(Mars 2015)

Curriculum Vitae (version longue) de

Frédéric Gava

Table des matières

1 Notice

Frédéric Gava
Maître de Conférences, 27ème section ; Numéro de qualification : 166182
Né le 10 octobre 1979 à Mont-Saint-Aignan (34 ans) ; Nationalité : française
Situation familiale : 2 enfants ; Service national : exempté

1.1 Adresses

Adresse personnelle :
11 rue Jacques Boutrolle, Résidence d'Estaimbuc, Appt 201
76130 Mont-Saint-Aignan
Tél : 02 35 88 63 50
Adresse professionnelle :
Laboratoire d'Algorithmique, Complexité et Logique (LACL)
Université de Paris-Est Créteil Val-de-Marne (UPEC)
61, avenue du Général De Gaulle, bâtiment P2 du CMC
94010 Créteil cedex
Fax : 01 45 17 66 01
Contact :
Tél : 06 44 10 19 26
Courriel : `frederic.gava@univ-paris-est.fr`
Page web : `http://www.lacl.fr/gava/`

1.2 Fonctions

Fonctions précédentes :

De septembre 2002 à 2005, j'étais allocataire de recherche en Informatique à l'Université de Paris XII–Val-de-Marne, affilié à l'équipe « Systèmes Communicants » du Laboratoire d'Algorithmique, Complexité et Logique (LACL), thèse préparée sous la direction de Frédéric Loulergue. J'étais également moniteur des universités à partir de septembre 2003 à l'IUP MIAGE de l'Université de Paris XII. J'étais ATER (Attaché Temporaire d'Enseignement et de Recherche) à temps plein à l'Université de Paris XII, de septembre 2005 à 2006, avec un demi-service en faculté des Sciences et un autre demi-service en faculté de Droit.

Maître de Conférences à temps plein à l'Université de Paris XII (nouvellement appelée UPEC), LACL, de septembre 2006 (titulaire depuis 2007) à septembre 2011, avec un service d'enseignement en faculté de Droit.

Fonction actuelle :

Maître de Conférences à temps plein à l'Université de Paris Est Créteil (UPEC), LACL, avec un service d'enseignement à la faculté des Sciences Économique et de Gestion, ESIAG (École Supérieure d'Informatique Appliquée à la Gestion)

Bénéficiaire de la PEDR de septembre 2008 à septembre 2012.
Bénéficiaire de la PES (B) de septembre 2013 à septembre 2017.

1.3 Diplômes

Année	Diplôme	Mention	Lieu
2012	HDR		Université de Paris-Est
2002-2005	Doctorat d'informatique	THF	Université de Paris XII
2001-2002	DEA Programmation	Bien	Université de Paris VII
2000-2001	Maîtrise d'informatique	Bien	Université de Rouen
1999-2000	Licence d'informatique	Bien	Université de Rouen
1997-1999	DEUG MIAS		Université de Rouen
1997	Baccalauréat Général, série Scientifique		Lycée Camille Saint-Saëns de Rouen.

2001-2002 DEA Programmation au Laboratoire Preuves, Programmes et Systèmes (PPS), Université de Paris VII.

Stage : Un système de type polymorphe pour le langage BSML avec traits impératifs, sous la direction de Frédéric Loulergue.

Mention : Bien

Résumé court : Le langage BSML étend la programmation ML par des opérations parallèles BSP, un modèle structuré de parallélisme qui propose un modèle de coût à la fois fiable et portable. Le polymorphisme paramétrique simple de ML n'est pas suffisant pour la sûreté d'exécution de BSML (problème d'imbrication du parallélisme). Nous avons conçu une analyse statique garantissant cette sûreté.

2002-2005 Doctorat au LACL, Université de Paris XII.

Titre : Approches fonctionnelles de la programmation parallèle et des méta-ordinateurs ; Sémantiques, implantations et certification.

Jury :

Président :	Thierry PRIOL	IRISA-INRIA Rennes
Rapporteurs :	Murray COLE	Univ. de Edinburgh
	Walter DOSCH	Univ. de Lübeck
Examinateurs :	Jocelyn SÉROT	Univ. de Clermont-Ferrand
	Anatol SLISSENKO	Univ. de Paris XII – Val-de-Marne
Directeur :	Frédéric LOULERGUE	Univ. d'Orléans

Mention : Très honorable avec félicitations du jury

Prix de thèse : **Prix de thèse de la fondation d'entreprise EADS dans le domaine des STIC**

Résumé court : Pour que la programmation parallèle devienne aussi répandue que la séquentielle, elle se doit de posséder tous les mécanismes d'abstraction standards. Mais cette abstraction ne doit pas être obtenue au prix de coûts d'exécution prohibitifs ou imprévisibles. Les recherches de cette thèse (dans le cadre du projet CARAML) mêlent à la fois la conception de sémantiques opérationnelles pour des extensions parallèles explicites et structurées (basées initialement sur le modèle BSP) du langage fonctionnel ML, la preuve des propriétés de ces sémantiques, la compilation vers une machine abstraite, la certification des programmes à l'aide du système Coq, la multi-programmation de programmes fonctionnels parallèles, la conception de nouveaux modèles de programmation structurés (extentions pour les mémoires externes parallèles et le *meta-computing*) et le développement de bibliothèques pour le langage OCaml.

