Département d'Informatique de l'École Normale Supérieure

Rapport scientifique de l'équipe

Interprétation abstraite et sémantique

(1998-2001)

Membres de l'équipe

- Responsable : Patrick Cousot, professeur à l'ENS ;
- Autre membre permanent :

  Laurent Mauborgne, maître de conférences à l'ENS ;
- Post-doctorant :

  Arnaud Venet, 03/1999 – 07/1999 ;
- Doctorants :

  Jérôme Feret, élève normalien en 4ème année, doctorant depuis juin 2000 ;

  Antoine Miné, élève normalien en 4ème année, stagiaire puis doctorant depuis juin 2000 ;

David Monniaux, élève normalien de septembre 1998 à août 1999, assistant moniteur normalien à l'Université de Paris IX (Dauphine) depuis septembre 1999 ;

Frank Védrine, assistant moniteur normalien, 9/1997 – 9/2000.

Table des matières

Activité scientifique de l'équipe

1 Interprétation abstraite

L'*interprétation abstraite* est une théorie de l'approximation des structures mathématiques intervenant dans la définition de la sémantique des langages informatiques, qu'il s'agisse de langages de programmation ou de langages de spécification de ces systèmes informatiques. Diverses introductions sont proposées dans [3, 4, 14].

2 Conception de hiérarchies de sémantiques par interprétation abstraite

La *sémantique* d'un programme spécifie formellement les comportements possibles d'un système informatique exécutant ce programme en interaction avec un environnement quelconque. L'interprétation abstraite offre un point de vue constructif et unificateur sur les sémantiques des langages de programmation. En effet, en faisant varier le niveau d'observation de ces comportements à l'exécution, qui peut être plus ou moins précis, l'interprétation abstraite permet de construire des hiérarchies de sémantiques intégrant toutes les sortes de sémantiques existantes ainsi que de nouvelles variantes [9, 10]. La conception de hiérarchies de sémantiques pour des familles de langages de programmation sert de fondements pour concevoir une grande variété d'analyseurs statiques pour ces langages.

3 Analyse statique par interprétation abstraite

L'application la plus connue de l'interprétation abstraite est l'*analyse statique* [7]. Si l'approximation est suffisamment grossière, l'abstraction d'une sémantique peut permettre d'en donner une version moins précise mais calculable par l'ordinateur. De ce fait un ordinateur est capable d'analyser le comportement de programmes et de logiciels avant même de les exécuter. La

perte d'information ne permet pas de répondre à toutes les questions relatives à la sémantique concrète, mais toutes les réponses données par calcul effectif de la sémantique abstraite sont toujours justes. Les applications de l'analyse statique concernent la compilation optimisante et la transformation de programmes mais principalement la mise au point et la vérification des conditions de bon fonctionnement des systèmes informatiques (comme les systèmes critiques embarqués pour le contrat européen DAEDALUS).

4 Algèbres de propriétés abstraites

Il est encore et toujours nécessaire d'étendre la gamme des analyses disponibles pour permettre de choisir le meilleur compromis entre la précision de l'analyse et son coût, en particulier pour analyser de très grands programmes (de plusieurs centaines de milliers à quelques millions de lignes). Par conséquent, une activité essentielle de l'équipe sur l'analyse statique concerne la recherche de nouvelles approximations de classes de propriétés de programmes qui soient à la fois expressives et performantes.

Cette idée se formalise par des *algèbres abstraites* utilisées pour définir des sémantiques approchées de langages de programmation dont le calcul effectif permet de déterminer automatiquement et statiquement les propriétés dynamiques d'un programme.

Une algèbre abstraite comprend d'une part un *domaine abstrait* dont les éléments représentent les propriétés abstraites considérées par l'analyse. D'autre part, une algèbre abstraite comprend des *opérations abstraites* qui formalisent par exemple des approximations de transformateurs de prédicats pour les primitives des principaux langages de programmation. L'étude mathématique de la correspondance entre les algèbres de propriétés abstraites et la sémantique concrète doit être complétée par la conception de structures de données et d'opérations informatiques conduisant à une implantation en machine efficace. Ces algèbres abstraites peuvent être développées et étudiées indépendamment des langages de programmation et des analyses particulières.

Cette étude des algèbres de propriétés abstraites est particulièrement importante pour l'application pratique de l'interprétation abstraite, puisque le domaine abstrait et les structures de données utilisées pour représenter ses éléments vont déterminer à la fois la précision et la complexité des analyses. Pour une plus grande précision, les structure de données pour les domaines abstraits doivent permettre de représenter autant d'objets que possibles, et pour une meilleure complexité, ils doivent permettre de calculer efficacement les opérations abstraites associées, deux objectifs difficilement à conciliables. D'autre part, les analyses les plus fines utilisent en général la possibilité d'approximer certains de ces calculs de manière dynamique à l'aide d'opérateurs d'élargissement. Ces élargissements doivent donc aussi être facilités par les

structures de données des domaines abstraits.

Nous étudions également la façon de construire ces algèbres par composition d'algèbres plus simples. D'un point de vue informatique les algèbres abstraites s'implantent comme des modules et des foncteurs fournis à des analyseurs statiques génériques permettant de mettre en œuvre de nombreuses analyses et donc de réaliser de nombreuses expérimentations.

5 Algèbres de propriétés abstraites symboliques

Il s'agit de représenter des propriétés et donc des ensembles infinis d'objets informatiques symboliques (par exemple séquences, listes, arbres, graphes) eux-mêmes généralement infinis (puisqu'il faut bien prendre en compte le cas des programmes dont l'exécution ne termine pas).

En général, les structures de données classiques disponibles ne sont pas bien adaptées, et ce pour deux raisons : d'une part les opérations abstraites ne sont pas forcément celles pour lesquelles les représentations sont les plus efficaces, d'autre part les possibilités d'approximation et plus encore d'élargissement ne sont en général pas offertes. Il faut donc soit adapter les structures de données existantes en inventant des élargissements compatibles, soit inventer de nouvelles structures de données. Cette dernière voie permet d'utiliser, dès la conception des structures de données, les possibilités d'approximations sûres offertes par le cadre de l'interprétation abstraite.

5.1 Relations entre ensembles finis

Les analyses précises doivent tenir compte des liens entre les propriétés de différentes parties des programmes. Par exemple, on peut vouloir représenter les relations entre les propriétés d'entrée et celles de sortie pour fournir une analyse modulaire. Si les propriétés des différentes parties des programmes peuvent être finies, on peut utiliser des relations entre ensembles finis pour coder les liens entre les propriétés. L'avantage de cette approche, c'est qu'il existe une manière classique et efficace de les représenter, à condition que le nombre d'ensembles à relier soit lui-même fini. Il s'agit des Binary Decision Diagrams (BDDs), qui partagent autant que possible les sous-relations isomorphes. Ces représentations peuvent être adaptées à la représentation de fonctions d'ordre supérieur, et il est possible de définir des opérateurs d'élargissement basés sur leur taille. Ces extensions ont été appliquées avec succès à l'analyse de nécessité [13].

