

CRISS:
Contrôle de Ressources et d'Interférence dans les
Systèmes Synchrones

<http://www.pps.jussieu.fr/~amadio/Criss/criss.html>

Projet ACI Sécurité Informatique 2003-2006

Rapport final

Nr Aide 03262. Réf. CNRS 500176.

Roberto M. Amadio
Université Paris 7, amadio@pps.jussieu.fr

25 septembre 2006

Résumé

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.

1 Participants

Site	Responsable
INRIA-Sophia	G. BOUDOL
LIF, Marseille	S. COUPET-GRIMAL
LIPN, Villetaneuse	P. BAILLOT
LORIA, Nancy	J.-Y. MARION
PPS, Paris	R. AMADIO

1.1 Participants (détail)

INRIA G. Boudol, I. Castellani, F. Dabrowski (en thèse depuis 2003, support ACI Sécurité Informatique), S. Epardaud (en thèse depuis 2004, support MENRT), A. Matos (en thèse de 2002 à 2005, support Egide).

LIF S. Coupet-Grimal, S. Dal Zilio, W. Delobel (en thèse depuis 2004, support MENRT).

LIPN P. Baillot, D. De Carvalho (en thèse depuis 2003, support ACI Nouvelles Interf. Math.), V. Mogbil.

LORIA G. Bonfante, J.-Y. Marion, J.-Y. Moyen, R. Péchoux (en thèse depuis 2004, support MENRT).

PPS R. Amadio (au LIF de 2003 à 2004).

2 Articulation du projet

Le projet s'articule en 6 parties :

1. Contrôle de ressources pour programmes fonctionnels du premier ordre.
2. Contrôle de ressources à l'ordre supérieur et logique linéaire.
3. Modèle de threads synchrones.
4. Non-interférence.
5. Contrôle de ressources pour systèmes synchrones.
6. Génération de certificats et preuve formelle.

Dans la suite nous décrivons pour chaque partie :

1. la problématique,
2. les résultats principaux.
3. le logiciel expérimental développé et
4. la liste des publications.

La page du projet¹ contient les pointeurs aux publications et aux logiciels développés ainsi que le programme des réunions organisées dans le cadre du projet.

Pour les motivations et le contexte scientifique du projet nous référons le lecteur au descriptif complet du projet soumis en 2003 disponible à l'adresse

[http : //www.pps.jussieu.fr/ ~ amadio/Criss/criss03.pdf](http://www.pps.jussieu.fr/~amadio/Criss/criss03.pdf)

2.1 Contrôle de ressources pour programmes fonctionnels du premier ordre

2.1.1 Problématique

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. 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. Nous nous intéressons au problème de la synthèse automatique de quasi-interprétations pour des programmes avec récursion générale.

2.1.2 Résultats principaux

1. Nous avons constaté qu'en pratique des 'petits' polynômes suffisent. Nous nous sommes donc intéressés aux polynômes à plusieurs variables sur l'algèbre max-plus. Nous avons montré que le problème de la synthèse est NP-dur et dans NP dans le cas particulier des polynômes multi-linéaires. Un certain nombre d'heuristiques ont été proposées et mise-en-oeuvre pour la synthèse de ces polynômes (voir notamment le système CROCUS).
2. Nous nous sommes intéressés à une généralisation de la notion de quasi-interprétation qu'on appelle *sup-interprétation* qui permet de traiter un certain nombres de fonctions comme le pgcd et le logarithme qui échappent à une analyse par quasi-interprétation.
3. Nous avons commencé à utiliser la notion de quasi-interprétation pour caractériser le calcul en temps ou en espace logarithmique.

2.1.3 Logiciel expérimental

G. Bonfante. CROCUS un synthétiseur de quasi-interprétations. Septembre 2006.

2.1.4 Publications

1. J.Y. Moyen Analyse de la complexité et transformation de programmes. Thèse de Doctorat, Nancy, 2004.

¹[http : //www.pps.jussieu.fr/~amadio/Criss/criss.html](http://www.pps.jussieu.fr/~amadio/Criss/criss.html)

