

Action Concertée Incitative
SÉCURITÉ INFORMATIQUE
 Descriptif complet du projet

I - FICHE D'IDENTITÉ DU PROJET

Nom du Projet :

CRISS¹

Titre du Projet :

Contrôle de Ressources et d'Interférence pour Systèmes Synchrones

Type du Projet

Projet de recherche	Projet de recherche multi-thématiques	Projet de recherche avec infrastructure	Autre
XXXXXXXXXXXXXXXXXX			

Durée du projet 36 mois

Description courte du Projet :

Le projet CRISS se focalise sur les questions de sécurité soulevées par le concept de *code mobile* que l'on envisage aujourd'hui d'utiliser dans des domaines aussi variés que les réseaux programmables, les jeux en réseau et les cartes à puces. Les premières propositions – les "applets" de JAVA – ont visé à assurer l'intégrité de l'environnement d'exécution. Nous nous intéressons à deux autres classes de propriétés qui apparaissent naturellement dans le contexte du code mobile :

- La dérivation de bornes sur les ressources nécessaires à l'exécution du code afin d'éviter des attaques de type déni de service.
- Le contrôle du flux d'information afin d'éviter la fuite de données confidentielles (non-interférence).

L'objectif du projet est de développer des méthodes automatiques pour assurer et certifier ces propriétés et ceci dans le cadre d'un langage de coordination de modules séquentiels qui permet une exécution synchrone et déterministe.

Coordinateur du projet :

Nom	Prénom	Laboratoire (sigle éventuel et nom complet)
AMADIO	Roberto	Laboratoire d'Informatique Fondamentale de Marseille (UMR CNRS 6166)

¹Poignard malais à lame sinueuse.

Université de Provence

Équipes ou laboratoires partenaires

INRIA Sophia Antipolis, Projet MIMOSA.
--

LIPN Villetaneuse.

LORIA Nancy, Projets Calligramme et Prothéo.
--

**A - IDENTIFICATION DU COORDINATEUR ET DES
AUTRES
PARTENAIRES DU PROJET :****A1 - Coordinateur du Projet :**

M. Prénom Nom	Roberto AMADIO
Fonction	Professeur des Universités
Laboratoire	Laboratoire d'Informatique Fondamentale de Marseille (UMR-CNRS 6166)
Adresse	CMI, 39 rue Joliot-Curie, 13453 Marseille
Téléphone	04 91 11 36 14
Fax	04 91 11 36 02
Mél	amadio@cmi.univ-mrs.fr

Identification de l'équipe ou du laboratoire

Équipe ou Laboratoire	Laboratoire d'Informatique Fondamentale de Marseille (UMR-CNRS 6166), équipe Modélisation et Vérification.
Adresse	CMI, 39 rue Joliot-Curie, 13453 Marseille

Organisme de rattachement financier de l'équipe pour le présent projet

Université de Provence

Responsable du projet au sein de l'équipe ou du laboratoire

M. ou Mme. Prénom Nom	Solange COUPET-GRIMAL
Fonction	Maître de Conférences
Téléphone	04 91 11 36 17
Fax	04 91 11 36 02
Mél	solange@cmi.univ.fr

Autres membres de l'équipe participant au projet

Nom	Prénom	Poste statutaire	% du temps de recherche consacré au projet
AMADIO	Roberto	Professeur des Universités	60%
DAL ZILIO	Silvano	Chargé de Recherche du CNRS	30%
JAKUBIEC	Line	Maître de Conférences	50%

Références :

- R. Amadio R. and S. Coupet-Grimal. Analysis of a guard condition in type theory (extended abstract). In *Proc. FOSSACS, ETAPS 98, Springer Lect. Notes in Comp. Sci. 1378*. Springer, pp. 48–62.
- R. Amadio. Max-plus quasi-interpretations. In *Proc. Typed Lambda Calculi and Applications (TLCA) 2003, Valencia*, Springer Lecture Notes in Computer Science, to appear.
- R. Amadio. On modelling mobility. *Theoretical Computer Science*, 240 :147-176, 2000.
- R. Amadio R. and C. Meyssonnier. On decidability of the control reachability problem in the asynchronous π -calculus. *Journal of Nordic Computing*, 9(2), pp.70–101.
- R. Amadio and S. Prasad. Modelling IP mobility. *J. of Formal Methods in System Design*, 17(1), pp.61–99.
- G. Barthe, G. Dufay, L. Jakubiec, B. Serpette et S. de Sousa A Formal Executable Semantics of the Java Card Platform. In *Proceedings of ESOP'01*. D. Sands (Ed.). Springer LNCS, 2001.
- G. Boudol and S. Dal Zilio. An Interpretation of Extensible Objects. In *Proc. International Symposium on Fundamentals of Computation Theory (FCT '99), Springer Lect. Notes in Comp. Sci. 1684*.
- S. Coupet-Grimal. An Axiomatization of Linear Temporal Logic in the Calculus of Inductive Constructions (Part 1). *The Journal of Logic and Computation*, to appear.
- S. Coupet-Grimal and C. Nouvet. Formal Verification of an Incremental Garbage Collector (Part 2). *The Journal of Logic and Computation*, to appear.
- S. Coupet-Grimal and L. Jakubiec. Hardware Verification using Co-induction in Coq. In *TPHOLs'99*. Springer LNCS 1690.
- S. Dal Zilio and A. Gordon. Region analysis and a π -calculus with groups. *Journal of Functional Programming*, 12(3), pp.229–292.
- S. Dal Zilio. An Interpretation of Typed Concurrent Objects in the Blue Calculus. In *Proc. International Conference on Theoretical Computer Science (IFIP TCS 2000), Springer Lect. Notes in Comp. Sci. 1872*. Springer.
- L. Jakubiec. *Vérification de Circuits dans Coq*. Université de Provence, 1999.

Identification de l'équipe ou du laboratoire

Équipe ou Laboratoire	INRIA-Sophia-Antipolis, Projet MIMOSA.
Adresse	BP 93, 06902 Sophia Antipolis Cedex

Organisme de rattachement financier de l'équipe pour le présent projet

INRIA Sophia-Antipolis

Responsable du projet au sein de l'équipe ou du laboratoire

M. ou Mme. Prénom Nom	Gérard BOUDOL
Fonction	Directeur de Recherche INRIA
Téléphone	04 92 38 79 40
Fax	04 92 38 79 98
Mél	Gerard.Boudol@sophia.inria.fr

Autres membres de l'équipe participant au projet

Nom	Prénom	Poste statutaire	% du temps de recherche consacré au projet
CASTELLANI	Ilaria	Chargée de Recherche INRIA	40 %
MATOS	Ana	Doctorante	50 %

Références :

- R. Amadio, G. Boudol and C. Lhoussaine. The distributed receptive π -calculus. *ACM Transactions of Programming Languages and Systems (TOPLAS)*, 2003 (to appear).
- G. Boudol. The π -calculus in direct style. *Higher-Order and Symbolic Computation* 11 (1998).
- G. Boudol. The recursive record semantics of objects revisited. *J. of Functional Programming*, 2003 (to appear).
- G. Boudol and I. Castellani. Noninterference for Concurrent Programs. In Proc. *ICALP*, Springer Lecture Notes in Comp. Sci., 2001.
- G. Boudol and I. Castellani. Noninterference for Concurrent Programs and Thread Systems. *Theoretical Computer Science*, 281(1) :109-130, 2002.
- G. Boudol, P. Zimmer. Recursion in the call-by-value λ -calculus. *FICS* (2002).

