralisations intéressantes telles les automates décrivant les systèmes temps réel [46], et les systèmes probabilistes. Nous n'avons encore qu'effleuré ces sujets, mais nous avons déjà montré comment obtenir des sémantiques opérationnelles vraiment parallèles, temps réel (éventuellement continu). 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.

1.3 Interprétation abstraite

1.3.1 Fondements de l'interprétation abstraite

La correspondance entre les propriétés concrètes et abstraites s'établit traditionnellement par une correspondance de Galois [13]. Ceci correspond au cas où toute propriété concrète a une meilleure approximation dans le domaine de propriétés abstraites. Cette propriété est très intéressante car la sémantique concrète et l'approximation définie par la correspondance de Galois définit entièrement la sémantique qui se déduit constructivement de la sémantique concète 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 [4, 16]. Ceci conduit à définir des notions de complétude relatives à divers degrés d'abstraction [18].

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 résulats 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 r:'esolus grossièrement [35, 36, 5].

Comme évoqué précédemment, les techniques d'analyse sémantique de programmes ont été principalement présentées sont formes 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 [19] 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 [50].

1.3.2 Analyse sémantique de programmes

Domaines abstraits

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 congruences arithmétiques en interprétation abstraite Nous avons étudié les propriétés de congruence linéaire permettant de lier les valeurs entières ou rationnelles x_1, ..., x_n de variables $\mathtt{X}_1$, ..., $\mathtt{X}_n$ d'un programme sous la forme $a_1x_1 + \cdots + a_nx_n \equiv b[m]$ où a_1, ..., a_n, b and m sont des constantes déterminées automatiquement par l'analyse [27]. On peut ainsi exprimer indirectement des propriétés de régularité dans les programmes, par exemple dans une structure de données (en comptant une position dans cette structure) ou lors de communications sur un canal (en liant le rang de la communication à une propriété de l'objet transmis.

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 variantes 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 [55, 29]. Une application est prévue en collaboration avec N. Halbwachs en vérification de modules.

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 (introduit par N. Jones et S. Muchnick) ou sur celui de contraintes ensemblistes introduit 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 [20]. 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 [51, 20].

Dans cette ligne, une recherche sur la construction de nouvelles représentations pour des ensembles infinis en interprétation abstraite vient de commencer. Il s'agit de trouver des représentations efficaces, en particulier du point de vue du calcul de point fixe, à des ensembles typiquement définis par des grammaires, auxquelles on ajoute la possibilité d'avoir des intersections, des projections et des négations.

Combinaisons de domaines abstraits Il est intéressant de combiner des domaines abstraits pour décomposer une propriété complexe en éléments simples ou pour rassembler des analyses de programmes conçues indépendamment. Diverses études de réductions et complétions sont faites dans [47, 3, 17, 28].

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 un domaine abstrait c'est-à-dire une algèbre de types, ensuite un ensemble de règles logiques assignant à chaque expression du langage un type, enfin un algorithme de reconstruction des types calculant pour chaque expression le type le plus général. Classiquement cet algorithme est basé sur l'algorithme d'unification de Robinson. Une autre approche consiste à utiliser l'interprétation abstraite. Les travaux d'Alan Mycroft et de Neil Jones ont, en 1984, ouvert la voie. Cependant, certaines difficultés font que cette approche connaßt un relatif succès uniquement dans le typage de Prolog. Dans le domaine des langages fonctionnels, cette approche n'a connu,à quelques exceptions près, que peu de développements.

Initialement, nous nous étions fixé comme objectif d'étudier les parallèles entre les systèmes de règles d'inférence et les systèmes de types obtenus par interprétation abstraite. Cependant, au cours de l'évolution des travaux, il a nous a été possible de montrer que les systèmes d'inférence de types classiques pouvaient être reformulés en utilisant l'interprétation abstraite [39]. Il n'était dès lors plus possible de dire qu'une approche était meilleure que l'autre, puisqu'il nous a été possible de montrer que les résultats obtenus par les systèmes de types pouvaient être obtenus par interprétation abstraite. Ceci nous permet d'enrichir les théories de types existantes avec la théorie et les techniques de l'interprétation abstraite.

En effet, il nous a été possible de définir un type comme valeur abstraite de l'ensemble des valeurs que le programme peut être amené à manipuler. En fait, la définition n'est pas immédiate et nécessite de décomposer l'abstraction en deux abstractions distinctes. La première transformation consiste à abstraire l'ensemble des valeurs par des "types abstraits" dénotant une approximation supérieure de cet ensemble. Toutefois, étant donné la taille gigantesque du treillis de types que nous avons du définir précédemment, il a été nécessaire de nous restreindre à une approximation de ce dernier. La seconde approximation consiste donc à sélectionner certains sous-ensembles de ces types abstraits et d'effectuer une projection vers ces sous-ensembles. Dans ce modèle, à l'opposé de la démarche classique, la notion de "type principal" est caractérisée — c'est-à-dire la notion de type le moins restrictif pour l'expression donnée tout en assurant la correction de l'exécution — par un plus grand point fixe.