Lorsque les propriétés à analyser sont plus fines, comme les propriétés de terminaison ou les propriétés temporelles comme l'équité, il est nécessaire de relier un nombre infini d'ensembles. On peut dans ce cas utiliser une autre représentation classique, les automates de Büchi, mais leur complexité est trop grande vis-à-vis des opérations abstraites. Nous proposons donc une extension des BDDs, les graphes de décisions [27], qui gardent les propriétés

de partage des BDDs. Les graphes de décision sont plus efficaces que les automates de Büchi, au prix d'une légère perte d'expressivité. Cette nouvelle représentation permet d'exprimer des propriétés de vivacité ou d'équité, mais impose d'approximer les unions de relations.

5.2 Ensembles d'arbres

Les langages d'arbres permettent d'exprimer les propriétés les plus complexes, surtout lorsqu'ils contiennent des arbres infinis. On peut par exemple facilement exprimer des propriétés de traces d'exécutions concurrentes, de sécurité de protocoles crytpographiques ou encore d'état de la mémoire en cours d'exécution. Encore une fois, les représentations classiques, que ce soit par automates ou grammaires, ne sont pas adaptées à l'analyse de programmes.

Nous proposons une nouvelle représentation, fondée sur une décomposition canonique des ensembles d'arbres. Cette décomposition extrait d'une part la forme arborescente de l'ensemble, par un partage préfixe maximum, et d'autre part des relations entre les sous-arbres de cette forme arborescente [41]. La forme arborescente peut être représentée de manière très efficace par des squelettes d'arbres [28], qui partagent aussi naturellement les sous-arbres et ensembles de sous-arbres communs. Ces squelettes constituent une première approximation des propriétés, facile à manipuler et structurellement proche des ensembles d'arbres, ce qui facilite la mise au point d'opérateurs d'élargissement.

Ces représentations sont ensuite raffinées à l'aide de relations pour former des schémas d'arbres [29]. Le pouvoir expressif et la complexité de ces structures dépend du type de représentation utilisée pour les relations. En utilisant des relations finies, on peut déjà exprimer des langages hors de portée des automates classiques, comme $\{a^n b^n c^n \mid n \geq 0\}$. Avec les graphes de décision décrits plus haut, on peut facilement exprimer des preuves de terminaison sous hypothèse d'équité. La validation pratique de ces techniques devrait pouvoir être obtenue grâce à leur application dans le cadre du contrat européen DAEDALUS.

6 Algèbres de propriétés abstraites numériques

La majeure partie des programmes informatiques manipulent des variables numériques entières ou à virgule flottante. L'analyse des valeurs prises par ces variables est très importante car elle permet à elle seule de découvrir plusieurs types de bogues très répandus, comme les dépassements de tableau, les divisions par zéro ou les boucles infinies. On utilise pour cela des *domaines numériques abstraits*. Actuellement, parmi les domaines numériques abstraits existants et utilisés, on cite : le domaine des intervalles,

peu coûteux (coût linéaire) mais peu précis, et le domaine des polyèdres plus précis mais très coûteux (coût exponentiel).

Dans le but d'améliorer la granularité des analyses numériques, nous avons développé deux nouveaux domaines numériques abstraits : le domaine des *différences bornées* [30] et le domaine des *octogones* [58]. Ces domaines sont basés d'une part sur une structure de données qui a déjà fait ses preuves dans le domaine de la *vérification de modèles* (« *model-checking* ») : les Difference-Bound Matrices (DBMs) et d'autre part sur l'algorithmique des graphes pondérés. Le coût (coût cubique en temps et quadratique en mémoire) et la précision de ces domaines est intermédiaire entre ceux du domaine des intervalles et ceux des polyèdres. Un point important à noter est que ces domaines sont *relationnels*, c'est-à-dire aptes à capturer des relations numériques entre plusieurs variables, ce qui n'est pas le cas du domaine des intervalles.

En pratique, ces domaines abstraits se sont révélés un bon compromis et ils ont permis des analyses hors de portée du domaine des intervalles pour un coût beaucoup plus faible que le domaine des polyèdres.

7 Analyses statiques de systèmes mobiles

De vastes réseaux de communications sont utilisés pour répondre aux besoins de notre société. Ces derniers permettent de concevoir des systèmes de processus distants, dans lesquels plusieurs processus sont exécutés en différents sites d'un réseau et interagissent en communiquant. De nouveaux processus peuvent être créés dynamiquement. De plus, ce qui est plus récent, la distribution des processus peut varier au cours du temps : c'est la programmation mobile.

Il est très difficile de prouver que le comportement d'un très gros programme est conforme à une spécification donnée. C'est encore plus délicat dans le cadre de la programmation mobile. Pourtant, nous nous devons de montrer que les systèmes mobiles que nous utilisons sont sûrs. Ainsi, dans le cadre de la téléphonie mobile, une société qui utilise un réseau de processus mobiles doit être sûre que son réseau ne demandera pas d'utiliser plus de ressources qu'elle ne peut en fournir et que les contraintes de confidentialité sur les informations qui circulent sur ce réseau ne peuvent pas être violées.

7.1 Modèles du code mobile

Avant tout, nous avons besoin d'un modèle pour décrire la mobilité et nous permettre d'isoler et de mieux comprendre les problèmes qui lui sont liés. Ces modèles servent ensuite de base à la conception de langages de haut niveau utilisables en pratique. Plusieurs modèles de la mobilité ont été proposés. Le π-calcul de Milner code implicitement la mobilité. Il met en jeu des systèmes de processus qui interagissent en échangeant des noms de

canaux. Ces interactions permettent non seulement de synchroniser l'exécution des processus, mais aussi de changer dynamiquement la distribution de ces processus. Le π-calcul est à la base du langage ERLANG que la société ERICSON a utilisé pour développer des réseaux de téléphonie mobile. Le modèle des *ambients mobiles* code explicitement la mobilité. Il représente à la fois la structure hiérarchique des sites administratifs d'un réseau, et la répartition des processus sur ces sites. En interagissant, les processus peuvent déplacer les sites dans lesquels ils sont exécutés.

7.2 Analyses statiques de code mobile

Plusieurs analyses ont été proposées pour garantir ou prouver des propriétés sur les systèmes mobiles. Elles considèrent essentiellement des propriétés de sûreté (« *safety properties* »), comme la confidentialité de l'information qui circule dans les systèmes de communication ou le nombre de processus qui sont utilisés par les systèmes. Elles consistent essentiellement à montrer que les configurations qui conduiraient à la perte de la propriété que nous cherchons à valider n'apparaissent jamais.

Nous distinguons deux types de propriétés qui suffisent à garantir l'absence de telles configurations :

- l'analyse des interférences permet de détecter quels processus sont susceptibles d'interagir parce qu'ils communiquent sur un même canal ou qu'ils sont localisés sur un même site. Nous avons réalisés une analyse des interférences à la fois dans le cas de la mobilité implicite [26] et dans celui de la mobilité explicite [56]. Ces analyses permettent de distinguer les objets créés par des instances récursives de processus, ce qui est fondamental dans l'étude de la mobilité ;
- une analyse du nombre de processus ou analyse de forme, qui permet de détecter, indépendamment des objets qui leur sont communiqués, quelle est la répartition des processus dans le système. Nous avons réalisé une analyse qui permet de compter le nombre de processus utilisés par un système mobile, dans le cas de la mobilité implicite [12]. Elle fait apparaître des exclusions mutuelles et permet de garantir le non-épuisement des ressources.