2. G. Bonfante, J.-Y. Marion et Jean-Yves Moyen. Quasi-interpretations and small space bounds. In Proc. Rewriting techniques and applications (RTA 2005), SLNCS, 2005.
3. P. Baillot, U. Dal Lago et J.-Y. Moyen. On quasi-interpretations, blind abstractions and implicit complexity. Preprint HAL ccsd-00023668. Presented at 8th International Workshop on Logic and Computational Complexity (LCC'06) (Satellite de FLOC-LICS), 2006.
4. G. Bonfante, J.-Y. Marion et R. Péchoux. A characterisation of alternating log time by first-order functional programs. In Proc. LPAR, SLNCS, 2006.
5. J.-Y. Marion et R. Péchoux. Resource analysis by sup-interpretation. In Proc. FLOPS 2006, SLNCS 3945, 2006.
6. G. Bonfante. Some programming languages for LOGSPACE and PTIME. In Proc. AMAST 06, SLNCS, 2006.
7. J.-Y. Marion et R. Péchoux. Quasi-friendly sup-interpretations. Presented at 8th International Workshop on Logic and Computational Complexity (LCC'06) (Satellite of FLOC-LICS), 2006.

2.2 Contrôle de ressources à l'ordre supérieur et logique linéaire

2.2.1 Problématique

Les techniques basées sur les quasi-interprétations sont limitées aux langage fonctionnels de premier ordre. Notre approche au contrôle de ressources à l'ordre supérieur repose sur des fragments de la la *logique linéaire*.

2.2.2 Résultats principaux

Nous avons introduit un fragment de logique linéaire (DLAL pour *dual light affine logique*) qui permet de surmonter deux limitations majeures du fragment LAL (pour *light affine logic*) et ceci sans perte d'expressivité en pratique.

1. DLAL assure une borne polynomiale en temps par rapport à une variété de stratégies de réduction du λ -calcul alors que LAL nécessite une traduction dans les réseaux de preuve.
2. DLAL a un algorithme d'inférence de type efficace y compris pour un système de type polymorphe (qui a été mis en oeuvre) alors qu'à ce jour le meilleur algorithme d'inférence de type pour LAL couvre seulement les types simples et sa complexité demeure inconnue.

2.2.3 Logiciel expérimental

V. Atassi. A type inference program in OCAML for elementary linear logic. November 2005.

2.2.4 Publications

1. P. Baillot, V. Mogbil. Soft lambda-calculus : a language for polynomial time computation. In Proc. FOSSACS 2004, SLNCS, 2004.
2. P. Baillot, K. Terui. Light types for polynomial time computation in lambda-calculus. In Proc. LICS 2004, 2004.
3. P. Baillot, K. Terui. A feasible algorithm for typing in Elementary Affine Logic. In Proc. International Conference on Typed Lambda Calculi and Applications (TLCA'05), SLNCS, 2005.
4. V. Atassi, P. Baillot, K. Terui. Verification of Ptime reducibility for system F terms via Dual Light Affine Logic. In Proc. Computer Science Logic 06, SLNCS, 2006.

2.3 Modèle de threads synchrones

2.3.1 Problématique

Nous adoptons le style de la programmation “réactive” (que nous appelons néanmoins *synchrone*) de Boussinot et De Simone, 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. Nous souhaitons dépasser le cadre des systèmes à états finis tels qu’on les programme en ESTEREL ou en LUSTRE par exemple, car nous voulons analyser 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.

2.3.2 Résultats principaux

1. Nous avons proposé un modèle de la mobilité (ULM) qui suit le paradigme d’un calcul globalement *asynchrone* et localement *synchrone* (au sens mentionné ci-dessus). Dans ce modèle les agents mobiles peuvent se déplacer avec leur état qui consiste d’une pile et d’une mémoire. Comme plusieurs agents peuvent partager les mêmes références, l’accès à une référence peut entraîner une suspension (au même sens que la lecture d’un signal absent). Nous introduisons une analyse statique formulée comme un système de types et effets qui permet de conclure que l’utilisation de certaines références est ‘sûre’, c’est-à-dire leur utilisation ne nécessite pas un test de présence préalable.
2. Nous avons revisité le modèle de programmation synchrone SL qui est à la base de la programmation ‘réactive’. Nous en avons proposé une formalisation plus générale qui comprend la génération de threads et les définitions récursives. Par le biais d’une traduction CPS, nous avons montré que ce nouveau modèle se réduit à une forme récursive terminale qui repose sur un seul opérateur de synchronisation (le **present**). Par ailleurs, nous avons introduit une sémantique compositionnelle du modèle basée sur la notion de bisimulation. Ensuite, le modèle a été étendu avec des types de données du premier ordre. Le résultat de cet extension est une sorte de π -calcul

synchrone. Nous avons montré que notre approche à la sémantique de SL s'applique aussi à ce modèle étendu.