Si nous décidons de nous restreindre à l'algèbre de types de ML, nous pouvons démonter que le système d'inférence de types que nous obtenons en utilisant la méthode précédemment introduite est équivalent au système ML+ d'Alan Mycroft. Pour le système de types ML+, le problème d'inférence de types est indécidable. En conséquence, le calcul du plus grand point fixe ne termine pas systématiquement. Tout d'abord, nous avons développé des méthodes autorisant l'introduction d'un opérateur d'élargissement permettant d'accélérer la convergence de l'algorithme de typage et d'en assurer la terminaison. Finalement, nous avons exhibé toute une famille d'opérateurs d'élargissement soit modélisant le comportement de systèmes d'inférences déjà existants comme ceux de ML ou de Miranda, soit améliorant de façon substantielle les systèmes de types basés sur l'algèbre d'Hindley/Milner [31, 33, 32, 40].

De plus, il nous a été possible de nous caractériser des systèmes de types beaucoup plus puissants, comme les types fractionnels [30], les systèmes de types

de Coppo-Dezani ainsi que les système F de Girard par interprétation abstraite. Actuellement, nous cherchons à définir des opérateurs d'élargisemment nous permmettant d'exhiber un sous-ensemble non trivial décidable de ces deux systèmes de types.

Système d'analyses par règles d'inférences construites par interprétation abstraite Ayant démontré qu'il est possible de considérer un type comme une valeur abstraite particulière, il est maintenant possible d'introduire des "types" modélisant les propriétés normalement décrites par des modèles de valeurs absraites utilisées en interprétation abstraite. Pour l'instant, nous nous sommes intéressés aux propriétés utiles pour l'analyse de nécessité (strictness analysis, c'est-à-dire une analyse qui détermine les arguments d'une fonction dont l'evaluation immédiate ne change pas le comportement de la fonction) [34]. Ceci nous a permis de concevoir un système d'inférence des propriétés de nécessité pour des langages fonctionnels typés ou non typés d'ordre supérieur. Nous envisageons d'étendre le modèle aux propriétés utilisées pour l'analyse de temps de liaison (binding time analysis, c'est-à-dire une analyse qui détermine les parties d'un programme qui peuvent être évaluées au moment de la compilation).

Extension des systèmes de types Un intérêt d'utiliser l'interprétation abstraite afin de défininir des système de types est de pouvoir combiner l'analyse de types avec d'autres types d'analyse. La combinaison de notre analyse avec les interprétations abstraites d'ensembles 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 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 deréception du programme. Ceci nous permet d'inférer les divers types des valeurs reçues à partir des types des valeurs envoyées. De même, nous avons développé un système de types pour les listes hétérogènes, c'est-à-dire des listes comprenant des valeurs de types différents.

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 considér'ees pour les objets manipulés par le programme.

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 [37]

et opérationnelles étendues à des comportements infinis [19]. 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 [7].

Analyse de projection Ces recherches formalisent l'analyse par projections de J. Hughes et P. Wadler dans un cadre d'interprétation abstraite, la comparent à l'analyse par idéaux et l'appliquent à un langage de premier ordre avec des types de données récursifs [53].

Le treillis abstrait de projections qui a été défini présente la particularité d'être disjonctif et non-uniforme : sont donc bien représentées les propriétés respectivement des fonctions à plusieurs arguments et du contenu des structures de donnés récursives comme les listes. Un opérateur d'élargissement est utilisé pour obtenir des classes de propriétés uniformes.

Analyse de comportement par interprétation abstraite L'analyse de comportement, introduite dans [7], 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).

L'analyse de nécessité peut être obtenue par interprétation abstraite de programmes fonctionnels basée sur une sémantique par réduction de graphes [54, 2, 1] ou relationnelles [6, 49]. Un problème récurrent 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 [55, 29].

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 [7] 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) [56]. 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 iterés abstraits sont également valables.

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) et la combinaison des analyses descendantes et ascendantes (les analyses classiques utilisent l'une ou l'autre mais pas les deux) [3, 15, 17].

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. Cette dimension décrit le degré de parallélisme effectif lors de l'exécution d'opérations concurrentes.

On peut définir diverses sortes de systèmes de transitions de dimension supérieure. Les systèmes totaux 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 permettent de plus de modéliser l'interférence entre plusieurs actions parallèles.

Programmes parallèles communicants 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, Concurrent Pascal, CML) et une propriété de correction sémantique de l'interprétation abstraite a été prouvée [21]. 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 un langage à mémoire partagée comme Concurrent Pascal. Il suffit alors de donner les règles définissant les transitions petit-pas autorisées entre deux configurations. Les transitions de dimension supérieure se construisent aisément à l'aide des transitions de dimension plus basse.

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.

En ce moment, une implémentation d'un prototype d'un analyseur de Concurrent Pascal basée sur une telle sémantique est en cours. Certaines techniques d'implémentation comme l'utilisation de transitions d'un grain plus grossier que dans les définitions sémantiques (Virtual Coarsening) permettent d'envisager l'interprétation abstraite précise de programmes de taille raisonnable.

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 interblocages (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.

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 est 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 [23].