L'étude des systèmes mobiles est un enjeu important. Jusqu'à maintenant, toutes les analyses ont essentiellement porté sur des propriétés de sûreté et ne sont efficaces que sur des petits exemples. Il est indispensable de prendre en compte les propriétés de vivacité et de proposer des méthodes pour leur analyse. Il est important de passer à l'échelle industrielle et de considérer l'étude des langages réels comme ERLANG. Ceci ne peut être fait sans une analyse modulaire des réseaux, qui permette non seulement d'analyser un système morceau par morceau, mais aussi de regrouper l'information obtenue.

8 Analyses statiques probabilistes

Il est parfois nécessaire de considérer des programmes au comportement probabiliste, que ce soit pour l'étude des algorithmes probabilistes proprement dits ou pour l'étude de systèmes embarqués placés dans un environnement décrit de façon probabiliste. Dans ce dernier cas, il est en particulier intéressant d'estimer la probabilité d'atteindre des états de panne ou la probabilité que le temps d'exécution dépasse un certain seuil. Nous avons donc recherché des méthodes permettant d'obtenir par interprétation abstraite de telles estimations, ou du moins des majorants sur les valeurs considérées.

Ces recherches ont été menées avec deux objectifs :

- la réutilisation des techniques d'interprétation abstraite déjà existantes, y compris si possible au niveau de l'implantation ;
- l'exactitude mathématique de l'analyse ; en particulier, l'analyse ne devra pas introduire des hypothèses supplémentaires sur les probabilités (par exemple l'indépendance de certaines variables).

Afin de pouvoir garantir l'exactitude mathématique des analyses, nous avons dû définir des sémantiques adaptées aux programmes probabilistes et donner des relations d'abstraction ou d'équivalence entre elles. Parmi les sémantiques possibles, certaines donnent directement des possibilités d'analyse par interprétation abstraite :

- sémantique dénotationnelle en avant (suivant Kozen et [33]) ;
- sémantique dénotationnelle en arrière (adjoint linéaire de la précédente) [35].

Nous avons développé des treillis abstraits adaptés à ces sémantiques :

- sommes finies [33] ;
- localisation des mesures [33] ;
- gaussiennes ;
- queues sous-exponentielles (permettant l'analyse de terminaison de certains programmes et la validation d'analyses statistiques de temps moyens) [59].

Ces méthodes souffrent néanmoins d'une certaine complexité, aussi nous avons développé une méthode d'implantation plus simple, combinant interprétation abstraite et statistiques de Monte-Carlo [34 , 60]. Cette méthode fournit des bornes supérieures d'évènements rares ainsi qu'une probabilité que cette borne soit valable. Une analyse préalable par interprétation abstraite ordinaire permet de limiter le nombre d'échantillons nécessaires à une bonne estimation. L'analyse est parallélisable avec un gain linéaire et peu de communications, ce qui permet de l'implanter sur un réseau de PC, peu coûteux par rapport à une machine parallèle conventionnelle. Cette méthode étant assez simplement adaptable à un analyseur préexistant, le CEA doit prochainement l'implanter dans son analyseur industriel Caveat.

Nous avons en outre développé un petit analyseur prototype de démonstration pour deux de nos analyses [31 , 34].

9 Analyses statiques temporelles

Les analyses statiques classiques portent essentiellement sur les propriétés de sûreté (« *safety* ») ou d'invariance (comme l'absence d'erreurs à l'exécution). Au delà des propriétés de sûreté, il est maintenant fréquent de vouloir vérifier automatiquement des propriétés de programmes plus fines et donc, en général, également beaucoup plus complexes comme les propriétés temporelles [25]. L'exemple le plus connu de propriété temporelle est la vivacité (« *liveness* »), par exemple pour démontrer automatiquement la terminaison de processus parallèles équitables partageant des données communes [29].

10 Analyses statiques et vérification de modèles (« *model checking* »)

La vérification de modèles a pour limite l'hypothèse de finitude, voire la complexité pratique notamment en mémoire. L'abstraction, c'est-à-dire l'approximation par interprétation abstraite, est donc nécessaire pour la vérification de modèles infinis.

Nous avons étudié la limite des tendances actuelles de la recherche visant à automatiser l'abstraction interactivement en montrant pour les spécifications de sûreté que la découverte de l'abstraction et la preuve de la correction de l'abstraction sont logiquement équivalentes à une preuve formelle de correction de la spécification [5 , 15].

Une autre tendance de la recherche en vérification de modèles est l'abstraction d'un système de transition concret par un système de transition abstrait de manière à pouvoir réutiliser les vérificateurs de modèles existants. Nous avons proposé une autre façon de procéder consistant à effectuer en parallèle une analyse statique abstraite et la vérification de modèle concret, de manière à réduire l'espace des états à explorer tout en utilisant une plus grande variété d'abstractions non forcément réduites à des systèmes de transition abstraits [11].

11 Réalisation mécanisée d'interpréteurs abstraits

Un interpréteur abstrait doit fournir un résultat sûr ; mais peut-on faire confiance à l'analyse proposée et à son implantation ? Nous avons étudié la possibilité de formaliser dans un environnement de preuve assistée par ordinateur (Coq) la correction d'un interpréteur abstrait telle qu'elle est définie par la théorie de l'interprétation abstraite. Cette correction est notamment assurée par l'usage de modules paramétriques prouvés corrects selon la spécification de leurs paramètres. En instanciant ces modules, il est possible

de produire automatiquement un interpréteur abstrait certifié. Nous avons illustré cela par la production d'un petit interpréteur certifié [44].

12 Applications de l'analyse statique par interprétation abstraite

12.1 Analyse statique de protocoles cryptographiques

L'étude des protocoles cryptographiques (commerce électronique, authentification pour l'accès à un système informatique) peut se faire à deux niveau : au niveau des primitives cryptographiques ou au niveau de la correction du protocole lui-même, les primitives étant supposées vérifier certaines propriétés. Nous nous sommes intéressés à ce second aspect des choses.

Supposant idéales les primitives cryptographiques, nous exprimons les protocoles dans le modèle de Dolev-Yao, où les données échangées sont des termes sur une signature décrivant les primitives utilisées et les constantes générées. Il est alors possible de définir une sémantique des protocoles modélisant le cas d'un intrus contrôlant le réseau. S'il n'est alors évidemment pas possible dans un modèle où l'intrus a un pouvoir aussi grand de démontrer des propriétés de vivacité, il est néanmoins possible de s'assurer de la sécurité du protocole : certaines données secrètes ne doivent en aucun cas tomber en la possession de l'intrus, ou certaines machines ne doivent pas parvenir à un état marquant le bon fonctionnement tout en ayant accepté certaines données incorrectes.

Nous approchons cette sémantique par interprétation abstraite en utilisant un domaine d'automates d'arbres [31 , 61]. Cette analyse a été implantée dans un prototype.