Identification de l'équipe ou du laboratoire

Équipe ou Laboratoire	LORIA, Projets Calligramme (commun à l'école des Mines de Nancy) et Prothéo.
Adresse	LORIA, Campus Scientifique - B.P. 239, 54506 Vandoeuvre-lès-Nancy

Organisme de rattachement financier de l'équipe pour le présent projet

LORIA

Responsable du projet au sein de l'équipe ou du laboratoire

M. ou Mme. Prénom Nom	Jean-Yves MARION
Fonction	Professeur de l'école des Mines de Nancy
Téléphone	03 83 59 20 18
Fax	03 83 41 30 79
Mél	Jean-Yves.Marion@loria.fr

Autres membres de l'équipe participant au projet

Nom	Prénom	Poste statutaire	% du temps de recherche consacré au projet
BONFANTE	Guillaume	Maître de Conférences	30%
GNAEDIG	Isabelle	Chargée de recherche INRIA	40%

Références :

- G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. On termination methods with space bound certifications. In *Andrei Ershov Fourth International Conference "Perspectives of System Informatics"*, Lecture Notes in Computer Science. Springer, 2001.
- O. Fissore, and I. Gnaedig and H. Kirchner. Termination of rewriting with local strategies. In *Proc. of the IJCAR Workshop Strategies in Automated Deduction*, Gramlich and Bonacina (eds), 2001. Electronic Notes in Theoretical Computer Science, volume 58.
- H. Kirchner and I. Gnaedig. Termination and normalisation under strategies – Proofs in ELAN. *Third International Workshop on Rewriting Logic and Applications, WRLA'2000*, Kanazawa (Japan), Electronic Notes in Theoretical Computer Science, volume 36, 2001.
- J.-Y. Marion. Actual arithmetic and feasibility. In L. Fribourg, editor, *International Workshop on Computer Science Logic - CSL'2001, Paris, France*, volume 2142 of *Springer Lecture notes in Computer Science*, pages 115–129, 2001.
- J.-Y. Marion and J.-Y. Moyen. Efficient first order functional program interpreter with time bound certifications. In *LPAR*, volume 1955 of *Lecture Notes in Computer Science*, pages 25–42. Springer, Nov 2000.
- J.-Y. Marion. *Complexité implicite des calculs, de la théorie à la pratique*. Université de Nancy, 2000. Habilitation à diriger des recherches.

Identification de l'équipe ou du laboratoire

Équipe ou Laboratoire	Laboratoire d'Informatique de Paris Nord (UMR-CNRS 7030), équipe Logique, Calcul, Raisonnement
Adresse	99, av. Jean-Baptiste Clément 93430 Villetaneuse

Organisme de rattachement financier de l'équipe pour le présent projet

Université Paris-Nord

Responsable du projet au sein de l'équipe ou du laboratoire

M. ou Mme. Prénom Nom	Patrick BAILLOT
Fonction	Chargé de Recherche du CNRS
Téléphone	(01 49 40 40 67)
Fax	(01 48 26 07 12)
Mél	pb@lipn.univ-paris13.fr

Autres membres de l'équipe participant au projet

Nom	Prénom	Poste statutaire	% du temps de recherche consacré au projet
MOGBIL	Virgile	Maître de Conférences	30 %

Références :

- P. Baillot. Checking polynomial time complexity with types. In *Foundations of Information Technology in the Era of Network and Mobile Computing (Proceedings IFIP TCS 2002)*. Kluwer Academic Press, 2002.
- P. Baillot and M. Pedicini. Elementary Complexity and the Geometry of Interaction. *Fundamenta Informaticae*, 45(1-2), pp.1-31, 2001.
- P. Baillot. Stratified coherent spaces : a denotational semantics for Light Linear Logic. to appear in *Theoretical Computer Science (Special Issue on ICC 2000)*.
- P. Baillot. Type inference for polynomial time complexity via constraints on words. Tech. report 2003-02, Laboratoire d'Informatique de Paris-Nord, january 2003. Available from <http://www-lipn.univ-paris13.fr/~baillot>.
- V. Mogbil. Quadratic correctness criterion for Non commutative Logic *Proceedings of the 15th International Workshop Computer Science Logic (CSL'01)*, LNCS 2142, pp.69-83, Springer Verlag, 2001
- V. Mogbil and T. Krantz Encoding Hamiltonian circuits into multiplicative linear logic *Theoretical Computer Science*, 266 (1-2) pp. 987-996, 2001.

B1 – Objectifs et contexte :

La maîtrise de la *complexité* des programmes est un aspect important de la sécurité informatique, dans le cadre de systèmes “*distribués, ouverts, mobiles, ubiquitaires...*”. Dans les systèmes embarqués par exemple, on doit composer avec des ressources limitées, et pour certaines applications, le fait de ne pas “sortir des bornes” peut avoir un caractère critique. Un autre cas est le code mobile : il est bien clair que le système hôte, qui offre des ressources de calcul pour le code migrant, doit obtenir des garanties sur le temps de calcul et l’espace mémoire requis, de façon à accepter ou refuser sur ce critère le code migrant, faute de quoi sa propre sécurité – intégrité de ses données, disponibilité en tant que serveur – serait mise en cause. Un autre cas encore où l’on aimerait avoir un contrôle sur la complexité des programmes est celui des applications développées sur les cartes à puces : le développeur d’une telle application devrait être capable de fournir des garanties quant à la demande en ressources utilisées, faute de quoi les fournisseurs de cartes ne pourront promouvoir, auprès de leurs divers clients, l’utilisation des cartes multi-applications qu’ils envisagent de développer, chaque client étant naturellement désireux de voir sa propre application s’exécuter normalement, sans en être empêchée par les autres.

Un autre aspect essentiel de la sécurité dans le type d’applications que nous venons de suggérer est celui du contrôle – du rejet, en fait – des *interférences* possibles entre applications concurrentes, ou entre l’application et le système hôte. Par “*interférence*” on entend le fait que des données privées, possédées par les applications ou le système hôte, pourraient être divulguées publiquement, et, de façon plus générale (et plus forte), que la connaissance de données privées pourrait influencer la nature des résultats publics fournis par un programme. Contrôler qu’il n’y a pas d’interférence entre divers composants d’un système va donc au delà du contrôle de la classique propriété de *confidentialité*. C’est un problème qui se pose par exemple dans le cas des cartes à puce multi-applications comme dans le cas du code mobile.

La recherche que nous nous proposons de mener se situe en amont des applications que nous avons suggérées : nous n’envisageons pas de donner, pour les quelques cas évoqués ci-dessus, ou d’autres, des solutions qui pourraient être utilisées dans un cadre commercial, pour la simple raison que ces solutions n’existent pas, même en théorie, à l’heure actuelle. On peut bien entendu traiter le problème du contrôle de la complexité en exerçant de façon dynamique (c’est-à-dire, en cours d’exécution) un contrôle sur les applications, leur allouant une quantité de ressources déterminée *a priori*. De même, dans le cas du contrôle des interférences, on pourrait imaginer interdire purement et simplement aux applications de coopérer, et donc de partager certaines données à l’exécution, ce qui serait évidemment une “solution” (radicale! Notons que la notion de “*sandbox*” utilisée en JAVA pour le code mobile relève d’une telle approche). Toutefois de telles techniques ne sont clairement pas des solutions satisfaisantes aux problèmes posés. En effet, l’objectif est, d’une part, de garantir que les applications s’exécutent de façon efficace – ce qui est contradictoire avec un contrôle de la consommation de ressources en cours d’exécution –, et, d’autre part, d’assurer que les programmes puissent coopérer en toute sécurité – ce qui nécessite d’aller au delà d’un simple contrôle des droits d’accès. Par conséquent, des méthodes d’*analyse statique* sont requises ici. De plus, il ne peut s’agir que de méthodes *formelles*, grâce auxquelles on peut démontrer a priori, en vérifiant éventuellement algorithmiquement cette démonstration, que les garanties visées sont bien apportées. L’objectif du projet proposé est donc d’étudier des méthodes mathématiques d’analyse statique des programmes, dans le cas de la complexité comme dans celui des interférences. Cette unité de méthode apparaît clairement dans le fait que l’approche par *typage* est utilisée dans les deux cas.

Cet objectif n’est, sur le plan scientifique, pas complètement nouveau : les diverses caractérisations qui ont pu être données de certaines classes de complexité sont évidemment une réponse au problème posé (le problème du contrôle des interférences n’a été quant à lui abordé que récemment). Toutefois, nous visons, sinon des caractérisations, du moins des “conditions suffisantes” s’agissant de la complexité comme de la non-interférence, qui soient applicables à la programmation “usuelle” – on en est assez loin pour le moment –, et vérifiables statiquement sans trop d’effort. De plus, comme nous l’avons suggéré ci-dessus, nous aimerions traiter plus particulièrement le cas de programmes *concurrents*, voire *mobiles*, et ceci à notre connaissance n’a pas été tenté jusqu’à présent. De façon plus précise, le style de programmation concurrente que nous nous proposons d’étudier est celui de la programmation *synchrone*. Nous pensons que ce style pourrait être bien adapté au cadre du “*global computing*”, en ce qu’il propose des ingrédients utiles (comme la réaction à l’absence d’un événement par exemple) pour affronter certaines caractéristiques d’un “environnement global” de calcul, comme la non-fiabilité des communications, les déconnexions temporaires, la dégradation des performances de certains sites, etc. De plus, ce style de programmation a une sémantique formelle, ce qui est évidemment une caractéristique essentielle pour notre projet.