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 [26]. Tout ceci n'a pas encore été fait en toute généralité et se trouve donner des idées d'algorithmes d'analyse statique par interprétation abstraite qui doivent encore être développés, pour aider à la vérification de programmes parallèles ainsi que pour améliorer les performances de certains compilateurs. Nous désirerons également développer un lien avec les travaux (méthodes géométriques) de M. Herlihy, N. Shavit, M. Saks, H. Attiya... sur les protocoles de systèmes distribués, et ceux de J. Gunawardena sur les bases de données parallèles. Nous sommes proches d'un traitement sémantique de ces questions, en utilisant le modèle des automates de dimension supérieure et de bonnes notions géométriques.

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).

Part II

Eléments d'appréciation de l'activité du laboratoire

Chapter 2

Collaborations

2.1 Sémantique et Interprétation Abstraite

- Projet "Interprétation abstraite et parallélisme" du GDR C^3 (Communication, Concurrence et Coopération), 1991–93 ;
- Contrat "Action Informatique 92" du MRT, *Techniques symbolique pour l'analyse statique et la vérification des systèmes*, BULL (J. Goubault), IMAG (N. Halbwachs), LIENS (P. Cousot), novembre 1992 à mai 1996.
- Projet "Sémantique, interprétation abstraite et parallélisme" du GDR PRS (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) depuis juillet 1992.
- 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) depuis décembre 1993.
- Projet NSF-ESPRIT "Atlantique" liant l'École Normale Supérieure (P. Cousot), l'École Polytechnique (R. Cousot), l'Université d'Århus (F. Nielson & H. Nielon), 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) depuis janvier 1993.
- 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, cordinateur), 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 3

Missions, conférences et séminaires

3.1 Sémantique et Interprétation Abstraite

- Invitations et séjours à l'étranger :
  - R. Cridlig : stage de DEA à l'Université de Cambridge (mars–juillet 1991), invitation au laboratoire d'informatique de l'Imperial College à Londres en avril 1993 ;
  - É. Goubault : : stage de DEA à l'Imperial College(mars–juillet 1991).
- Participation à des conférences :
  - P. Cousot : ICLP'91 (Paris, juin 1991), JTASPEFL'91 (Bordeaux, octobre 1991), POPL'92 (Albuquerque, Nouveau-Mexique, janvier 1992), PLILP'92 (Louvain, Belgique, août 1992), WSA'92 (Bordeaux, septembre 1992), Int. Kolloquium, Sonderforschungsbereich 124, VLSI – Entwurfsmethoden und Parallelität (Sarrebrücken, Allemagne, septembre 1993), WSA'93 (Padoue, septembre 1993), Mathematics of Programming Workshop on Galois Connections (Utrecht, Pays-Bas, septembre 1993), MASK Meeting (Koblenz, Allemagne, octobre 1993), ICCL'94 (Toulouse, mai 1994), SWAILP'95 (Eilat, juin 1995), FPCA'95 (La Jolla, Californie, juin 1995), CAV'95 (Liège, juillet 1995), GULP-PRODE (Salerno, septembre 1995) ;
  - R. Cridlig : ML Workshop et LFP (San Francisco, juin 1992), WSA'92 (Bordeaux, septembre 1992), FPCA'93 (Copenhague, juin 1993, démonstration du compilateur Camlot), WSA'93 (Padoue, septembre 1993) ; JFLA (Noirmoutier, janvier 1994), PEPM'95 (La Jolla, Californie, 1995) ;
  - É. Goubault : SemaGraph'91 (Nijmegen, 1991), Concur'92 (New York, 1992), Concur'93 (Hildesheim, 1993), WSA'93 (Padoue, 1993), PEPM'95 (La Jolla, Californie, 1995) ;
  - L. Mauborgne : SAS'94 (Namur, 1994) ;
  - B. Monsuez : JTASPEFL'91 (Bordeaux, octobre 1991), WSA'92 (Bordeaux, septembre 1992), FST & TCS (New Delhi, décembre 1992), FMP'93 (Novossibirsk, juillet 1993), WSA'93 (Padoue, septembre 1993), SAS'94 (Namur, septembre 1994), PEPM'95 (La Jolla, Californie, juin 1995).
- Séminaires :

- P. Cousot : Université de Cambridge (juin 1991), Université de Padoue (décembre 1992), Carnegie Mellon University (septembre 1994), École Polytechnique (avril 1995) ;
- R. Cridlig : INRIA Rocquencourt (octobre 1991), Université de Cambridge (juin 1992), École Normale Supérieure (février 1993, décembre 1993 et mars 1995), CRIN, Nancy (mars 1994) ;
- É. Goubault : École Normale Supérieure (1992, 1993, 1994, 1995), École des Mines de Paris (1992), Imperial College, London (1992, 1993), CRIN, Nancy (1994) ;
- L. Mauborgne : École Normale Supérieure (1994) ;
- B. Monsuez : Université de Cambridge (juin 1992), École Normale Supérieure (octobre 1992, mars 1993, juin 1993, octobre 1993, février 1995), Université de Nancy (avril 1993), LRI, Université Paris XI (avril 1994) ;
- F. Védrine : École Normale Supérieure (1995) ;

Chapter 4

Accueil de chercheurs

4.1 Sémantique et Interprétation Abstraite

Professeurs et directeurs de recherche invités

- Neil Jones, un mois, comme professeur invité de l'École Normale Supérieure en novembre 1993 ;
- Leslie Lamport, un mois, comme professeur invité de l'École Normale Supérieure en septembre 1995.