12.2 Test abstrait de logiciel

Les sociétés modernes sont très informatisées et donc hautement sensibles aux bogues dans les logiciels, en particulier ceux qui pilotent des systèmes critiques que l'on trouve dans les transports, la médecine ou le commerce électronique. Les travaux sur la vérification des logiciels sont donc très nombreux et la fiabilité et la sûreté de fonctionnement du logiciel constitue certainement une des applications principales de l'analyse statique de programmes. Cependant, toutes les méthodes de vérification de logiciels ont des limites théoriques liées aux problèmes d'indécidabilité (ce qui implique ultimement une intervention humaine) et des limites pratiques dues aux difficultés d'usage. De ce fait, le test est encore la méthode la plus utilisée pour mettre au point les programmes industriels.

Le test abstrait [8] repose sur l'idée de vérifier la correction des programmes par morceaux sur des spécifications partielles, mais en remplaçant

les jeux de données par des spécifications fournies sous forme de propriétés abstraites spécifiant les comportements souhaités des programmes. Les exécutions sur ces jeux de données sont alors remplacées par des analyses statiques.

Le test abstrait étant facilement compréhensible par analogie avec les méthodes de test classiques, on peut espérer qu'il n'y aura pas de difficultés d'usage. L'intervention humaine, ultimement nécessaire, est limitée à la compréhension des algèbres abstraites utilisées à des niveaux de raffinement variés. On peut donc espérer un excellent taux de couverture pour un coût humain raisonnable, même pour de grands programmes.

12.3 Tatouage de logiciels par interprétation abstraite

Parmi les applications originales de l'analyse statique par interprétation abstraite, citons le tatouage de code mobile permettent au propriétaire d'incruster dans le code source de manière indécelable et indélébile une greffe logicielle identifiant ce composant [55]. La marque indélébile à inclure n'est pas inscrite directement dans le texte du programme, ce qui la rendrait trop facilement modifiable, mais dans le comportement à l'exécution de l'objet sans que cela altère ses fonctionnalités d'origine. Ces informations sont invisibles lors de l'utilisation de l'objet mais peuvent être extraites par analyse statique.

13 Perspectives

L'équipe travaille sur le thème de l'efficacité et de la fiabilité des systèmes informatiques qui est porteur au delà des évolutions rapides de l'informatique et vertical depuis la théorie jusqu'à, plus récemment, l'industrialisation d'abord aux USA puis en Europe. L'équipe a une visibilité internationale que traduisent les invitations, les visiteurs et les contrats. Elle participe au transfert technologique (au travers de l'essaimage des doctorants et par les contrats impliquant des industriels).

L'équipe est petite (avec deux enseignants permanents dont un depuis septembre 2000 et trois doctorants), avec un grand taux de renouvellement. L'équipe a donc des possibilités de croissance, ce qui est nécessaire pour répondre aux problèmes et satisfaire aux besoins de formation par la recherche résultant de la phase actuelle d'industrialisation de l'interprétation abstraite.

Eléments d'appréciation de l'activité du laboratoire

1 Collaborations

– Réseaux européens « capital humain et mobilité » :

ABILE : Abstract interpretation for declarative languages.
R. Barbuti (U. de Pise), M. Bruynooghe (U. de Louvain), P. Codognet (INRIA Rocquencourt), M.-M. Corsini (U. de Bordeaux), P. Cousot (ENS), R. Cousot (École polytechnique), P. Devienne (U. de Lille), G. Filè (U. de Padoue), M. Hanus (Max-Planck-Institut für Informatik, Saarbrücken), M. Hermenegildo (U. polytechnique de Madrid), N. Jones (U. de Copenhagen), B. L. Charlier (Facultés U. de Namur), G. Levi (U. de Pise), J. Małuszyński (U. de Linköping), A. Mycroft (U. de Cambridge) et U. Nilsson (U. de Linköping).
HCM european network, jan. 1995 – déc. 1997.

– Contrats européens « Technologie des sociétés d'information » :

DAEDALUS : Validation of critical software by static analysis and abstract testing.
P. Cousot (ENS), R. Cousot (École polytechnique), A. Deutsch (Polyspace technologies), C. Ferdinand (AbsInt), É. Goubault (CEA), N. Jones (DIKU), D. Pilaud (Polyspace technologies), F. Randimbivololona (EADS-Airbus), M. Sagiv (U. Tel Aviv), H. Seidel (U. Trèves) et R. Wilhelm (U. Sarrebruck).
Project IST-1999-20527 of the european 5^{th} Framework Programme (FP5), oct. 2000 – oct. 2002.

– Contrats bilatéraux (programmes d'actions intégrées) :

Sécurité de systèmes distribués par interprétation abstraite.
P. Cousot (ENS) et R. Giacobazzi (U. Vérone).
Programme d'actions intégrées franco-italiennes Galilée, jan. 1999 – déc. 2000.

Model-checking et analyse statique.
P. Cousot (ENS) et A. Podelski (Max-Planck-Institut für Informatik).
Programme d'actions intégrées franco-allemandes PROCOPE, jan. 2000 – déc. 2000.

– Contrats institutionnels français :

TUAMOTU : Tatouage électronique sémantique de code mobile Java™.
P. Cousot (ENS), R. Cousot (École polytechnique) et M. Riguidel (Thomson-CSF Communications).
Project RNRT 1999 n° 95, oct. 1999 – oct. 2001.

Analyses statiques probabilistes.
P. Cousot (ENS) et É. Goubault (CEA).
Contrat ENS — CEA, n° SAV 27234/VSF, jan. 1999 – déc. 2001.

– Contrats industriels :

Étude des procédés de signature logicielle pour les objets mobiles écrits en Java™.
P. Cousot (ENS) et M. Riguidel (Thomson-CSF Communications).
Contrat ENS — Thomson-CSF Communications, jan. 1999 – déc. 2000.

2 Missions, conférences et séminaires

– P. COUSOT :

séminaires à Sarrebruck (Allemagne – juin 1997) [69], Obernai (septembre 1997) [36], KAIST (Taejon, République de Corée – novembre 1997) [45], Paris (janvier 1998) [65], Udine (Italie – septembre 1998) [67], Dagstuhl Seminar 99151 (Allemagne – avril 1999) [72], Paris (mai 1999) [68], Rennes (janvier 2000) [70], KAIST (Taejon, République de Corée – juin 2000) [20, 71], Montréal (Canada – septembre 2000) [63] ;

conférences à Paris (septembre 1997) [7], Sydney (Australie – décembre 1997) [24], Valence (Espagne – juin 1998) [18], Pise (SAS'98, Italie – septembre 1998), Padoue (Italie – mai 1999) [16], Paris (septembre 1999), Venise (SAS'99, Italie – septembre 1999), Osaka (Japon – novembre 1999) [19], Boston (USA – janvier 2000) [25], Schloß Ringberg (Allemagne – février 2000) [23], Santa Barbara (SAS'00, USA – juin 2000), Austin (USA – juillet 2000) [5, 62], L'Aquila (Italie – août 2000) [4, 8], Sarrebruck (Allemagne – août 2000) [21], La Réunion (novembre 2000) [15], Santa Cruz (USA – janvier 2001) [37], Londres (POPL'01, UK – janvier

2001), Redmond (USA – février 2001) [64], Munich (Allemagne – mars 2001) [22], Gênes (ESOP'01, Italie – avril 2001) ;

cours à Nantes (avril 1998) [48], Udine (Italie – septembre 1998) [47], Paris (janvier 1999) [49], Marktoberdorf (Allemagne – juillet/août 1998) [46 , 53 , 66], Paris (octobre 2000) [50].