Le projet le plus proche du nôtre dans ses objectifs est certainement MRG (“*Mobile Resource Guarantees*”) du programme européen Global Computing, mais il faut préciser que ce projet ne s’attaque pas aux aspects de concurrence, aux problèmes d’interférence et à la définition d’un langage de coordination

synchrone (même si ses résultats sont évidemment très intéressants, et spécialement pour nous). Nous reviendrons plus en détail ci-dessous sur l'état de l'art en matière d'analyse statique de la complexité des programmes, et du contrôle de la non-interférence ; il nous semble qu'une conclusion que l'on peut tirer de cet état de l'art est que l'approche que nous proposons est novatrice, puisqu'il s'agit pour nous de nous appuyer – tout en les développant – sur des résultats acquis dans le domaine de l'analyse statique de la complexité et de la non-interférence, pour les transférer et les étendre à une programmation concurrente et mobile, reposant sur le modèle synchrone.

B2 – Description du projet :

Le projet s'articule en 6 parties. Pour chaque partie nous indiquons un responsable et les sites participants. L'articulation des 6 parties est résumée en section 0.0.7.

0.0.1 Inférence automatique de bornes de complexité pour programmes fonctionnels

Responsable : Jean-Yves MARION. Sites participants : LORIA, LIF, LIPN.

L'extraction de bornes de complexité pour des langages fonctionnels (du premier ordre) a une longue histoire et des motivations variées. Dans une perspective fondamentale on est intéressé à des caractérisations de petites classes de complexité par des *algèbres de fonctions* (voir [Cl95] pour un survol). Un premier exemple est la caractérisation due à Cobham [Co65] du temps polynomial par le biais de la notion de récursion bornée sur la notation.

La caractérisation de Cobham est basée sur des définitions par récursion primitive sur une notation binaire avec la restriction que la taille du résultat de la fonction définie est bornée explicitement par un polynôme. Du point de vue de la programmation, une limite de la récursion bornée sur la notation est que le programmeur doit *trouver* la borne polynomiale tout en définissant une fonction par *récursion primitive*. En d'autres termes la récursion bornée sur la notation n'offre pas d'aide à la recherche automatique de bornes et en plus elle impose l'utilisation de la récursion primitive.

Il y a quelques années, Bellantoni-Cook [BC92] et Leivant [Le94] ont introduit une notion de *ramification*. Dans [BC92] cette notion est exprimée par une distinction entre arguments *normaux* et *sûrs* et une restriction dans la façon dans laquelle la récursion primitive peut être appliquée et les fonctions composées ([Le94] introduit une notion proche de *niveau*). En se conformant à cette discipline de programmation, le programmeur est dispensé du problème de trouver explicitement une borne. La borne est implicite dans les contraintes imposées par la ramification et si nécessaire elle peut être calculée explicitement. En suivant ces idées, Marion [Ma01] a proposé une arithmétique de temps polynomial c'est-à-dire pour laquelle toutes les fonctions prouvablement totales sont PTIME. Par rapport aux autres propositions dans ce domaine ce système offre l'avantage d'avoir une formulation très naturelle.

La *ramification* ne règle pas tous les problèmes. Il a été observé [Ca97] que cette discipline de programmation exclut plusieurs algorithmes naturels. Pour améliorer la situation, une approche suivie par Marion *et al.* [Ma00, MM00, BMM01] a été d'étendre la notion de ramification à différents types d'*ordres récursifs sur les chemins* (une famille d'ordres de simplification qui comprend l'ordre lexicographique et par multi-ensemble, ...). Dans une direction différente, Jones [Jo97] et Hofmann [Ho02] ont proposé de considérer des programmes avec récursion générale pour lesquels on peut trouver une borne sur la taille des données utilisées pendant le calcul. Jones obtient une telle borne par une restriction syntaxique (sévère) sur la forme des programmes, alors que Hofmann introduit un système de typage "linéaire" équipé d'un type *ressource* qui assure que les valeurs calculées n'augmentent pas de taille (*non-size increasing*).

La notion de *quasi-interprétation* a été proposée par Marion *et al.* dans le contexte du travail sur les ordres récursifs *ramifiés* sur les chemins mentionnés ci-dessus. Plus précisément, les quasi-interprétations sont extraites d'une preuve de terminaison dans l'ordre récursif ramifié et elles sont alors utilisées pour borner l'espace nécessaire au calcul du résultat du programme. Les quasi-interprétations sont inspirées par les *interprétations polynomiales* qui comptent parmi les outils traditionnels pour démontrer la *terminaison* de systèmes de réécriture de termes (voir par exemple [BN98]). La limite des (quasi-)interprétations est que d'abord il faut synthétiser une et ensuite il faut vérifier qu'elle est bien adaptée au système de réécriture en question (en général même le problème de la vérification est indécidable par réduction du dixième problème de Hilbert).

Nous nous proposons d'étudier le problème de la synthèse automatique de quasi-interprétations pour des programmes avec récursion générale. A cause des problèmes de complexité mentionnés ci-dessus, il semble nécessaire de se limiter à un espace de petits polynômes. Une proposition récente est de se restreindre aux polynômes à plusieurs variables sur l'algèbre max-plus [BCOQ92]. Dans [Am02] il est montré que dans ce cas le problème de la synthèse est NP-dur et dans NP dans le cas particulier des polynômes multi-linéaires. Ces derniers ont déjà un pouvoir d'expression considérable et ils généralisent

0.0.2 Complexité, ordre supérieur et logique linéaire

Responsable : Patrick BAILLOT. Sites participants : LIPN, LORIA.

Les techniques présentées dans la section précédente traitent essentiellement des langages fonctionnels du premier ordre. Une approche particulièrement prometteuse pour une extension de ces résultats à l'ordre supérieur est issue de la théorie de la démonstration de la *logique linéaire*. Cette logique repose sur l'idée de contrôler la duplication d'arguments par le biais de *modalités*. En considérant des règles pour les modalités plus sévères que celles de la logique linéaire, induisant une utilisation modérée de la duplication d'arguments, des systèmes logiques avec une dynamique interne (élimination des coupures) PTIME ont été proposés, comme la *logique linéaire (ou affine) "light"* [Gi98, AR02] ou la *logique linéaire "soft"* [La02]. Ces logiques fournissent des systèmes de types polymorphes pour le lambda-calcul qui garantissent des bornes de complexité en temps : si un programme est bien typé, alors il est PTIME.

Une des difficultés pour l'utilisation de la logique linéaire "légère" en programmation réside dans la sophistication des types qu'elle définit. Le typage est par conséquent délicat et ne peut être laissé à charge de l'utilisateur. Partant de cette constatation nous avons étudié le problème de l'inférence de type pour le lambda-calcul en logique légère [Ba02, Ba03]. Nous avons montré la décidabilité de ce problème (existence de type) dans le cas sans polymorphisme en le ramenant à un problème de résolution de contraintes, consistant en des systèmes d'inéquations sur les mots. Ces contraintes peuvent être ramenées à un problème de programmation entière dans le cas où une partie de l'information de typage est fournie avec le terme [Ba02].

Les problèmes restant ouverts ici concernent la complexité de cette inférence de type, ainsi que sa modularité ; en particulier une notion de type principal (à la manière de celle proposée pour la *logique linéaire élémentaire* dans [CR03]) pourrait être utile. D'autre part la question de l'intégration du polymorphisme, éventuellement sous une forme restreinte, doit être étudiée. Nous souhaitons à partir des techniques décrites dans l'état de l'art ci-dessus étudier des systèmes qui permettent à la fois de caractériser des algorithmes courants et d'avoir une programmation suffisamment générale (réutilisation de fonctions, ordre supérieur, polymorphisme). Une première possibilité sera de chercher à intégrer les méthodes du typage linéaire de Hofmann mentionnées en section 0.0.1 avec les caractéristiques de la logique "light" (ordre supérieur, réutilisation de fonctions). La logique linéaire "soft" de Lafont ([La02]) pourrait aussi s'avérer utile dans cette voie car elle repose sur l'idée de contrôler statiquement l'espace disponible pour le calcul.

0.0.3 Langage de coordination pour systèmes synchrones

Responsable : Gérard BOUDOL. Sites participants : INRIA-Sophia, LIF.

La programmation utilisant les "threads" classiques, en JAVA par exemple, est extrêmement difficile et sujette à erreurs, pour une raison essentielle qui est que le programmeur ne maîtrise pas leur ordonnancement. La sémantique du programme apparaît donc comme non-déterministe, ce qui rend la mise au point quasiment impossible (pas de "replay"), et oblige le programmeur à installer des verrous dans son code pour protéger la manipulation des données partagées. Le code ne peut donc être modulaire – on ne peut réutiliser du code qui n'est pas conçu pour le "multi-threading" –, il est aussi inefficace – à cause des protections sur les données –, et les risques d'inter-blocage sont grands. De plus, la coordination entre les threads est illusoire, car le résultat dépend en fait de l'ordonnancement choisi par le "scheduler".