Chapter 5

Diffusion de la connaissance

5.1 Sémantique et Interprétation Abstraite

- Organisation à l'École Normale Supérieure, d'un séminaire régulier "sémantique et interprétation abstraite" où les orateurs É. Badouel (IRISA), D. Bolignano (BULL), F. Bourdoncle (DEC-PRL), P. Codognet (INRIA-Rocquencourt), M.-M. Corsini (LABRI, Bordeaux), R. Cridlig (LIENS), M. Debabi (BULL), A. Deutsch (INRIA, Rocquencourt), F. Fages (LIENS), J.-C. Fernandez (LGI, Grenoble), É. Goubault (LIENS), J. Goubault (BULL), P. Granger (ENSTA), K. Havelund (LIENS), S. Holzbacher (LIX), J.-L. Imbert (GIA, Marseille), T. Jensen (LIX), P. Jouvelot (CRI-ENSMP), C. Lecoutre (LIFL, Lille), R. Lissajoux (Thomson-LCR), F. Masdupuy (LIX/École des Mines), D. Méry (CRIN & INRIA Lorraine), B. Monsuez (LIENS), X. Nicollin (VERIMAG, Grenoble), S. Pinchinat (LIFIA, Grenoble), P. Quinton (IRISA, Rennes), A. Rauzy (LABRI, Bordeaux), D. Rémy (INRIA Rocquencourt), F. Ruget (CHORUS), J. Sifakis (VÉRIMAG, Grenoble), F. Védrine (LIENS) ont été principalement des jeunes chercheurs et des chercheurs étrangers de passage N. Benton (Cambridge University), C. Colby (Carnegie-Mellon University), M. Gabbrielli (CWI, Amsterdam), R. Giacobazzi (Université de Pise), R. Heckmann (Univ. Saarbrücken), N. Heintze (Carnegie Mellon University), N. D. Jones (DIKU), P. Iyer (N.C. State Univ./ENS Cachan), A. Kfoury (Boston University), L. Lamport (DEC-SRC, Palo Alto), B. Le Charlier (F.U.N.D.P., Namur), G. Levi (Université de Pise), I. Mackie (Imperial College), J. Muylaert-Filho (Imperial College), A. Mycroft (Cambridge University), F. Nielson (DAIMI, Århus), D. Sands (DIKU, Copenhague) ;

- Organisation de réunions de groupes de travail :

  - P. Cousot a organisé la réunion du groupe de travail 2.3 de l'IFIP à Pouilly en Auxois (octobre 1991) et a participé à la réunion du groupe de travail 2.4 de Madrid (mai 1992) ;
  - P. Cousot a organisé la réunion du projet Esprit BRA 3124 "Sémantique" au Mont Saint Michel (novembre 1991), participation de R. Cridlig, É. Goubault et B. Monsuez ;
  - É. Goubault et B. Monsuez ont organisé la réunion du projet Esprit BRA 6809 "Lomaps" à l'École Normale Supérieure en décembre 1994, participation de R. Cridlig, P. Cousot, L. Mauborgne et et F. Védrine.

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

- P. Cousot a participé au séminaire “Les problèmes nouveaux en programmation” organisé par l’institut d’expertise et de prospective de École Normale Supérieure en septembre 1991 ;
- P. Cousot, R. Cridlig, É. Goubault et B. Monsuez ont participé aux réunions du groupe de travail Esprit BRA 6128 “Sémantique” à Nice (novembre 1991) dans l’Îßle de Barra (Écosse, juin 1992), avec L. Mauborgne à Londres (juillet 1993) et avec F. Védrine à celle d’Århus (juillet 1994) ;
- B. Monsuez a participé à la réunion préparatoire du projet Esprit BRA 6809 “Lomaps” à Århus en ctobre 1993 ;
- R. Cridlig, É. Goubault, B. Monsuez, L. Mauborgne et F. Védrine ont participé à la réunion du projet Esprit BRA 6809 “Lomaps” à Fontainebleau en mai 1994 ;
- 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 ;
- É. Goubault a participé aux groupes de travail “Topologie Algébrique” (ENS, 1992) et “K-Théorie Topologique” (ENS, 1993) ;

Chapter 6

Réalisation et diffusion de logiciels, brevets

6.1 Sémantique et Interprétation Abstraite

6.1.1 Brevets

É. Goubault est le premier inventeur des brevets européens numéro 91480054.5 "A vector quantizing method for coding signals and system for implementing said method" (en date du 29/03/91 déposé pour IBM France, puis breveté aux Etats-Unis) et numéro 92480097.2 "Improved method for coding digital data using vector quantizing techniques and device for implementing said method" (en date du 30/06/92 déposé pour IBM France).

6.1.2 Réalisation et diffusion de logiciels

Compilation des langages fonctionnels de Cridlig Ce travail commencé dans l'équipe ICSLA de l'INRIA-Rocquencourt a débouché sur la réalisation effective d'un compilateur optimisant et portable pour le langage Caml-Light (un des deux dialectes dominants de ML). Ce compilateur Camlot a été distribué sur le réseau Internet et est utilisé par bon nombre de développeurs Caml-Light [22].