2.3.3 Logiciel expérimental

S. Epardaud. The embedding of the ULM model in a Scheme language, and a Compiler and Virtual Machine for this embedding. January 2005.

2.3.4 Publications

1. G. Boudol ULM, a core programming model for global computing. In Proc of ESOP04, LNCS, 2004.
2. S. Epardaud. Mobile Reactive Programming in ULM. Scheme Workshop 2004, In Proc. Fifth ACM SIGPLAN Workshop on Scheme and Functional Programming, 2004.
3. R. Amadio, G. Boudol, F. Boussinot, I. Castellani. Reactive concurrent programming revisited. Extended abstract presented at the workshop Algebraic Process Calculi : the first twenty five years and beyond, Bertinoro, August 2005. To appear in Electronic Notes in TCS.
4. R. Amadio. The SL synchronous language, revisited. Technical Report, Université Paris 7, Laboratoire PPS, November 2005. To appear in Journal of Logic and Algebraic Programming.
5. R. Amadio. A synchronous pi-calculus Technical Report, Université Paris 7, Laboratoire PPS, June 2006.
6. F. Dabrowski et F. Boussinot, Cooperative threads and preemptive computations. Presented at Workshop on Thread Verification, 2006 (Satellite of FLOC-LICS), 2006.

2.4 Non-interférence

2.4.1 Problématique

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. En suivant les travaux de Volpano et Smith on peut formaliser les techniques d'analyse sont formalisées comme un système de typage. L'objectif est d'étendre cet approche à un modèle comme ULM (cf. section 2.3).

2.4.2 Résultats principaux

- Nous avons proposé des systèmes de typage et une méthodologie de preuve qui permettent d'assurer la propriété de non-interférence pour une variété de constructions

de programmation qui comprennent l'ordre supérieur, les références, les threads avec exécution synchrone ou asynchrone et la mobilité du code.

- Par ailleurs nous avons proposé un mécanisme de déclassification qui permet au programmeur de postuler l'admissibilité de certains flots d'information et au système de type de vérifier la cohérence d'utilisation de ces postulats.

2.4.3 Publications

1. G. Boudol, I. Castellani, A. Matos. Typing non-interference for reactive programs. February 2004. Extended abstract in Foundations of Computer Security 2004 Workshop. Full version in Journal of Logic and Algebraic Programming, to appear.
2. G. Boudol, A. Matos. On declassification and the non-disclosure policy. In Proc IEEE Computer Security Foundations Workshop 2005.
3. G. Boudol. On typing information flow. In Proc. International Colloquium on Theoretical Aspects of Computing. SLNCS, 2005.
4. A. Almeida Matos. Typing secure information flow : declassification and mobility. Thèse. École des Mines (Sophia-Antipolis). January 2006.

2.5 Contrôle de ressources pour systèmes synchrones

2.5.1 Problématique

La *réactivité* est une propriété essentielle d'un programme synchrone. De façon informelle, elle garantit qu'à chaque instant le programme recevant une entrée 'réagit' en produisant une sortie. Si un programme manipule des valeurs de taille variables comme listes, arbres, graphes, . . . alors l'analyse peut aller au delà de la réactivité. En particulier notre objectif est de développer des techniques d'*analyse statique* qui garantissent une forme de *réactivité efficace* (en temps polynomial).

2.5.2 Résultats principaux

Nous avons proposé une méthodologie pour annoter les programmes et des techniques d'analyse statique qui permettent de garantir la réactivité efficace de programmes exprimés dans le π -calcul synchrone (cf. section 2.3). Parmi les annotations, mentionnons celles des threads qui visent à expliciter un comportement cyclique et celles des signaux qui visent à mettre en évidence une stratification dans le flot de données. A partir de ces annotations nous générons un système d'inégalités entre termes du premier ordre. Si ce système d'inégalités est satisfait par une quasi-interprétation polynomiale (cf. section 2.1) alors le programme associé réagit à chaque instant en temps polynomial.

2.5.3 Publications

1. R.M. Amadio, S. Dal Zilio. Resource Control for Synchronous Cooperative In Proc. CONCUR 2004, SLNCS, 2004.

2. R.M. Amadio, S. Dal Zilio. Resource Control for Synchronous Cooperative Threads. *Theoretical Computer Science*, 358 :229-254, 2006.
3. R. Amadio, F. Dabrowski. Feasible reactivity for synchronous cooperative threads. In *Proc. Expressiveness in Concurrency Workshop*, September 2005. *Electronic Notes in TCS*, volume 154, issue 3, 2006.
4. F. Dabrowski. PhD Thesis. Forthcoming.