La programmation *synchrone* offre une alternative au "multi-threading" classique : le parallélisme y est déterministe, la communication par événements diffusés rend le code plus modulaire, la synchronisation y est déterminée par le code des "threads" eux-mêmes. Toutefois, pour être adapté au code mobile, qui est le scénario envisagé dans ce projet, l'ajout dynamique de "threads" doit être possible, ce que n'autorise pas la programmation en ESTEREL par exemple, où cela crée des problèmes de causalité. Nous adopterons donc le style de la programmation "réactive"⁽²⁾ de [Bo96], où la réaction à l'absence est plus faible qu'en ESTEREL : elle n'a pas lieu dans l'instant, mais est reportée à l'instant suivant. Cette simple modification suffit à éliminer tout problème de causalité, et autorise la création ou l'ajout dynamique de "thread". D'autre part, nous pensons que les primitives de la programmation synchrone (à la Boussinot) sont bien adaptées pour la programmation du code mobile, et plus généralement de systèmes avec un parallélisme dynamique – chargement de nouvelles applications sur une carte à puce

²que nous appellerons néanmoins *synchrone*, terme qui pour nous signifie "partager une échelle de temps commune". Selon [CP96], un programme est *réactif* s'il a la propriété de réagir en un temps fini à des événements, propriété que nous souhaitons évidemment obtenir, mais qui n'est pas nécessairement donnée par la sémantique du langage considéré.

par exemple. En effet, la communication par événements diffusés permet de programmer du code qui peut s’intégrer aisément dans des contextes variés, et d’autre part la possibilité de réaction à la présence ou l’absence d’un événement permet de programmer du code adapté à un environnement incertain, qui est une caractéristique majeure d’un “réseau global” comme le web.

L’un de nos objectifs – que nous pensons atteindre rapidement – est donc la conception d’un noyau de langage pour le code mobile, et plus généralement pour le parallélisme dynamique, reposant sur le style de la programmation synchrone. Nous devons dépasser le cadre des systèmes à états finis tels qu’on les programme en ESTEREL ou en LUSTRE par exemple, car nous voulons pouvoir maîtriser aussi, du point de vue de la complexité et de la sécurité, les calculs que peuvent faire les programmes, au delà de l’aspect de synchronisation pure. Notre objectif est donc d’étendre le noyau de la programmation synchrone, et plus précisément la variante “réactive” de [Bo96] – i.e. la manipulation du parallélisme et des événements – à un langage enrichi avec des constructions de la programmation impérative et fonctionnelle (nous pensons à un noyau semblable à celui de ML ou de SCHEME). Nous devons ici prendre en compte ce qui a été fait sur la complexité des programmes fonctionnels, puisque l’un de nos objectifs est d’étendre les résultats obtenus dans ce cadre à un langage avec du parallélisme (dynamique) et de la synchronisation (voir section 0.0.5) – un autre objectif étant d’étendre le travail fait sur le typage de la non-interférence à ce langage (voir section 0.0.4).

0.0.4 Non-interférence pour systèmes synchrones

Responsable : Ilaria CASTELLANI. Sites participants : INRIA-Sophia, LIF.

Le problème de la non-interférence est le suivant : étant donnée une politique de sécurité, qui prend la forme d’un treillis de niveaux de sécurité, il s’agit de trouver des moyens de garantir que l’exécution d’un programme ne peut donner lieu à des fuites d’information d’un niveau de sécurité vers un autre qui ne lui est pas supérieur. Par exemple, on veut garantir que deux *applets* partageant des données mais ayant également chacune des données *privées* – donc de niveaux de sécurité incomparables – puissent coopérer sans qu’il y ait une fuite de données privées de l’une vers l’autre. Ce problème est posé depuis longtemps, mais a trouvé assez récemment une solution très élégante, avec les travaux de Volpano et Smith qui ont formalisé une technique d’analyse due à Denning sous la forme d’un système de types, obtenant ainsi une preuve de correction de cette analyse.

Dans [BC01], puis dans [BC02], nous avons raffiné la technique de Volpano et Smith [VSI96, SV98] qui, dans le cas de programmes concurrents, était beaucoup trop restrictive. Notre idée a été de tenir compte dans les types, non seulement du niveau de sécurité des variables affectées par un programme – c’était l’idée originale de Volpano et Smith [VSI96] –, mais encore du niveau des variables testées (dans un branchement conditionnel ou une boucle *while*). La contrainte que l’on impose alors est que l’on ne peut affecter une variable que d’un niveau supérieur ou égale à celui des variables préalablement testées. Nous montrons que ce système de types assure bien la propriété souhaitée – absence de fuite –, tout en étant moins restrictif que celui de Volpano et Smith [SV98]. Le même système de types a été – fait remarquable – découvert indépendamment par Smith lui-même, et publié à peu près simultanément [Smi01]. Dans [BC02], nous étendons ce système de types au cas où des programmes (séquentiels) concurrents sont soumis à un ordonnancement. Nous montrons comment modéliser cette situation, qui en général introduit de nouvelles possibilités de fuites, par de nouvelles primitives de synchronisation, et nous généralisons le résultat montrant que les programmes typables sont sûrs. A notre connaissance, c’est le seul résultat sur cette question qui n’utilise pas une approche probabiliste, et qui permette de classifier, par typage, les mécanismes d’ordonnancement.

Notre objectif est d’étendre ces techniques de typage au modèle de programmation synchrone discuté en section 0.0.3. Dans un premier temps, nous utiliserons l’idée des types de [BC02] pour le cas d’un langage très simple, où l’on a quelques constructions de la programmation impérative – celles qui sont considérées dans [VSI96] –, ainsi que celles de la programmation synchrone (ou plutôt “réactive” [Bo96]), c’est-à-dire la composition parallèle et les primitives permettant de manipuler les événements. Ensuite, nous étendrons les résultats obtenus au langage évoqué dans la section 0.0.3, c’est-à-dire au langage obtenu en ajoutant les fonctions. Le cas du noyau de langage impératif et fonctionnel (le noyau de ML), sans parallélisme ni synchronisation, a été traité précédemment par Pottier et Simonet [PS03]; notre objectif est donc d’étendre également ce dernier résultat.

0.0.5 Contrôle de ressources pour systèmes synchrones

Responsable : Silvano DAL ZILIO. Sites participants : LIF, INRIA-Sophia, LIPN.

Dans la section 0.0.1 nous avons évoqué un certain nombre de méthodes qui permettent d’inférer automatiquement des bornes sur les ressources utilisées par un code. Ces méthodes se focalisent essen-

tiellement sur un modèle de programmation séquentiel : un programme prend une entrée et retourne un résultat. Notre objectif est de reconsidérer la situation dans le cadre plus général d'un système de processus séquentiels et communicants. En particulier, nous imaginons la situation dans laquelle certains de ces processus exécutent du code chargé dynamiquement pendant l'exécution. La question est alors de prévoir comment les ressources nécessaires au fonctionnement du système vont dépendre des ressources nécessaires à chaque processus.

Une étude préliminaire de ce problème portera sur le modèle des réseaux de Kahn [Ka74], dans lequel chaque processus est vu comme un noeud d'un réseau qui transforme un flot de messages en entrée en un flot de messages en sortie. Les réseaux de Kahn peuvent être implémentés de façon simple et efficace. En particulier, plusieurs auteurs ont considéré le problème de la construction d'un ordonnancement qui assure la vivacité du réseau et l'exécution en mémoire bornée (voir, par exemple, [LM87, Ca92, Bu93, Pa95, CP96]). Par ailleurs, les réseaux de Kahn s'intègrent très bien à un style de programmation fonctionnel : les flots sont un cas particulier de type co-inductif et la transformation de flots peut être représentée par du filtrage (voir [CP96, Po02, HPS96] pour des exemples d'intégration). Notons que les travaux que nous venons de mentionner font toujours l'hypothèse que la taille des messages qui circulent dans le réseau est bornée à priori. Cette hypothèse est liée au fait que les noeuds du réseau sont vus comme des transducteurs finis. Nous nous intéressons au cas plus général où les messages peuvent être d'un type inductif arbitraire (listes, arbres, ...). Dans ce cas, une analyse supplémentaire est nécessaire pour s'assurer que si la taille des entrées est bornée alors la taille des messages qui circulent dans le réseau est aussi bornée.