Ce compilateur inaugure une nouvelle voie dans la compilation des langages fonctionnels. En effet, il traduit le flot de contrôle d'un langage fonctionnel d'ordre supérieur vers les constructions efficaces d'un langage impératif du premier ordre, en l'occurrence le langage C. Cette traduction est effectuée *directement* de fonction ML à fonction C. Les valeurs fonctionnelles sont traduites en fermetures, et une analyse statique est effectuée pour déterminer les variables liées à passer dans les environnements au cours de l'exécution. Une analyse similaire permet de décider si une fonction ML peut être intégrée à une fonction C ou bien si elle doit être déplacée au niveau supérieur (toplevel). Quant aux structures de données dynamiques de ML, elles sont allouées dans le tas C par un gestionnaire de pages mémoires et récupérées automatiquement grâce à un ramasse-miettes (garbage collector) conservatif.

Chapter 7

Évaluation de la recherche

7.1 Sémantique et Interprétation Abstraite

- P. Cousot :
  - membre du comité de rédaction de la revue "Science of Computer programming" ;
  - président du comité de programme de WSA'93 (Padoue) ;
  - membre des comités de programme de ICCL'92, PEPM'93, FMPA'93, ICCL'94, SAS'94, PLILP'95, SAS'95 ;
  - 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), Universités de Cambridge, Melbourne, Sarrebrück ;
  - 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, Ministère de l'Industrie et du Commerce Extérieur (appel à propositions "Informatique 92") ;
  - expertise (reviewer) de projets européens ESPRIT II et III :
    - Esprit B.R.A. project 3070 SEMAGRAPH (M. Sleep (coordinateur), Londres, 2–3 septembre 1991 ; Paris, 16–17 novembre 1992),
    - Esprit B.R.A. project 3096 SPEC (W. De Roever (coordinateur), Sigtuna, Suède, 4–5 septembre 1991 ; Grenoble, 20–22 mai 1992),
    - Esprit B.R.A. project 6021 REACT (P. Wolper (coordinateur), Héraklion, Crète, 5–6 juin 1993 ; Liège, Belgique, 8 juillet 1994),
    - Esprit B.R.A. project 7071 PROCOS II (C. Hoare (coordinateur), Bruxelles, 24 septembre 1993 & 30 septembre 1994) ;
  - évaluateur (referee) pour les revues Acta Informatica, Information Processing Letters, Science of Computer Programming, TOPLAS et les conférences ICCL'92, FMPA'93, WSA'93, ESOP'94, ICCL'94, ILPS'94, SAS'94, PEPM'94, POPL'95, PLILP'95, SAS'95 ;

- R. Cridlig :
  - évaluateur (referee) pour les conférences SAS'94, PLILP'95, SAS'95 ;
- É. Goubault :
  - évaluateur (referee) pour les revues Fundamenta Informaticae (1993), Information Processing Letters (1994), Theoretical Computer Science (1994) et les conférences SAS'94, STACS'95, PEPM'95, PLILP'95, CONCUR'95 ;
- B. Monsuez :
  - membre du conseil de Laboratoire du LIENS.
  - évaluateur (referee) pour les conférences SAS'94, PLILP'95, SAS'95 ;

Chapter 8

Encadrement doctoral

8.1 Direction de thèses

Nous donnons uniquement la liste des thèses soutenues entre 1991 et 1995 et dirigées par un membre du laboratoire.

- Patrick Cousot
  - P. Granger, École Polytechnique, soutenue en juillet 1991
    Analyse sémantique de congruence
  - O. Mallet, École Polytechnique, soutenue en juin 1992
    Interprétation abstraite appliquée à la compilation et la parallélisation en programmation logique
  - François Bourdoncle, École Polytechnique, soutenue en novembre 1992
    Sémantiques des langages impératifs d'ordre supérieur et interprétation abstraite
  - Jean Goubault, École Polytechnique, soutenue en septembre 1993
    Démonstration automatique en logique classique : complexité et méthodes,
  - F. Masdupuy, École Polytechnique, soutenue en septembre 1993
    Analyse sémantique relationnelle des indices de tableaux par congruences et trapézoßdes rationnels
  - B. Monsuez, École Polytechnique, soutenue en février 1994
    Typage par interprétation abstraite

8.2 Participation à d'autres jurys de thèses

- P. Cousot : 17 jurys (dont 5 habilitations), 4 présidences, 7 rapports.

Chapter 9

Enseignement

9.1 2ème cycle

9.1.1 En France

- Magistère de mathématiques fondamentales et appliquées et informatique de l'ENS (MMFAI) : M. Bidoit, Y. Caseau, G. Castagna, G. Cousineau, P. Cousot, R. Cridlig, R. DiCosmo, F. Fages, M. Fernandez, É. Goubault, A. Joux, M. Laurent, P. Matherat, M. Pocchiola, J. Stern, S. Vaudenay, F. Védrine.
- École polytechnique : P. Cousot, R. Cridlig, F. Fages, É. Goubault, L. Mauborgne, B. Monsuez, J. Stern, S. Vaudenay.
- ENSTA 1ère et 2ème année : R. Cridlig, É. Goubault.
- Encadrement de deux stagiaires de l'ENSTA 1ère année en février 1995 (É. Goubault, R. Cridlig).