– Jérôme Feret :

séminaires à l'École normale supérieure (mars et novembre 2000), P.P.S. (Chevaleret, janvier 2001), LORIA, Nancy (janvier 2001) ;

conférences à Pise (SAS'98, septembre 1998), Venise (SAS'99, septembre 1999), Paris (ICFP'99, septembre 1999), Paris (PPDP'99, septembre 1999), Santa Barbara (SAS'00, juin 2000) [26], State College (GETCO'00, USA, août 2000) [12], State College (CONCUR'00, USA, août 2000).

– Laurent Mauborgne :

séminaires à Paris (juin 1999) [74], Paris (janvier 2000) [77], Copenhague (Danemark – mars 2000) [75], Paris (mars 2000) [78], Saarbrück (Allemagne – avril 2000) [79], Bruxelles (Belgique – décembre 2000) [76] ;

conférences à Pise (SAS'98, Italie – septembre 1998), Venise (Italie – septembre 1999) [27], Berlin (Allemagne – mars 2000) [28], Santa Barbara (USA – juin 2000) [29], Londres (POPL'01, UK – janvier 2001), Gênes (ETAPS'01, Italie – avril 2001) ;

cours à Copenhague (Danemark – février-mars 2000) [54] ;

séjour à Saarbrück (Allemagne – avril-mai 2000).

– Antoine Miné :

séminaire à Grenoble (novembre 2000) ;

conférence à Santa Barbara (USA – juin 2000) ;

école d'été à Cahmina (Portugal – septembre 2000).

– David Monniaux :

séminaires à Dagstuhl (Allemagne – avril-mai 2000), Londres (Royaume-Uni – avril 2000), Göteborg (Suède – juin 2000) ;

conférences à Mordano (CSFW'12, Italie – juin 1999) [32], Venise (SAS'99, Italie – septembre 1999) [31], Santa Barbara (SAS'00, USA – juillet 2000) [33], Sarrebruck (Allemagne – août 2000), Londres (POPL'01, Royaume-Uni – janvier 2001) [34], Gênes (ESOP'01, Italie – avril 2001) [35].

3 Accueil de chercheurs

Professeurs et directeurs de recherche invités

- Peter Lee (Carnegie Mellon University), juillet 1997. (Professeur invité ENS) ;
- David Schmidt (Kansas State University), juin 1998. (Professeur invité ENS) ;
- Reinhard Wilhelm (Universität des Saarlandes), septembre 1999. (Professeur invité ENS) ;
- Andreas Podelski (Max-Planck-Institut für Informatik), mars 2000. (Professeur invité ENS) ;
- Barbara Ryder (Rutgers University), mai 2001. (Professeur invité ENS).

Post-doctorants invités

- Germán Puebla Sánchez (Universidad Politécnica de Madrid), octobre 1997 – décembre 1997.
- Jean-François Raskin (F.U.N.D.P., Namur), octobre 1997 – décembre 1997.

4 Diffusion de la connaissance

- Organisation à l'ENS du séminaire SIA « 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 :

  1997/98 : Bruno Blanchet, Ferrucio Damiani, Christèle Faure & Mohamed Tadjouddine, Éric Goubault, Francesca Levi, Corrado Priami, Andreas Podelski, Marc Pouzet , German Puebla, Jean-François Raskin, David Schmidt, Francesca Scozzari, Stanislav Tzolovski, Franck Védrine, Bernhard Steffen ;

  1998/99 : Martín Abadi, Salvador Lucas Alba, Isabelle Attali, Bruno Blanchet, Pascal Brisset, Dominique Cansell, Marco Comini, Charles Consel, Loïc Correnson, Roberto Giacobazzi, Éric Goubault, Jean Goubault-Larrecq, Nabil El Kadhi, Nicolas Halbwachs, Vania Joloboff, Leslie Lamport, Yanhong Annie Liu, Renaud Marlet, Laurent Mauborgne, Dominique Méry, David Monniaux, Andreas Podelski, Scott D. Stoller, Anatole Ursu, Paolo Volpe ;

  1999/2000 : Bruno Blanchet, François Bourdoncle, Witold Charatonik, Patrick Cousot, Jérôme Feret, Éric Goubault, Jean Goubault-Larrecq, Jean-Luc Lambert, Leslie Lamport, Laurent Mauborgne, Nicolas Mercouroff, Dominique Méry, David Monniaux,

Andreas Podelski, Jean-Pierre Talpin, Tomás Uribe, Franck Védrine, Reinhard Wilhelm ;

2000/2001 : Bruno Blanchet, Francesco Logozzo, David Monniaux, Jean-François Raskin, Barbara Ryder, Andrei Sabelfeld, Helmut Seidel.

- P. Cousot est membre du groupe de travail WG 2.3 de l'IFIP sur la « méthodologie de la programmation ». Il participe à ses réunions de travail [37] ou les organise régulièrement en France [36] ;
- Participation à des Écoles pour jeunes chercheurs [48] ;
- Cours de doctorat sur l'interprétation abstraite à l'étranger [45, 47, 53].

5 Réalisation et diffusion de logiciels, brevets

La taille de l'équipe ne permet pas de développer des analyseurs statiques de qualité industrielle (dont le coût de développement est souvent supérieur à 25 personnes/années) et d'assurer ensuite leur maintenance et le suivi de leur évolution. Par conséquent, les logiciels développés dans l'équipe sont des prototypes dont le coût de développement est de l'ordre de quelques personnes/mois (en général 3 à 9 personnes/mois). Ces prototypes servent essentiellement à l'expérimentation nécessaire pour estimer le coût et la précision des analyses.

- P. Cousot et A. Venet ont développé un tatoueur de méthodes de classes Java™ dans le cadre du contrat TUAMOTU [55] ;
- J. Feret a implanté un analyseur pour les systèmes mobiles spécifiés en π-calcul. Le propotype est sur le serveur WEB du DI ;
- A. Miné a programmé un prototype d'analyseur statique basé sur les domaines numériques abstraits [30, 58] ;
- D. Monniaux a développé un analyseur de protocoles cryptographiques par interprétation abstraite dans un domaine d'automates d'arbres ;
- D. Monniaux a développé un analyseur de programmes probabilistes écrits dans un sous-ensemble du langage C. Cet analyseur est disponible sur la Toile en téléchargement (`http://www.di.ens.fr/~monniaux/download/absinthe.tar.gz`) et en consultation directe (`http://cgi.dmi.ens.fr/cgi-bin/monniaux/absinthe`) ;
- D. Monniaux a développé des bibliothèques d'interfaçage entre le langage Objective Caml et les librairie GMP (calcul scientifique) et Gtk ; ces bibliothèques sont mises à disposition sur la toile (`http://www.di.ens.fr/~monniaux/programmes.html.fr`) ;
- D. Monniaux a développé un classificateur d'adresses IP selon leur origine géographique utilisant des structures de données habituellement utilisées pour l'analyse de programmes.