2.6 Génération de certificats et preuve formelle

2.6.1 Problématique

Dans les dernières années la notion de code certifié (*proof-carrying code*) s'est révélée comme une méthode particulièrement intéressante pour assurer la sécurité du code mobile. Notre objectif est de mener un travail d'expérimentation sur un modèle de machine virtuelle extrêmement simplifié en nous concentrant sur le contrôle de ressources. Le but ici est d'avoir une plate-forme d'expérimentation dont l'intérêt est plutôt pédagogique que commercial.

2.6.2 Résultats principaux

1. Nous avons formalisé une machine virtuelle et le code octet associé d'un langage fonctionnel du premier et d'un noyau de langage de threads synchrones. Nous avons montré comment reformuler les conditions pour le contrôle de ressources considérées dans les sections 2.1 et 2.5 au niveau du code octet. Cette reformulation est basée sur une *analyse de forme* qui permet de reconstruire statiquement la 'forme' des valeurs présentes sur la pile au moment de l'exécution. La machine virtuelle et le code octet ont été formalisés dans le système de preuve Coq et la correction de l'analyse de forme a été validée par le système.
2. Les techniques pour le contrôle de ressources discutées dans la section 2.1 font souvent appel à des ordres récurrents sur les chemins (RPO). Traditionnellement, la bonne fondation de ces ordres repose sur le théorème de Kruskal. La preuve de ce théorème étant non-constructive, sa formalisation en Coq s'avère problématique. Nous nous sommes intéressés à une méthode de preuve directe et constructive de la bonne fondation de RPO et nous sommes arrivés à mener à bien cette preuve dans le système Coq.

2.6.3 Logiciel expérimental

1. LIF Marseille team. Implementation of shape analysis for the byte code of a first-order functional language. January 2005.
2. S. Coupet-Grimal, W. Delobel. A generic development of Kildall algorithm in Coq and its specialisation to type and shape analysis of functional bytecode. June 2005.

3. S. Coupet-Grimal, W. Delobel. Two contributions concerning MPO and RPO to the Coq library on rewriting and termination (CoLoR).

2.6.4 Publications

1. R.M. Amadio, S. Coupet-Grimal, S. Dal Zilio, L. Jakubiec. A functional scenario for bytecode verification of resource bounds. In Proc. CSL 2004, SLNCS, 2004.
2. S. Coupet-Grimal, W. Delobel. A uniform and certified approach for two static analyses. In Proc. TYPES 2004, Filliatre et al (eds.), SLNCS, 2004.
3. R.M. Amadio, S. Dal Zilio. Resource Control for Synchronous Cooperative Threads. *Theoretical Computer Science*, 358 :229-254, 2006.
4. S. Dal Zilio, R. Gascon. Resource bound certification for a tail recursive machine. Extended abstract in Proc. Third Asian Symposium on Programming Languages and Systems (APLAS 2005), SLNCS, 2005.
5. S. Coupet-Grimal, W. Delobel. An Effective Proof of the Well-Foundedness of the Multiset Path Ordering. To appear in AAECC Journal (Applicable Algebra in Engineering Communication and Computing).
6. S. Coupet-Grimal, W. Delobel. Une preuve effective de la bonne fondation de l'ordre récursif multi-ensemble sur les chemins. Dans Journées Francophones des Langages Applicatifs, JFLA 2006. (Version en français de l'article ci-dessus).
7. S. Coupet-Grimal, W. Delobel. A Constructive Axiomatization of the Recursive Path Ordering. RR LIF 28-2006.

3 Diffusion

- R. Amadio a été *General chair* du *18th Computer Security Foundations Workshop*, Aix-en-Provence, 20-22 Juin 2005.
- R. Amadio a participé à l'organisation de la Spring School on Security au CIRM Marseille en 2005. A cette occasion G. Boudol a proposé un cours sur la non-interférence,
- R. Amadio, P. Baillot et J.-Y. Marion ont participé à l'organisation de GEOCAL 2006 (CIRM Marseille). Dans ce contexte P. Baillot et J.-Y. Marion ont proposé un cours sur Complexité et Logique.