9.2 3ème cycle

9.2.1 En France

- DEA IMA, filière programmation, sémantique et preuve : P. Cousot, P-L. Curien, É. Goubault, B. Monsuez, G. Longo.
- DEA d'Informatique d'Orsay, encadrement d'un stagiaire, 1993 (É. Goubault),
- DEA IMA, encadrement de stagiaires, 1991 (P. Cousot), 1993 (P. Cousot), 1994 (P. Cousot & B. Monsuez),

Chapter 10

Distinctions honorifiques

- P. Cousot : **chevalier dans l'ordre National du Mérite**, décret du 24 juin 1993 ;

10.1 Contrats

10.1.1 Équipe "Sémantique et interprétation abstraite"

Bibliography

Livres

[1] Goubault (É.), Hankin (C.), van Eekelen (M.) et Nocker (E.). – Abstract reduction: towards a theory via abstract interpretation. *In : Term Graph Rewriting: Theory and Practice*, éd. par Sleep (R.), Plasmeijer (R.) et van Eekelen (M.), chap. 9, pp. 117–129. – John Wiley & Sons, Ltd., 1993.

[2] Goubault (É.) et Hankin (C.). – A lattice for the abstract interpretation of term graph rewriting systems. *In : Term Graph Rewriting: Theory and Practice*, éd. par Sleep (R.), Plasmeijer (R.) et van Eekelen (M.), chap. 10, pp. 131–140. – John Wiley & Sons, Ltd., 1993.

Articles invités

[3] Cousot (P.) et Cousot (R.). – Abstract interpretation and application to logic programs. *Journal of Logic Programming*, vol. 13, n° 2–3, 1992, pp. 103–179.

[4] Cousot (P.) et Cousot (R.). – Abstract interpretation frameworks. *Journal of Logic and Computation*, vol. 2, n° 4, août 1992, pp. 511–547.

[5] Cousot (P.) et Cousot (R.). – Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, papier invité. *In : Programming Language Implementation and Logic Programming, Proceedings of the Fourth International Symposium, PLILP'92*, éd. par Bruynooghe (M.) et Wirsing (M.), pp. 269–295. – Springer-Verlag, Berlin, Allemagne, 1992.

[6] Cousot (P.) et Cousot (R.). – Galois connection based abstract interpretations for strictness analysis, papier invité. *In : Proceedings of the International Conference on Formal Methods in Programming and their Applications*, éd. par rner (D. Bjø), Broy (M.) et Pottosin (I.V.), pp. 98–127. – Springer-Verlag, Berlin, Allemagne, juin 28–juillet 2, 1993.

[7] Cousot (P.) et Cousot (R.). – Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), papier invité. *In : Proceedings of the 1994 International Conference on Computer Languages*. pp. 95–112. – IEEE Computer Society Press, Los Alamitos, Californie, U.S.A., mai 16–19, 1994.

[8] * Cousot (P.) et Cousot (R.). – Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form, papier invité. *In : Computer Aided Verification, Proceedings of the 7th International Conference, CAV'95*, éd. par Wolper (P.). – Springer-Verlag, Berlin, Allemagne, juillet 1995. À paraβtre.

Articles dans des revues internationales avec comité de lecture

[9] Cousot (P.) et Cousot (R.). – "A la Burstall" intermittent assertions induction principles for proving inevitability properties of programs. *Theoretical Computer Science*, vol. 120, 1993, pp. 123–155.

[10] Goubault (É.) et Lebris (C.). – Groupes ext^1 et déformations de représentations de certaines algèbres de lie nilpotentes. *Bulletin des Sciences Mathématiques*, vol. $2^{\text{ème}}$ série, n° 116, 1992.

Conférences invitées

[11] Cousot (P.) et Cousot (R.). – Constructing a hierarchy of semantics of functional programs by abstract interpretation, conférence invitée. *In :* *Workshop on Static Analysis, WSA'92.* – Bordeaux, France, 23–25 septembre 1992.

[12] Cousot (P.) et Cousot (R.). – Abstract interpretation of parallel programs, conférence invitée. *In :* *Int. Kolloquium, Sonderforschungsbereich 124, VLSI – Entwurfsmethoden und Parallelität.* – Sarrebrücken, Allemagne, 2–3 septembre 1993.

[13] Cousot (P.) et Cousot (R.). – Galois connections and abstract interpretation, conférence invitée. *In :* *Mathematics of Programming Workshop on Galois Connections.* – Universiteit Utrecht, Pays-bas, 13–14 septembre 1993.

[14] Cousot (P.) et Cousot (R.). – Inductive definitions of semantics and connections between semantics as used in abstract interpretation, conférence invitée. *In :* *MASK Meeting.* – Koblenz, Allemagne, 6–8 octobre 1993.