6 Participation à l'évaluation de la recherche

– P. Cousot :

membre du comité de rédaction de la revue *SCP* (jusqu'en 2000) ;

président du comité de programme de SAS'01 ;

membre du comité de programme de SAS'99, SAS'00, WSAGV'2000 ;

Évaluateur pour les conférences internationales ECOOP'00, ESOP'99 ESOP'00, ESOP'01, KR'00, LICS'99 PEPM'97, PLDI'01, PLI'99, POPL'97, SAIG'00, SAS'97, SAS'98 et SAS'99 ;

Évaluateur pour les journaux ACM Computing Surveys, ACM TOPLAS, Acta Informatica, SCP, JACM, JASE, TCS, ;

Évaluateur pour les soumissions de projets européens IST (FP 5 RTD Programmes), autrichiens « Fonds zur Förderung der wissenschaftlichen Forschung » (FWF), italiens (Istituto Superiore Mario Boella, ISMB et « Centre International des Sciences Mécaniques » (CISM), Udine), finlandais (Finish Programme for Centers of Excellence in Research 2001–2007 de l'académie de Finlande), néerlandais « Stichting informatica-onderzoek in Nederland » (SION, the Netherlands Computer Science Research Foundation) pour le compte de « the Netherlands Organisation for Scientific Research (NWO) » ;

Évaluateur pour le projet européen Esprit L.T.R. project 23498 vires, Prof. P. Wolper (coordinateur) ;

Membre du conseil scientifique du département EECS du KAIST (Taeduk Science Complex, Taejon, République de Corée) ;

Expert pour le recrutement d'enseignants et chercheurs (Ben-Gurion University of the Negev (Israël), Carnegie Melon University (Computer Science Department), Imperial College of Science, Technology and Medicine de Londres (Angleterre), Rutgers University, New Jersey, (Computer Science Department), Université de Cambridge (Angleterre), Université de Melbourne (Australie), Université de Sarrebrück (Allemagne)).

– J. Feret :

Évaluateur pour les conférences ESOP'00, PPDP'00, ESOP'01 et SAS'01.

– L. Mauborgne :

Évaluateur pour les conférences POPL'97, FMPPTA'97, FMPPTA'98, ICCL'98, SAS'99, ESOP'00, PPDP'00, ESOP'01, ESOP'01 et SAS'01.

– A. Miné :

Évaluateur pour les conférences ESOP'01, PADO-II et SAS'01.

– D. Monniaux :

Évaluateur pour les conférences FMPPTA'99, FMPPTA'2000, FMPPTA'01, ESOP'01 et SAS'01.

7 Encadrement doctoral

Direction de thèses

– P. Cousot :

J. Goubault, *Logique, complexité, démonstration automatique et thèmes annexes* [40], Habilitation à diriger les recherches, Université de Paris 9, Dauphine, soutenue le 27 juin 1997 ;

L. Mauborgne, *Représentation d'ensembles d'arbres pour l'interprétation abstraite* [41], Thèse de doctorat de l'École polytechnique en informatique, soutenue le 25 novembre 1999 ;

F. Védrine, *Analyses totales de programmes par l'interprétation abstraite* [42], Thèse de doctorat de l'École polytechnique en informatique, soutenue le 28 janvier 2000.

– P. Cousot et D. Parigot :

L. Correnson, *Sémantique équationelle* [39], Thèse de doctorat de l'École polytechnique en informatique, soutenue le 1er avril 2000.

– P. Cousot et A. Deutsch :

B. Blanchet, *Analyse d'échappement, Applications à ML et Java*™ [38], Thèse de doctorat de l'École polytechnique en informatique, soutenue le 7 décembre 2000.

Participation à d'autres jurys de thèses

– P. Cousot : 7 jurys dont 1 présidence et 3 rapports.

Direction de projets de DEA

– P. Cousot :

D. Monniaux, *Réalisation mécanisée d'interpréteurs abstraits* [44]. 1998 ;

A. Miné, *Représentation d'ensembles de contraintes de somme ou de différence de deux variables et application à l'analyse automatique de programmes* [43]. 2000.

8 Enseignement

Deuxième cycle

– P. Cousot :

Directeur des études pour l'informatique à l'ENS ;

Co-responsable du magistère MMFAI ;

Cours « Langages de programmation et compilation » au MMFAI [51] ;

Cours « Sémantiques des langages de programmation » au MMFAI [52].

– L. Mauborgne :

TDs d'algorithmique à l'École Polytechnique (1997-99 et 2000-01) ;

TDs de compilation au MMFAI (1999-2001) ;

TDs d'algorithmique et programmation au MMFAI (1999-2001).

– A. Miné :

Encadrement de travaux dirigés pour le cours de programmation système à l'École Polytechnique (2001).

– D. Monniaux :

Travaux dirigés et pratiques d'algorithmique et programmation en première année de DEUG MASS à l'Université Paris 9 Dauphine ;

Participation à l'évaluation des stages en entreprise à l'École Nationale Supérieure de Techniques Avancées (1999).

Troisième cycle

– P. Cousot :

Cours *Fondement de l'interprétation abstraite* du DEA « Programmation : sémantique, preuves et langages ».

– J. Feret :

Participation au cours d'analyse statique par interprétation abstraite de Radhia Cousot, dans le cadre du DEA « Programmation : sémantique, preuves et langages » (2001).

– L. Mauborgne :

Participation au cours d'analyse statique par interprétation abstraite de Radhia Cousot, dans le cadre du DEA « Programmation : sémantique, preuves et langages » (1999-2001) ;

Cours sur les représentations d'ensembles d'arbres en analyse de programme à l'Université de Copenhague (2000).

– D. Monniaux :

Participation au cours d'analyse statique par interprétation abstraite de Radhia Cousot, dans le cadre du DEA « Programmation : sémantique, preuves et langages » (2001).

Publications

Édition d'actes ou d'ouvrages collectifs

[1] P. Cousot, É. Goubault, J. Gunawardena, M. Herlihy, M. Raussen et V. Sassone (édité par). – *GEometry and Topology in COncurrency theory.* – Elsevier, 2001, volume 39. `http://www.elsevier.nl/locate/entcs/volume39.html`.

Chapitres de livres ou d'ouvrages collectifs

[2] P. Cousot. – The calculational design of a generic abstract interpreter. *In : Calculational System Design*, éd. par M. Broy et R. Steinbrüggen, pp. 421–505. – NATO Science Series, Series F : Computer and Systems Sciences. IOS Press, 1999, volume 173.

Articles invités

[3] P. Cousot. – Directions for research in approximate system analysis. *ACM Comput. Surv.*, vol. 31, n° 3es, septembre 1999.

[4] P. Cousot. – Abstract interpretation : Achievements and perspectives, papier invité. *Proc. SSGRR 2000 Computer & eBusiness International Conference*, 31 juillet – 6 août 2000.

[5] P. Cousot. – Partial completeness of abstract fixpoint checking, papier invité. *Proc. 4^{th} Int. Symp. SARA '2000. Horseshoe Bay, TX, USA, LNAI 1864*, 26–29 juillet 2000, pp. 1–25.