Comme nous venons de le voir, les réseaux de Kahn peuvent être vus comme un exemple de langage de coordination pour systèmes synchrones où l'on peut imaginer que le code associé à certains noeuds du réseau est chargé dynamiquement. Une limitation du modèle est que la topologie du réseau de communication est déterminée à l'avance ce qui implique, par exemple, qu'il faut borner *a priori* le nombre de noeuds où le code mobile pourra être exécuté. Ainsi, notre objectif principal ici sera d'adapter les méthodes pour le contrôle de ressources au langage de coordination esquissé en section 0.0.3, en nous concentrant d'abord sur la coordination de modules fonctionnels, ce qui devrait nous permettre d'adapter le travail décrit en section 0.0.1.

0.0.6 Machine virtuelle et génération de certificats

Responsable : Line JAKUBIEC. Sites participants : LIF, LORIA.

Dans les dernières années la notion de code certifié (*proof-carrying code*) [Ne97] s'est révélée comme une méthode particulièrement intéressante pour assurer la sécurité du code mobile. L'idée de base est que le producteur du code associe un certificat (une preuve) au code envoyé et que ce certificat peut être facilement vérifié par le consommateur avant que le code associé ne soit exécuté. Dans cette méthode, la tâche – ardue – de génération du certificat revient au producteur de code. La recherche s'est donc concentrée sur des méthodes qui permettent d'automatiser ce travail. Il semble clair qu'en général il est plus aisé d'analyser un code source qui est rédigé dans un langage haut niveau. Cependant, il est souhaitable que le code transmis soit rédigé dans le langage bas niveau d'une machine virtuelle. Ce souhait répond à la fois à des raisons d'efficacité (il n'est pas nécessaire de recompiler) et de sécurité (le producteur du code ne pourra pas exploiter, de façon volontaire ou involontaire, les faiblesses du processus de compilation). Ainsi le problème revient à effectuer d'abord une analyse automatique du code source et puis à en dériver un certificat pour le code objet.

Les premières expériences menées à Carnegie Mellon University par Lee, Necula *et al.* sur la génération de certificats se sont focalisées d'abord sur l'intégrité de l'environnement d'exécution. Ensuite, des variantes de l'approche ont été développées comme le langage assembleur typé de Morriset *et al.* [MWCG99] (*typed assembly language*) et une approche réductionniste à la notion de certificat de Appel *et al.* [Ap02] (*foundational proof-carrying code*). En Europe, depuis un an, le projet MRG ("Mobile Resource Guarantees") du programme européen Global Computing, mené par les Universités de Munich et d'Edimbourg, développe une plate-forme pour la génération de certificats pour un sous ensemble de la JVM (Java Virtual Machine) en se focalisant en particulier sur le contrôle de ressources. Notons qu'à notre connaissance, la question de la production de certificats pour la propriété de non-interférence n'a pas encore été abordée.

Une conclusion qu'on peut tirer de ces expériences est que le développement d'une plate-forme pour la certification du code d'une machine virtuelle avec des caractéristiques proches de celles de la commercialisation demande un investissement très important avec des retombées scientifiques difficiles à évaluer. Nous nous proposons de mener notre travail d'expérimentation sur un modèle de machine virtuelle extrêmement simplifié en nous concentrant sur le contrôle de ressources et la non-interférence. Le but ici est d'avoir une plate-forme d'expérimentation dont l'intérêt est plutôt pédagogique que commercial. Nous nous appuyons sur notre expérience avec le système de preuve COQ et sur son utilisation

AGI SÉCURITÉ INFORMATIQUE PROJET OBISSE dans la formalisation de machines virtuelles [BDJ00, BDJ01, CG02, CGN02, CGJ99] ainsi que sur les modèles et les techniques d'inférence automatique développées dans le projet (voir sections 0.0.1, 0.0.2 et 0.0.4). Le travail que nous nous proposons d'entreprendre se décompose en trois parties :

- La définition d'une machine virtuelle. La machine doit être suffisamment expressive pour implémenter notre langage de programmation. Typiquement, pour un langage fonctionnel du premier-ordre on considérera une variante de la machine de Landin/Krivine.
- La formalisation des propriétés souhaitées par un code de la machine virtuelle. Il s'agit par exemple de définir exactement quelles sont les ressources (temps, mémoire, ...) utilisées par un code pendant son exécution par la machine virtuelle.
- La génération automatique de certificats. Étant donné un programme P qui satisfait une propriété A d'après une certaine analyse statique π il s'agit de générer une preuve COQ $\langle \pi \rangle$ qui montre que le code objet $\langle P \rangle$ résultant de la compilation satisfait en effet la propriété A (ou sa traduction en termes de machine virtuelle).

0.0.7 Articulation du projet

En résumant, les 6 parties du projet sont :

1. Inférence automatique de bornes de complexité pour les programmes fonctionnels.
2. Complexité, ordre supérieur et logique linéaire.
3. Langage de coordination pour systèmes synchrones.
4. Non-interférence pour systèmes synchrones.
5. Contrôle de ressources pour systèmes synchrones.
6. Machine virtuelle et génération de certificats.

La partie 2 étend et complète la partie 1. Les résultats des parties 1-2 sont utilisés dans la partie 5. Les parties 4 et 5 dépendent du langage défini dans la partie 3. La méthodologie de génération de certificats développée dans 6 pourra s'appliquer *mutatis mutandis* à 1-2, 4, et 5.

0.0.8 Références bibliographiques

- [ABL02] R. Amadio, G. Boudol and C. Lhoussaine. The distributed receptive π -calculus. *ACM Transactions of Programming Languages and Systems (TOPLAS)*, 2003 (to appear).
- [ACG98] R. Amadio R. and S. Coupet-Grimal. Analysis of a guard condition in type theory (extended abstract). In *Proc. FOSSACS, ETAPS 98, Springer Lect. Notes in Comp. Sci. 1378*. Springer, pp. 48–62.
- [Am02] R. Amadio. Max-plus quasi-interpretations. In *Proc. Typed Lambda Calculi and Applications (TLCA) 2003, Valencia*, Springer Lecture Notes in Computer Science, to appear.
- [Am00] R. Amadio. On modelling mobility. *Theoretical Computer Science*, 240 :147-176, 2000.
- [AM02] R. Amadio R. and C. Meyssonier. On decidability of the control reachability problem in the asynchronous π -calculus. *Journal of Nordic Computing*, 9(2), pp.70–101.
- [AP00] R. Amadio and S. Prasad. Modelling IP mobility. *J. of Formal Methods in System Design*, 17(1), pp.61–99.
- [Ap02] A. Appel. Foundational proof-carrying code. In *Proc. LICS Conference, ACM-IEEE*, 2002.
- [AR02] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1) :1 – 39, January 2002.
- [As98] A. Asperti. Light affine logic. In *"Proceedings of the 13th Symposium on Logic in Computer Science"*. IEEE Computer Society, 1998.
- [Ba02] P. Baillot. Checking polynomial time complexity with types. In *Foundations of Information Technology in the Era of Network and Mobile Computing (Proceedings IFIP TCS 2002)*. Kluwer Academic Press, 2002.
- [Ba02a] P. Baillot. Stratified coherent spaces : a denotational semantics for Light Linear Logic. to appear in *Theoretical Computer Science (Special Issue on ICC 2000)*.
- [Ba03] P. Baillot. Type inference for polynomial time complexity via constraints on words. Tech. report 2003-02, Laboratoire d'Informatique de Paris-Nord, january 2003. available from <http://www-lipn.univ-paris13.fr/~baillot>.
- [BC92] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2 :97–110, 1992.
- [BC01] G. Boudol and I. Castellani. Noninterference for Concurrent Programs. In *Proc. ICALP*, Springer Lecture Notes in Comp. Sci., 2001.
- [BC02] G. Boudol and I. Castellani. Noninterference for Concurrent Programs and Thread Systems. *Theoretical Computer Science*, , 281(1) :109-130, 2002.