[15] Cousot (P.). – Abstract interpretation of logic programs, invited advanced tutorial. *In : Proceedings of the 8^{th} International Conference on Logic Programs*, éd. par Furukawa (K.). p. 940. – Paris, France, 25–28 juin 1991.

[16] Cousot (P.). – A tutorial on abstract interpretation,. *In :* *1994 International Conference on Computer Languages.* – mai 16–19, 1994.

[17] Cousot (P.). – Combining bottom-up and top-down in abstract interpretation of logic languages, conférence invitée. *In :* *Special Workshop on Abstract Interpretation of Logic Languages.* – juin 18–19 1995.

[18] Cousot (P.). – Completeness in abstract interpretation, conférence invitée. *In : GULP-PRODE'95.* – septembre 1995.

Communications dans des conférences internationales avec comité de lecture

[19] Cousot (P.) et Cousot (R.). – Inductive definitions, semantics and abstract interpretation. *In :* *Conference Record of the 19^{th} Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages.* pp. 83–94. – Albuquerque, Nouveau Mexique, 1992.

[20] * Cousot (P.) et Cousot (R.). – Formal language, grammar and set-constraint-based program analysis by abstract interpretation. *In :* *Proceedings of the 7^{th} ACM Conference on Functional Programming Languages and Computer Architecture.* – La Jolla, Californie, juin 1995. À paraβtre.

[21] Cridlig (R.) et Goubault (É.). – Semantics and analysis of Linda-based languages. *In : Proceedings of the 3rd International Workshop WSA'93 on Static Analysis*, éd. par Cousot (P.), Falaschi (M.), Filé (G.) et Rauzy (A.), pp. 72–86. – Springer-Verlag, Berlin, Allemagne, septembre 22–24, 1993.

[22] Cridlig (R.). – An optimizing ML to C compiler. *In : ACM SIGPLAN Workshop on ML and its Applications*. pp. 28–36. – San Francisco, Californie, 20–21juin1992.

[23] * Cridlig (R.). – 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*. – ACM Press, New York, U.S.A., juin 1995. À paraβtre.

[24] Goubault (É.) et Jensen (T. P.). – Homology of higher dimensional automata. *In : CONCUR'92, Proceedings of the Third International Conference on Concurrency Theory*, éd. par Cleaveland (W. R.), pp. 254–268. – Springer-Verlag, Berlin, Allemagne, août 1992.

[25] Goubault (É.). – Domains of higher-dimensional automata. *In : CONCUR'93, Proceedings of the 4th International Conference on Concurrency Theory*, éd. par Best (E.). pp. 293–307. – Springer-Verlag, Berlin, Allemagne, août 1993.

[26] * Goubault (É.). – Schedulers as abstract interpretations of higher-dimensional automata. *In : Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation, PEPM'95*. – ACM Press, New York, U.S.A., juin 1995. À paraβtre.

[27] Granger (P.). – Static analysis of linear congruence equalities among variables of a program. *In : TAPSOFT'91, Proceedings of the International Joint Conference on Theory and Practice of Software Development*, éd. par Abramsky (S.) et Maibaum (T.S.E.), pp. 169–192. – Springer-Verlag, Berlin, Allemagne, 1991.

[28] Granger (P.). – Improving the results of static analyses of programs by local decreasing iterations. *In : Proceedings of the 12th Foundations of Software Technology and Theoretical Computer Science Conference*. pp. 68–79. – New Delhi, Inde, 18–20 décembre 1992, Lecture Notes in Computer Science 652, 1992.

[29] Mauborgne (L.). – Abstract interpretation using TDGs. *In : Proceedings of the Static Analysis Symposium, SAS'94*, éd. par Le Charlier (B.), pp. 363–379,. – Springer-Verlag, Berlin, Allemagne, 1994.

[30] Monsuez (B.). – Fractional types. *BIGRE, Analyse Statique WSA'92, Bordeaux,* IRISA, Rennes, France, vol. 81–82, 23–25 septembre 1992, pp. 274–284.

[31] Monsuez (B.). – Polymorphic typing by abstract interpretation. *In : Proceedings of the 12th Foundations of Software Technology and Theoretical Computer Science Conference*. pp. 127–138. – New Delhi, Inde, 18–20 décembre 1992, Lecture Notes in Computer Science 652, 1992.

[32] Monsuez (B.). – Polymorphic types and widening operators. *In : Proceedings of the 3rd International Workshop WSA'93 on Static Analysis*, éd. par Cousot (P.), Falaschi (M.), Filé (G.) et Rauzy (A.), pp. 267–281. – Springer-Verlag, Berlin, Allemagne, septembre 22–24, 1993.

[33] Monsuez (B.). – Polymorphic typing for call-by-name semantics. *In : Proceedings of the International Conference on Formal Methods in Programming and their Applications*, éd. par rner (D. Bjø), Broy (M.) et Pottosin (I.V.), pp. 156–169. – Springer-Verlag, Berlin, Allemagne, juin 28–juillet 2, 1993.

[34] * Monsuez (B.). – 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.* – ACM Press, New York, U.S.A., juin 1995. À paraβtre.

Autres conférences

[35] Cousot (P.) et Cousot (R.). – Comparison of the Galois connection and widening/narrowing approaches to abstract interpretation. JTASPEFL'91. *BIGRE*, vol. 74, octobre 1991, pp. 107–110.