[6] P. Cousot. – Abstract interpretation based formal methods and future challenges, papier invité. *Proc. Int. Conf. at the Occasion of Dagstuhl's 10^{th} Anniversary, "Informatics — 10 Years Back, 10 Years Ahead"*, 28–31 août 2000.

[7] P. Cousot. – Abstract interpretation based static analysis parameterized by semantics, papier invité. *Proc. 4^{th} Int. Symp. SAS '97, LNCS 1302*, 8–10 septembre 1997, pp. 388–394.

[8] P. Cousot et R. Cousot. – Abstract interpretation based program testing, papier invité. *Proc. SSGRR 2000 Computer & eBusiness International Conference*, 31 juillet – 6 août 2000.

Articles dans des revues internationales avec comité de lecture

[9] P. Cousot. – Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *ENTCS*, vol. 6, 1997. – `http://www.elsevier.nl/locate/entcs/volume6.html`, 25 pages.

[10] P. Cousot. – Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoret. Comput. Sci.*, À paraître en 2001 (Version préliminaire dans [9]).

[11] P. Cousot et R. Cousot. – Refining model checking by abstract interpretation. *Aut . Soft . Eng.*, vol. 6, 1999, pp. 69–95.

[12] J. Feret. – Occurrence counting analysis for the π-calculus. *Electronic Notes in Theoretical Computer Science*, vol. 39, 2001.

[13] L. Mauborgne. – Abstract interpretation using typed decision graphs. *Science of Computer Programming*, vol. 31, n° 1, mai 1998, pp. 91–112.

Articles dans des revues nationales avec comité de lecture

[14] P. Cousot. – Interprétation abstraite. *TSI*, vol. 19, n° 1-2-3, janvier 2000, pp. 155–164.

Conférences invitées

[15] P. Cousot. – On completeness in abstract model checking from the viewpoint of abstract interpretation, conférence invitée. *In : Réunion Workshop on Implementation of Logics,* Saint Gilles, La Réunion. – 11–12 novembre 2000.

[16] P. Cousot. – Abstract interpretation and types, conférence invitée. *In : Workshop on "Static Analysis and Types",* Palazzo del Bó, Padoue, IT. – 17 & 18 mai 1999.

[17] P. Cousot. – Discrete fixpoint approximation methods in program static analysis, conférence invitée. *In : 7^{th} Int. Coll. on Numerical Analysis and Computer Science with Applications, NACSA' 98.* – Plovdiv, BG, 13–17 août 1998.

[18] P. Cousot. – Rule-based specifications and their abstract interpretation, conférence invitée. *In : Fourth Advanced Seminar on Foundations of Declarative Programming, ASFDP'98, Valence, Espagne.* – 15 juin 1998.

[19] P. Cousot. – Abstraction in abstract interpretation, conférence invitée. *In : Workshop on Refinement and Abstraction, ETL, Osaka, Japon.* – 15–17 novembre 1999.

[20] P. Cousot. – An overview of abstract interpretation and program static analysis, conférence invitée. *In : 1^{st} Int. Advisory Board Workshop, EECS Dept., KAIST, Taeduk Science Complex, Taejon, République de Corée.* – 14 juin 2000.

[21] P. Cousot. – Progress on abstract interpretation based formal methods and future challenges, conférence invitée. *In : Conference at the Occasion of Dagstuhl's 10th Anniversary, "Informatics — 10 Years Back, 10 Years Ahead",* Saarland University Campus, Sarrebruck, DE. – 28–31 août 2000.

[22] P. Cousot. – Abstract interpretation for software verification, conférence invitée. *In : Workshop on Formal Design of Safety Critical Embedded Systems (FEmSys '2001),* Munich, DE. – 21–23 mars 2001.

[23] P. Cousot et R. Cousot. – Abstract testing versus abstract model-checking, conférence invitée. *In : Schloß Ringberg Seminar on « Model Checking and Program Analysis » organisé par A. Podelski, B. Steffen et M. Vardi.* – 20–23 février 2000.

Communications dans des conférences internationales avec comité de lecture

[24] P. Cousot et R. Cousot. – Abstract interpretation of algebraic polynomial systems. *In : Proc. 6th Int. Conf. AMAST '97*, éd. par M. Johnson. *Sydney, AU, LNCS 1349*, pp. 138–154. – Springer-Verlag, 13–18 décembre 1997.

[25] P. Cousot et R. Cousot. – Temporal abstract interpretation. *In : 27th POPL.* pp. 12–25. – Boston, MA, janvier 2000.

[26] J. Feret. – Confidentiality analysis of mobile systems. *In : Seventh International Static Analysis Symposium (SAS'00). LNCS.* – Springer-Verlag, 2000.

[27] L. Mauborgne. – Binary decision graphs. *In : Static Analyis Symposium (SAS'99)*, éd. par A. Cortesi et G. Filé. *Lecture Notes in Computer Science*, volume 1694, pp. 101–116. – Springer-Verlag, 1999.

[28] L. Mauborgne. – Improving the representation of infinite trees to deal with sets of trees. *In : European Symposium on Programming (ESOP 2000)*, éd. par G. Smolka. *Lecture Notes in Computer Science*, volume 1782, pp. 275–289. – Springer-Verlag, 2000.

[29] L. Mauborgne. – Tree schemata and fair termination. *In : Static Analyis Symposium (SAS'00)*, éd. par J. Palsberg. *Lecture Notes in Computer Science*, volume 1824, pp. 302–320. – Springer-Verlag, 2000.

[30] A. Miné. – A new numerical abstract domain based on difference-bound matrices. *In : PADO II*, *Lecture Notes in Computer Science.* – 2001. À paraître.

[31] D. Monniaux. – Abstracting cryptographic protocols with tree automata. *In : Sixth International Static Analysis Symposium (SAS'99). Lecture Notes in Computer Science.* – Springer-Verlag, 1999.

[32] D. Monniaux. – Decision procedures for the analysis of cryptographic protocols by logics of belief. *In : 12th Computer Security Foundations Workshop*. – IEEE, 1999.

[33] D. Monniaux. – Abstract interpretation of probabilistic semantics. *In : Seventh International Static Analysis Symposium (SAS'00). Lecture Notes in Computer Science*. – Springer-Verlag, 2000.

[34] D. Monniaux. – An abstract Monte-Carlo method for the analysis of probabilistic programs (extended abstract). *In : 28th Symposium on Principles of Programming Languages (POPL '01)*. Association for Computer Machinery, pp. 93–101. – 2001.

[35] D. Monniaux. – Backwards abstract interpretation of probabilistic programs. *In : European Symposium on Programming. Lecture Notes in Computer Science*. – Springer-Verlag, 2001.

Autres conférences

[36] P. Cousot. – A few remarks on the abstraction and equivalence of semantics. *In : IFIP WG 2.3, Obernai*. – 26 septembre 1997.

[37] P. Cousot. – Introduction to a discussion on mechanical formal methods for software verification. *In : IFIP WG 2.3 Santa Cruz Meeting*. – 7–12 janvier 2001.

Thèses et habilitations

[38] B. Blanchet. – *Analyse d'échappement, Applications à ML et Java*™. – Palaiseau, Thèse de doctorat en informatique, École polytechnique, 7 décembre 2000.

[39] L. Correnson. – *Sémantique équationelle*. – Palaiseau, Thèse de doctorat en informatique, École polytechnique, 1er avril 2000.