- [BCO92] F. Baccelli, G. Cohen, G. Oster, and J.-P. Quadrat. *Synchronization and linearity*. Wiley, 1992. 14
- [BDJ00] G. Barthe, G. Dufay, L. Jakubiec, B. Serpette et S. de Sousa. Formalization in Coq of the Java Card Virtual Machine. In *Proceedings of FTfJP'00-ECOOP Workshop on Formal Techniques for Java Programs*. U. Haagen.
- [BDJ01] G. Barthe, G. Dufay, L. Jakubiec, B. Serpette et S. de Sousa. A Formal Executable Semantics of the Java Card Platform. In *Proceedings of ESOP'01*. D. Sands (Ed.). Springer LNCS, 2001.
- [BDZ99] G. Boudol and S. Dal Zilio. An Interpretation of Extensible Objects. In *Proc. International Symposium on Fundamentals of Computation Theory (FCT '99), Springer Lect. Notes in Comp. Sci. 1684*.
- [BMM01] G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. On termination methods with space bound certifications. In *Andrei Ershov Fourth International Conference "Perspectives of System Informatics"*, Lecture Notes in Computer Science. Springer, 2001.
- [BN98] F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.
- [BNS00] S. Bellantoni, K.-H. Niggl, and H. Schwichtenberg. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic*, 104(1-3), 2000.
- [Bo96] F. Boussinot. *La programmation Réactive*. Masson, 1996.
- [Bo98] G. Boudol. The π -calculus in direct style. *Higher-Order and Symbolic Computation* 11 (1998).
- [Bo03] G. Boudol. The recursive record semantics of objects revisited. *J. of Functional Programming*, 2003 (to appear).
- [BP01] P. Baillot and M. Pedicini. Elementary Complexity and the Geometry of Interaction. *Fundamenta Informaticae*, 45(1-2), pp.1-31, 2001.
- [Bu93] J. Buck. *Scheduling dynamic dataflow graphs with bounded memory using the token flow model*. PhD thesis, University of Berkeley, 1993.
- [BZ02] G. Boudol, P. Zimmer. Recursion in the call-by-value λ -calculus. *FICS* (2002).
- [Ca97] V. Caseiro. *Equations for defining polytime functions*. PhD thesis, University of Oslo, 1997.
- [Ca92] P. Caspi. Clocks in data flow languages. *Theoretical Computer Science*, 94 :125-140, 1992.
- [CG02] Coupet-Grimal S. An Axiomatization of Linear Temporal Logic in the Calculus of Inductive Constructions (Part 1). *The Journal of Logic and Computation*, to appear.
- [CGN02] S. Coupet-Grimal and C. Nouvet. Formal Verification of an Incremental Garbage Collector (Part 2). *The Journal of Logic and Computation*, to appear.
- [CGJ99] S. Coupet-Grimal and L. Jakubiec. Hardware Verification using Co-induction in Coq. In *TPHOLs'99*. LNCS 1690, Springer.
- [Cl95] P. Clote. Computation models and function algebras. In *Proc. Logic and computational complexity, Springer Lecture Notes in Comp. Sci. 960*, 1995.
- [Co65] A. Cobham. The intrinsic computational difficulty of functions. In *Proc. Logic, Methodology, and Philosophy of Science II, North Holland*, 1965.
- [Co71] S. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the ACM*, 18(1) :4-18, 1971.
- [CP96] P. Caspi and M. Pouzet. Synchronous Kahn networks. In *Proc. ACM Conf. on Functional Programming*, 1996.
- [CR03] P. Coppola and S. Ronchi della Rocca. Principal typing in Elementary Affine Logic. In *Proceedings TLCA 2003*, LNCS. Springer Verlag, 2003. to appear.
- [DZG02] S. Dal Zilio and A. Gordon. Region analysis and a π -calculus with groups. *Journal of Functional Programming*, 12(3), pp.229-292.
- [DZ00] S. Dal Zilio. An Interpretation of Typed Concurrent Objects in the Blue Calculus. In *Proc. International Conference on Theoretical Computer Science (IFIP TCS 2000), Springer Lect. Notes in Comp. Sci. 1872*. Springer.
- [FGK01] O. Fissore, and I. Gnaedig and H. Kirchner. Termination of rewriting with local strategies. In *Proc. of the IJCAR Workshop Strategies in Automated Deduction*, Gramlich and Bonacina (eds), 2001. Electronic Notes in Theoretical Computer Science, volume 58.
- [Ge69] H. Genrich. Das Zollstationenproblem. Technical report, St. Augustin : Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF/69-01-15, Revised version ISF/71-10-13, 1969.
- [Gi87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50 :1-102, 1987.
- [Gi98] J.-Y. Girard. Light linear logic. *Information and Computation*, 143 :175-204, 1998.
- [HC70] A. Holt and F. Commoner. Events and conditions. Technical report, Princeton, N. J. : Applied Data Research Inc., Information System Theory Project, 1970.
- [HJ03] M. Hofmann and S. Jost. Static prediction of heap space usage for first-order functional programs. In *Proc. ACM POPL*, 2003.
- [Ho00] M. Hofmann. A type system for bounded space and functional in-place update. *Nordic Journal of Computing*, 7(4) :258-289, 2000.

- [Ho02] M. Hofmann. The strength of non size-increasing computation. In *Proc. ACM POPL*, 2002.
- [Ho99] M. Hofmann. Linear types and non-size-increasing polynomial time computation. In *"Proceedings of the 14th Symposium on Logic in Computer Science"*. IEEE Computer Society, 1999.
- [Ho00a] M. Hofmann. Safe recursion with higher types and BCK-algebra. *Annals of Pure and Applied Logic*, 104(1-3), 2000.
- [HPS96] R. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *Proc. ACM POPL*, 1996.
- [Im99] N. Immerman. *Descriptive complexity*. Springer, 1999.
- [Jo97] N. Jones. *Computability and complexity, from a programming perspective*. MIT-Press, 1997.
- [Ka74] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. IFIP 74 Congress, North-Holland*, 1974.
- [KG00] H. Kirchner and I. Gnaedig. Termination and normalisation under strategies – Proofs in ELAN. *Third International Workshop on Rewriting Logic and Applications, WRLA'2000*, Kanazawa (Japan), Electronic Notes in Theoretical Computer Science, volume 36, 2001.
- [Ja99] L. Jakubiec. *Vérification de Circuits dans Coq*. Université de Provence, 1999.
- [La02] Y. Lafont. Soft Linear Logic and Polynomial Time. *Theoretical Computer Science* (to appear).
- [Le94] D. Leivant. Predicative recurrence and computational complexity i : word recurrence and poly-time. *Feasible mathematics II, Clote and Remmel (eds.)*, Birkhäuser :320–343, 1994.
- [LM87] E. Lee and D. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Trans. on Computers*, 1 :24–35, 1987.
- [Ma01] J.-Y. Marion. Actual arithmetic and feasibility. In L. Fribourg, editor, *International Workshop on Computer Science Logic - CSL'2001, Paris, France*, volume 2142 of *Springer Lecture notes in Computer Science*, pages 115–129, 2001.
- [MM00] J.-Y. Marion and J.-Y. Moyen. Efficient first order functional program interpreter with time bound certifications. In *LPAR*, volume 1955 of *Lecture Notes in Computer Science*, pages 25–42. Springer, Nov 2000.
- [Ma00] J.-Y. Marion. *Complexité implicite des calculs, de la théorie à la pratique*. Université de Nancy, 2000. Habilitation à diriger des recherches.
- [MK01] V. Mogbil and T. Krantz. Encoding Hamiltonian circuits into multiplicative linear logic. *Theoretical Computer Science*, 266 (1-2) pp. 987-996, 2001.
- [Mo01] V. Mogbil. Quadratic correctness criterion for Non commutative Logic. *Proceedings of the 15th International Workshop Computer Science Logic (CSL'01)*, LNCS 2142, pp.69-83, Springer Verlag, 2001
- [MWCG99] G. Morriset, D. Walker, K. Cray and N. Glew. From System F to Typed Assembly Language. In *ACM Transactions on Programming Languages and Systems*, 21(3) :528-569, 1999.
- [Ne97] G. Necula. Proof carrying code. In *Proc. ACM POPL*, 1997.
- [Pa00] L. Pareto. *Types for crash prevention*. PhD thesis, Chalmers University of Technology, 2000.
- [Pa95] Th. Park. *Bounded scheduling of process networks*. PhD thesis, University of Berkeley, 1995.
- [Po02] M. Pouzet. *Lucid Synchrone*. PhD thesis, Université Paris 6, 2002. Habilitation.
- [PS03] F. Pottier, V. Simonet. *Information flow inference for ML*. *ACM Transactions on Programming Languages and Systems* 25, 2003.
- [Pu92] W. Pugh. The omega test : a fast and practical integer programming algorithm for dependence analysis. In *Communications of the ACM*, 102-114, 1992.
- [Ro00] L. Roversi. Light affine logic as a programming language : a first contribution. *International Journal of Foundations of Computer Science*, 11(1), 2000.
- [Sa01] D. Sannella. Mobile resource guarantee. Ist-global computing research proposal, U. Edinburgh, 2001. <http://www.dcs.ed.ac.uk/home/mrg/>.
- [Smi01] G. Smith. *A new type system for secure information flow*. Computer Security Foundations Workshop, 2001.
- [SV98] G. Smith, D. Volpano. *Secure information flow in a multi-threaded imperative language*. POPL, 1998.
- [Te01] K. Terui. Light Affine Lambda-calculus and polytime strong normalization. In *Proceedings LICS'01*. IEEE Computer Society, 2001.
- [VSI96] D. Volpano, G. Smith, C. Irvine. *A sound type system for secure flow analysis*. *J. of Computer Security* 4, 1996.
- [WB00] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In *Proc. TACAS, Springer Lecture Notes in Comp. Sci. 1785*, 2000.