[36] Cousot (P.) et Cousot (R.). – Comparison of the Galois connection and widening/narrowing approaches to abstract interpretation, conférence invitée. *In : ICLP'91 Pre-Conference Workshop on Semantics-based Analysis of Logic Programs.* – Paris, France, 15 mai 1991.

[37] Cousot (P.) et Cousot (R.). – Relational abstract interpretation of higher-order functional programs. JTASPEFL'91. *BIGRE*, vol. 74, octobre 1991, pp. 33–36.

[38] Cousot (P.). – Constructing a hierarchy of semantics by abstract interpretation. *In : Workshop on Logic, Domains and Programming Languages.* – mai 24–27 1995.

[39] Monsuez (B.). – An attempt to find polymorphic types by abstract interpretation. *BIGRE, Actes JTASPEFL'91, Bordeaux,* IRISA, Rennes, France, vol. 74, octobre 1991, pp. 18–26.

Thèses

[40] Monsuez (B.). – *Typage par Interprétation Abstraite.* – Palaiseau, France, Thèse de doctorat de l'école polytechnique en informatique, École Polytechnique, 13 février 1994.

Notes de cours

[41] Cousot (P.). – *Algorithmique et programmation en Pascal (cours).* – Ellipses, Paris, France, 1992, *Cours de l'École Polytechnique.* 297 p.

[42] Cousot (P.). – *Algorithmique et programmation en Pascal (exercices et corrigés).* – Ellipses, Paris, France, 1992, *Cours de l'École Polytechnique.* 271 p.

[43] Cousot (P.). – *Calcul parallèle.* – École Normale Supérieure, Notes de cours, M.M.F.A.I., mars 1995. 70 p.

[44] Cousot (P.). – *Langages de programmation et compilation.* – École Normale Supérieure, Notes de cours, M.M.F.A.I., mars 1995. 133 p.

[45] Cousot (P.). – *Système d'exploitation Unix et réseaux d'ordinateurs et compilation.* – École Polytechnique, Notes de cours, janvier 1995. 306 p.

Articles soumis ou en préparation

[46] Goubault (É.). – *Transitions take time.* – LIENS, École Normale Supérieure, Paris, France, Soumis pour publication, 1995.

Rapports du DMI

[47] Cousot (P.) et Cousot (R.). – *Abstract Interpretation and Application to Logic Programs.* – Rapport de recherche n° LIENS-92-12, École Normale Supérieure, Paris, France, Laboratoire d'Informatique, juin 1992.

[48] Goubault (É.). – *Higher Dimensional Automata, Part I – Basic Definitions.* – Rapport de recherche. à paraβtre, École Normale Supérieure, Paris, France, Laboratoire d'Informatique, 1995.

Rapports de recherche publiés dans d'autres laboratoires

[49] Cousot (P.) et Cousot (R.). – *Forward and Backward Strictness Analysis by Abstract Interpretation of a Relational Semantics.* – Rapport de recherche n° LIX/RR/94/05, École Polytechnique, Palaiseau, France, Laboratoire d'Informatique, février 1994.

[50] Cousot (P.) et Cousot (R.). – *Compositional and Inductive Semantical Definitions in Fixpoint, Equational, Constraint, Closure-condition, Rule-based and Game-Theoretic Form.* – Rapport de recherche n° LIX/RR/95/01, École Polytechnique, France, Laboratoire d'Informatique, janvier 1995.

[51] Cousot (P.) et Cousot (R.). – *Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation.* – Rapport de recherche n° LIX/RR/95/02, École Polytechnique, France, Laboratoire d'Informatique, janvier 1995.

[52] ← Cridlig (R.). – *Un compilateur optimisant pour le langage ML.* – Rapport d'option, Palaiseau France, École Polytechnique, juillet 1991.

[53] Cridlig (R.). – *Projection analysis by abstract interpretation.* – Rapport de stage, École Polytechnique, École Normale Supérieure, Universités de Paris 6, 7 & 11, France, DEA "Informatique, Mathématiques et Applications", septembre 1992.

[54] Goubault (É.). – *Interprétation abstraite de systèmes de réécriture de graphes de termes.* – Rapport de stage, École Polytechnique, École Normale Supérieure, Universités de Paris 6, 7 & 11, France, DEA "Informatique, Mathématiques et Applications", 11 septembre 1991.

[55] Mauborgne (L.). – *Abstract interpretation using TDGs.* – Rapport de stage, École Polytechnique, École Normale Supérieure, Universités de Paris 6, 7 & 11, France, DEA "Informatique, Mathématiques et Applications", septembre 1993.

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

Brevets

[57] Goubault (É.). – *A vector quantizing method for coding signals and system for implementing said method.* – Premier inventeur du brevet européen numéro 91480054.5 déposé pour IBM France, puis breveté aux États-Unis, 29 mars 1991.

[58] Goubault (É.). – *Improved method for coding digital data using vector quantizing techniques and device for implementing said method'.* – Premier inventeur du brevet européen numéro 92480097.2 déposé pour IBM France., 30 juin 1992.