[40] J. Goubault. – *Logique, complexité, démonstration automatique et thèmes connexes*. – Paris, Habilitation à diriger les recherches, Université de Paris 9, Dauphine, 27 juin 1997.

[41] L. Mauborgne. – *Représentation d'ensembles d'arbres pour l'interprétation abstraite*. – Palaiseau, Thèse de doctorat en informatique, École polytechnique, 25 novembre 1999.

[42] F. Védrine. – *Analyses totales de programmes par l'interprétation abstraite*. – Palaiseau, Thèse de doctorat en informatique, École polytechnique, 28 janvier 2000.

Rapports de DEA

[43] A. Miné. – *Représentation d'ensembles de contraintes de somme ou de différence de deux variables et application à l'analyse automatique de programmes*. – Mémoire de stage de DEA, ENS, 2000. `http://www.eleves.ens.fr:8080/home/mine/stage_dea/index.shtml.en`.

[44] D. Monniaux. – *Réalisation mécanisée d'interpréteurs abstraits.* – Rapport de DEA , Université Paris VII, 1998.

Notes de cours

[45] P. Cousot. – Workshop on abstract interpretation. – 10–15 novembre 1997. KAIST, Taeduk Science Complex, Taejon, KR.

[46] P. Cousot. – Calculational design of semantics and static analyzers by abstract interpretation. – 28 juillet – 9 août 1998. NATO Int. Summer School 1998 on Calculational System Design. Marktoberdorf, DE. Organized by F.L. Bauer, M. Broy, E.W. Dijkstra, D. Gries and C.A.R. Hoare.

[47] P. Cousot. – Corso di interpretazione astratta. – 9–12 septembre 1998. Dottorato di Ricerca, Dip. di Informatica, Univ. di Udine, IT.

[48] P. Cousot. – Interprétation abstraite. – 1er avril 1998. École jeunes chercheurs en programmation, GDR Programmation du CNRS, École des Mines de Nantes, Nantes, 23 mars – 2 avr. 1998.

[49] P. Cousot. – Analyse statique de logiciels : du test exhaustif à la vérification automatique. – 28 janvier 1999. Séminaire de formation de l'Institut de l'École normale supérieure et du Collège de Polytechnique sur l'« Analyse Statique de Logiciels », Paris.

[50] P. Cousot. – Interprétation abstraite. – 24–25 octobre 2000. Journées ASPROM sur la Sûreté des Logiciels, Paris.

[51] P. Cousot. – Langage de programmation et compilation. – 10 octobre 2000. Magistère MMFAI.

[52] P. Cousot. – Sémantique des langages de programmation. – 10 octobre 2000. Magistère MMFAI.

[53] P. Cousot et R. Cousot. – Introduction to abstract interpretation. – 28 juillet – 9 août 1998. Course notes for the "NATO Int. Summer School 1998 on Calculational System Design", Marktoberdorff, DE.

[54] L. Mauborgne. – Sets of trees and abstract interpretation. – février – mars 2000. Cours à l'Université de Copenhague.

Brevets

[55] P. Cousot, R. Cousot, M. Riguidel et A. Venet. – Tatouage de logiciels. – Brevet en cours de dépôt.

Articles soumis ou en préparation

[56] J. Feret. – Abstract interpretation-based static analysis of mobile ambients. – Soumis à SAS'01.

[57] J. Feret. – Dependency analysis of mobile systems. – Soumis à CAV'01.

[58] A. Miné. – The octagon abstract domain, 2001. Soumis à SAS'01.

[59] D. Monniaux. – An abstract analysis of the probabilistic termination of programs. – 2001. Soumission au *Static Analysis Symposium*, 2001.

[60] D. Monniaux. – An abstract Monte-Carlo method for the analysis of probabilistic programs. – 2001. En préparation pour soumission au journal ACM TOPLAS.

[61] D. Monniaux. – Abstracting cryptographic protocols with tree automata. – 2001. Version étendue de [31]. En cours de soumission à *Science of Computer Programming*.

Miscellanea

[62] P. Cousot. – Contribution to the panel on "Abstractions in AI and software engineering". – 4$^{\text{th}}$ Int. Symp. SARA '2000, Horseshoe Bay, TX, USA, 26–29 juil. 2000.

[63] P. Cousot. – Research on abstract interpretation at ENS with a few words on software abstract watermarking. – Seminar, CS Department, Mc Gill University, Montreal, Canada, September 20$^{\text{th}}$, 2000.

[64] P. Cousot. – On the design of abstractions for software checking. – 12 février 2001. Microsoft Research, Redmond, WA, USA.

[65] P. Cousot. – From semantics to types systems by abstract interpretation. – 15 janvier 1998. Séminaire AFPLC "Typages".

[66] P. Cousot. – The Marktoberdorf'98 generic abstract interpreter. – novembre 1998. `http://www.di.ens.fr/~cousot/Marktoberdorf98.shtml`.

[67] P. Cousot. – Refining model checking by abstract interpretation. – 24 septembre 1998. Dip. di Informatica, Univ. di Udine, Italie.

[68] P. Cousot. – Interprétation abstraite et analyse statique. – 26 mai 1999. 10$^{\text{ème}}$ anniversaire du LIX, École polytechnique, Palaiseau.

[69] P. Cousot. – Design of semantics by abstract interpretation. – 2 juin 1997. MPI-Kolloquium, Max-Planck-Institut für Informatik Im Stadtwald, Sarrebruck, DE.

[70] P. Cousot. – Interprétation abstraite temporelle. – 11 janvier 2000. Séminaire de l'IRISA, Rennes.

[71] P. Cousot. – Partial completeness of abstract fixpoint checking. – 13 juin 2000. ROPAS seminar, EECS Dept., KAIST, Taeduk Science Complex, Taejon, République de Corée.

[72] P. Cousot et R. Cousot. – Abstract interpretation, temporal logic and data flow analysis. – 11–16 avril 1999. Dagstuhl Seminar 99151 on Program Analysis, Schloß Dagstuhl, Wadern, Allemagne.

[73] J. Feret. – Pi s.a. : Analyse statique de code mobile par interprétation abstraite. Prototypes.

[74] L. Mauborgne. – Graphes de décision binaire. – juin 1999. Séminaire SIA de l'ENS, Paris.

[75] L. Mauborgne. – Representation of sets of trees for abstract interpretation. – 8 mars 2000. Séminaire "Internet Technologies Meetings" de l'ITUC, Copenhague.

[76] L. Mauborgne. – Representation of sets of trees for abstract interpretation. – 30 novembre 2000. Séminaire de l'ULB, Bruxelles.

[77] L. Mauborgne. – Représentation d'ensembles d'arbres. – 31 janvier 2000. Séminaire du LIAFA, Paris.

[78] L. Mauborgne. – Représentation d'ensembles d'arbres pour l'interprétation abstraite. – 13 mars 2000. ENS, Paris.

[79] L. Mauborgne. – Sets of trees and abstract interpretation. – 11 avril 2000. Séminaire du MPI, Saarbrück.

[80] D. Monniaux. – Efficient storage for storing the country of internet protocol domains. – Possibilité de brevet.