Une activité de recherche fondamentale est prévue dans chaque partie de la proposition pendant toute la durée du projet. L'exception à la règle est la partie 0.0.3 (langage de coordination) qui doit impérativement être achevée pendant la première année pour permettre le démarrage des parties 0.0.4 (non-interférence pour le synchrone) et 0.0.5 (contrôle de ressources pour le synchrone) et rester ensuite relativement stable. Ce travail de définition du modèle sera basé sur les expériences menées à l'INRIA Sophia sur la programmation synchrone. Nous mentionnons dans la suite un échéancier qui concerne surtout le volet expérimental et technologique de notre projet.

0.0.9 Première année

- Définition du langage de coordination pour systèmes synchrones (voir section 0.0.3).
- Implantation d'une technique d'inférence automatique de bornes de complexité pour des programmes fonctionnels (voir section 0.0.1).
- Définition en Coq d'une machine virtuelle pour un langage fonctionnel et des ressources nécessaires à l'exécution d'un code (voir section 0.0.6)

0.0.10 Deuxième année

- Définition d'un système de typage qui garantit la non-interférence pour le langage de coordination (voir section 0.0.4).
- Extension des techniques d'inférence automatique de bornes de complexité au langage de coordination (voir section 0.0.5) et implantation de techniques d'inférence pour l'ordre supérieur (voir section 0.0.2).
- Génération automatique de certificats pour le code de la machine virtuelle développée dans la première année (voir section 0.0.6).

0.0.11 Troisième année

- Implantation de l'analyse statique du langage de coordination qui assure la non-interférence (voir section 0.0.4).
- Implantation de l'analyse statique du langage de coordination qui contrôle les ressources (voir section 0.0.5).
- Extension des techniques de génération automatique de certificats aux propriétés de non-interférence (voir section 0.0.6).

Le projet CRISS est issu du volet *code mobile* de l'action spécifique *Modèles formels pour la mobilité* dont Roberto AMADIO est le coordinateur et qui compte le LIF Marseille et le LIPN Villetaneuse parmi les participants. Par ailleurs, il y a une longue histoire de collaboration entre le LIF Marseille et l'INRIA-Sophia : certains membres du LIF participent au projet INRIA MIMOSA. Enfin, une collaboration informelle entre le LIF Marseille, le LORIA Nancy et le LIPN Villetaneuse sur le thème du contrôle de ressources est en cours depuis un an.

Le projet CRISS se base sur certains atouts de la recherche française en informatique fondamentale (logique linéaire), langages synchrones (Esterel), assistants de preuve (Coq) et les applique au problème de la sécurité du code mobile. Le projet vise à tester la pertinence de certaines méthodes d'analyse statique pour le contrôle de ressources et pour la non-interférence dans le cadre d'un environnement de programmation synchrone. Certaines de ces méthodes pourraient être mises en oeuvre dans le contexte de l'environnement de programmation basé sur SCHEME qui est développé dans le projet MIMOSA. Les résultats du projet seront diffusés au niveau européen par le biais de l'initiative *Global Computing* à laquelle le projet MIMOSA participe (projets Mikado et Profundis).

B4 – Summary (in English) :

Research on mobile code has been a hot topic since the late 90's with many proposals building, *e.g.* on JAVA platforms. Security issues are one of the fundamental problems that still have to be solved before mobile code can become a well-established and well-accepted technology. Application scenarios may include, for instance, programmable switches, network games, and applications for smart cards. Initial proposals have focused on the integrity of the execution environment (no memory faults). We identify two additional key properties that one should be able to guarantee of a mobile code :

- Bounds on the (computational) resources needed for the execution of the program, in order to avoid so called *denial of service* attacks.
- Limits on the information flow, in order to avoid the *leaking of confidential data* (non-interference).

These problems have already attracted considerable attention. Automatic extraction of resource bounds has mainly focused on (first-order) functional languages starting from Cobham's characterization of polynomial time functions by bounded recursion on notation. Here we explore various inference techniques that allow for efficient analyses while capturing a sufficiently large range of practical algorithms. A second concern, is the extension of these techniques to handle higher-order functions for which we rely on ideas from (light) linear logic.

Work on information flow and non-interference has started with Denning's flow analysis and, later, Volpano *et al.* have provided an elegant formalization and proof of correctness that relies on a type system. Here we have considered the extension of these techniques to systems of concurrent threads and we have found that an explicit analysis of the scheduling policy is needed.

While planning to advance the state of the art on these two problems, we also wish to test their applicability in the context of a realistic programming environment. In this respect, we think that in order to build reliable and predictable systems one has to go beyond the usual model of asynchronous threads endowed with point-to-point communication. Thus, a third goal of our project is to design a model for the coordination of sequential modules that allows for synchronous execution, broadcast communication, and deterministic behaviour.

This design will build on previous work on synchronous Kahn networks, the ESTEREL programming language, and its extensions to a full fledged programming language developed in the MIMOSA project. The dynamic reconfiguration of the communication architecture is an important aspect the design will have to take into account. In this respect, broadcast communication seems well adapted to support communication in a dynamically changing sets of threads.

Once the model is in place, we will consider how the work on the extraction of resource bounds and on the control of information flow can be adapted to it. The stress here will be on modularity and compositional reasoning so that static analyses for sequential threads can be reused possibly modulo some global coherence condition.

A fourth and final objective of the project is to experiment with the notion of *proof-carrying code* in the context of the security properties and the programming model studied in the project. As a first step, we plan to formalize a virtual machine and the related security properties in a proof assistant such as COQ. As a second step, we plan to experiment with the automatic derivation of proofs (certificates) for the object code of the virtual machine starting from the static analyses performed on the source code.

To summarize the project proposes to :

1. Advance the state of the art on the automatic inference of resource bounds for functional programs.
2. Design a model of synchronous sequential modules allowing for dynamic reconfiguration.
3. Define techniques to bound the resources and control the flow of information in the synchronous model.
4. Experiment with the notion of proof-carrying code in the framework of the model and the properties mentioned above.

ACI Séminaire Informatique - Projet GRIS
MOYENS FINANCIERS ET HUMAINS DEMANDÉS PAR CHAQUE ÉQUIPE

C1 - Demandes effectuées dans le cadre de l'ACI pour le présent projet :

Nom de l'équipe ou du laboratoire : LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE MARSEILLE, ÉQUIPE MODÉLISATION ET VÉRIFICATION.

Moyens demandés dans le cadre de la présente ACI (en KEuros TTC) :

Financements via le Fonds National de la Science :

	2003	2004	2005	Total
Équipement	2000	2000	2000	6000 Euros
Fonctionnement (dont CDD décrits ci-dessous)	16000	8000	8000	32000 Euros
Total / année	18000	10000	10000	38000 Euros

Dépenses de personnels (CDD)³ :

Nature de l'emploi (post-doc, ingénieur, assistant-ingénieur, ...)	assistant-ingénieur
Durée de l'emploi (en mois) ⁴	4 mois
Coût total de l'emploi	8000 Euros

Financements via les organismes de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre de post-docs (préciser pour chaque demande la durée en mois) ⁵				
Nombre d'accueils de chercheurs étrangers (préciser pour chaque demande la durée en mois)				
Nombre d'accueils en délégations ou détachements ⁶				

³Un tableau doit être rempli pour chaque demande de CDD.

⁴Doit être inférieure à 24 mois.

⁵Sauf demande argumentée, la durée d'un contrat de type post-doc ne pourra excéder 12 mois.

⁶Certaines des demandes déjà faites pour 2003-2004 pourront être attribuées au titre de l'ACI.

Allocations de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre d'allocations de recherche débutant en :	0	1	0	1

Justifications scientifiques de l'ensemble des demandes : Le LIF s'engage pour 1,1 homme/année (en considérant un engagement à 50% du responsable de site et un mi-temps pour les enseignants-chercheurs permanents).

- La demande d'un CDD la première année vise à financer un stage d'été d'un étudiant de DEA ou de DESS qui porte sur la génération de certificats (section 0.0.6).
- La demande d'une allocation en 2004 vise aussi le thème de la génération de certificats. Cette demande est moins prioritaire que celle de l'INRIA.
- Equipement et fonctionnement : voir récapitulatif global.

C2 - Autres soutiens financiers apportés au projet :

- Action Spécifique CNRS *Modèles formels pour la mobilité* (site coordinateur).
- Action Spécifique CNRS *Sécurité Logicielle* (site participant).
- Projet INRIA MIMOSA (commun à l'INRIA-Sophia, l'Université de Provence et l'Ecole des Mines de Paris).

Moyens demandés dans le cadre de la présente ACI (en KEuros TTC) :

Financements via le Fonds National de la Science :

	2003	2004	2005	Total
Équipement	2000	2000	2000	6000
Fonctionnement (dont CDD décrits ci-dessous)	8000	8000	8000	24000
Total / année	10000	10000	10000	30000

Dépenses de personnels (CDD)⁷ :

Nature de l'emploi (post-doc, ingénieur, assistant-ingénieur, ...)	
Durée de l'emploi (en mois) ⁸	
Coût total de l'emploi	

Financements via les organismes de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre de post-docs (préciser pour chaque demande la durée en mois) ⁹				
Nombre d'accueils de chercheurs étrangers (préciser pour chaque demande la durée en mois)				
Nombre d'accueils en délégations ou détachements ¹⁰				

⁷ Un tableau doit être rempli pour chaque demande de CDD.

⁸ Doit être inférieure à 24 mois.

⁹ Sauf demande argumentée, la durée d'un contrat de type post-doc ne pourra excéder 12 mois.

¹⁰ Certaines des demandes déjà faites pour 2003-2004 pourront être attribuées au titre de l'ACI.

Allocations de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre d'allocations de recherche débutant en :	1	0	0	1

Justifications scientifiques de l'ensemble des demandes : L'INRIA Sophia s'engage pour 0,9 homme/année (en considérant un engagement à 50% du responsable de site et un mi-temps pour les enseignants-chercheurs permanents).

- La demande d'allocation vise le thème Langage de coordination et contrôle de ressources (voir sections 0.0.3 et 0.0.5). On prévoit un co-encadrement de la thèse par l'INRIA-Sophia et le LIF. Cette demande d'allocation est prioritaire sur celles du LORIA et du LIF.
- Equipement et fonctionnement : voir récapitulatif global.

C2 - Autres soutiens financiers apportés au projet :

- Action Spécifique CNRS *Sécurité Logicielle* (site participant).

Non de l'équipe ou du laboratoire : BDRIA

Moyens demandés dans le cadre de la présente ACI (en KEuros TTC) :

Financements via le Fonds National de la Science :

	2003	2004	2005	Total
Équipement	2000	2000	2000	6000
Fonctionnement (dont CDD décrits ci-dessous)	8000	8000	8000	8000
Total / année	10000	10000	10000	30000

Dépenses de personnels (CDD)¹¹ :

Nature de l'emploi (post-doc, ingénieur, assistant-ingénieur, ...)	
Durée de l'emploi (en mois) ¹²	
Coût total de l'emploi	

Financements via les organismes de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre de post-docs (préciser pour chaque demande la durée en mois) ¹³				
Nombre d'accueils de chercheurs étrangers (préciser pour chaque demande la durée en mois)				
Nombre d'accueils en délégations ou détachements ¹⁴				

¹¹ Un tableau doit être rempli pour chaque demande de CDD.

¹² Doit être inférieure à 24 mois.

¹³ Sauf demande argumentée, la durée d'un contrat de type post-doc ne pourra excéder 12 mois.

¹⁴ Certaines des demandes déjà faites pour 2003-2004 pourront être attribuées au titre de l'ACI.

Allocations de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre d'allocations de recherche débutant en :	1	0	0	0

Justifications scientifiques de l'ensemble des demandes : Le LORIA s'engage pour 0,8 homme/année (en considérant un engagement à 50% du responsable de site et un mi-temps pour les enseignants-chercheurs permanents).

- La demande d'allocation vise les thème Inférence automatique de bornes de complexité pour programmes fonctionnels (voir section 0.0.1). Cette demande d'allocation est moins prioritaire que celle de l'INRIA Sophia.
- Equipement et fonctionnement : voir récapitulatif global.

C2 - Autres soutiens financiers apportés au projet :

- Projet RNTL Averroès.

Moyens demandés dans le cadre de la présente ACI (en KEuros TTC) :

Financements via le Fonds National de la Science :

	2003	2004	2005	Total
Équipement	2000	2000	2000	6000
Fonctionnement (dont CDD décrits ci-dessous)	7000	7000	7000	21000
Total / année	9000	9000	9000	27000

Dépenses de personnels (CDD)¹⁵ :

Nature de l'emploi (post-doc, ingénieur, assistant-ingénieur, ...)	0
Durée de l'emploi (en mois) ¹⁶	0
Coût total de l'emploi	0

Financements via les organismes de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre de post-docs				
Nombre d'accueils de chercheurs étrangers	1 (2 mois)	1 (2 mois)	1 (2 mois)	3 (6 mois)
Nombre d'accueils en délégations ou détachements ¹⁷	0	0	0	0

¹⁵ Un tableau doit être rempli pour chaque demande de CDD.

¹⁶ Doit être inférieure à 24 mois.

¹⁷ Certaines des demandes déjà faites pour 2003-2004 pourront être attribuées au titre de l'ACI.

Allocations de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre d'allocations de recherche débutant en :	0	0	0	0

Justifications scientifiques de l'ensemble des demandes : Le LIPN s'engage pour 0,65 homme/année (en considérant un engagement à 50% du responsable de site et un mi-temps pour les enseignants-chercheurs permanents).

- La demande d'accueil de chercheurs étrangers vise le thème Complexité, ordres supérieur et logique linéaire (voir section 0.0.2). Parmi les candidats possibles mentionnons Andrzej MURAWSKI de l'Université de Oxford et Kazushige TERUI du *National Institute of Informatics*, Tokyo.
- Equipement et fonctionnement : voir récapitulatif global.

C2 - Autres soutiens financiers apportés au projet :

- Action Spécifique CNRS *Modèles formels pour la mobilité* (site participant).

- Récapitulatif de l'engagement en homme/an des quatre sites participants (en considérant un engagement à 50% du responsable de site et un mi-temps pour les enseignants-chercheurs permanents) :

LIF	INRIA Sophia	LORIA	LIPN	Total
1,1	0,9	0,8	0,65	3,45

- La demande d'allocation de l'INRIA Sophia est prioritaire sur celles du LIF et du LORIA qui ont la même priorité.
- Les demandes en équipement couvrent l'achat de deux stations de travail par site.
- Les demandes en fonctionnement couvrent :
 - L'organisation d'une réunion par an.
 - L'organisation de séminaires de chercheurs dont l'activité est pertinente pour le projet.
 - Les déplacements de participants entre sites.
 - La prise en charge des missions des stagiaires et des doctorants impliqués dans le projet.
 - La présentation des résultats du projet dans des conférences internationales.
 - La participation à des écoles sur les thèmes du projet.

Financements via le Fonds National de la Science :

	2003	2004	2005	Total
Équipement	8000	8000	8000	24000
Fonctionnement (dont CDD décrits ci-dessous)	39000	31000	31000	101000
Total / année	47000	39000	39000	125000 Euros

Dépenses de personnels (CDD) :

Nature de l'emploi (post-doc, ingénieur, assistant-ingénieur, ..)	assistant-ingénieur
Durée de l'emploi (en mois)	4 mois
Coût total de l'emploi	8000 Euros

Financements via les organismes de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre de post-docs	0	0	0	0
Nombre d'accueils de chercheurs étrangers	1 (2 mois)	1 (2 mois)	1 (2 mois)	3 (6 mois)
Nombre d'accueils en délégations ou détachements	0	0	0	0

Allocations de recherche :

	2003-2004	2004-2005	2005-2006	Total
Nombre d'allocations de recherche débutant en :	2	1	0	3

¹⁷Sauf demande argumentée, la durée d'un contrat de type post-doc ne pourra excéder 12 mois.

ENGAGEMENT DU COORDINATEUR DU PROJET :

La présente page ne sera remplie que dans la version sous forme papier.

Je soussigné, Roberto AMADIO, coordinateur du projet CRISS, m'engage dans l'hypothèse où le présent projet serait retenu à :

- fournir un rapport d'évaluation à mi-parcours permettant au Conseil Scientifique d'apprécier l'avancement des travaux et la coopération des équipes participantes,
- un rapport à la fin de l'exécution du projet,
- maintenir régulièrement une page web résumant l'ensemble des activités du projet.

Signature du coordinateur du projet :

Visa du Directeur du Laboratoire ou de l'Unité de Recherche auquel appartient le coordinateur du projet :