2012 Habilitation à Diriger des Recherches au LACL, Université de Paris-Est.

Titre : BSP, bon à toutes les sauces ; Application to Functional Programming, Mechanised Verification and Security Protocols.

Jury :

Président :	Kevin HAMMOND	Univ. de St. Andrews
Rapporteurs :	Jean-Christophe FILLIÂTRE	Univ. de Paris-Sud/CNRS
	Joaquim GABARRÓ	Univ. de Catalunya
	Jaco VAN DE POL	Univ. de Twente
Examinateurs :	Herbert KUCHEN	Univ. de Münster
	Marco DANELUTTO	Univ. de Pisa
	Zhenjiang HU	Univ. de Sokendai/Tokyo/NII
	Frédéric LOULERGUE	Univ. d'Orléans
Directeur :	Gaétan HAINS	Univ. de Paris-Est

Résumé court : Dans ce travail, nous nous intéressons à un modèle de calcul parallèle, appelé BSP, qui permet de structurer l'exécution des programmes parallèles. Le manuscrit est organisé en trois parties, chacune correspondant à une thèse encadrée par l'auteur. Dans un premier temps, nous étudierons un langage fonctionnel pour la programmation BSP, appelé BSML, notamment ses primitives, son implantation, le typage des programmes BSML, une méthodologie pour la preuve de programmes dans l'assistant de preuves COQ, ainsi que diverses extensions, comme la gestion d'exceptions distribuées. Plusieurs applications sont données, notamment l'implantation de divers patrons algorithmiques. Dans un second temps, nous décrivons le développement d'un outil pour la preuve déductive d'algorithmes BSP, c'est-à-dire annoter le code par des assertions logiques, puis générer des obligations de preuve qui valident ou non les assertions. Cet outil, appelé BSP-WHY fonctionne par une transformation du code parallèle BSP en code WHY (un outil déjà existant de vérification déductive) purement séquentiel. Des sémantiques opérationnelles effectuées à l'aide de COQ sont aussi décrites. Dans un dernier temps, nous exploitons la nature bien structurée de protocoles de sécurité et nous la faisons correspondre au modèle BSP afin d'obtenir des algorithmes parallèles efficaces de model-checking des protocoles de sécurité. Nous considérons le problème de la vérification de formules logiques (temporelles) LTL et CTL* sur des protocoles de de sécurité. Les algorithmes BSP de calculs de l'espace des états des protocoles seront aussi vérifiés formellement avec l'outil BSP-WHY.

1.4 Formations doctorales et expériences professionnelles (hors université)

- Formations liées à l'enseignement supérieur suivies avec le CIES Jussieu en tant que moniteur ;
- Formations suivies avec l'École Doctorale SIMME (Sciences et Ingénierie : Matériaux, Modélisation et Environnement) ;
- École Jeunes Chercheurs en Programmation (EJCP 2004), organisée par l'IRISA-Rennes ;
- École d'hiver GRID 2002, organisée par le CNRS ;
- Divers : Stagiaire au Service Développement Informatique de la société Lubrizol (juillet-août 2001) ; Animateur dans divers Centres de Loisirs et Centres de Vacances (17 semaines, 1999-2001) ; Surveillant de cantine à l'école Camus à Mont-Saint-Aignan (2001) ; B.A.F.A. (Brevet d'Aptitude aux Fonctions d'Animateurs) et B.S.B. (Brevet de Surveillant de Baignade) ; Permis B.

2 Charges collectives

- **Membre du conseil du laboratoire** du LACL (Directrice + 2 Professeurs et 2 Maîtres de Conférences) dont la fonction principale est le classement des demandes de financement (2008-12) ;
- 2 fois membre de la commission de recrutement pour des postes de Maître de Conférences au LACL (2008 et 2011) ;
- Membre de la commission de recrutement (2014) pour le LIPN (Paris Nord) ;
- **Co-responsable du groupe de recherche (GDR) LaMHA** (Langages et Modèles de Haut-niveau pour la programmation parallèle, distribuée, de grilles de calcul et Applications) affilié au GDR GPL (Génie de la Programmation et du Logiciel) depuis 2009 ; Organisation de journées annuelles du groupe. Organisation de la session annuelle du groupe au journées nationales du GDR-GPL ;
- Participation à l'organisation de la journée nationale du GDR-GPL 2014 (au CNAM) ;
- De 2007 à 2011 j'étais **responsable du séminaire hebdomadaire du LACL** (co-responsable de 2006 à 2007) ;
- Administration système : j'aide à la gestion et à l'administration de la grappe de PC (20 nœuds) du LACL ;
- Animations :
  - Tuteur d'accueil lors des journées portes ouvertes de 2005 et de 2006 de l'Université de Paris XII ;
  - Orateur lors des journées de la Science de 2006 à UPEC ; présentation de l'histoire de l'Informatique au grand public ;
  - Participation à l'organisation du workshop ASM 2005 (Abstract State Machines), qui a eu lieu à l'Université Paris XII ;
  - Orateur lors des journées de l'ESIAG 2012, présentation d'une partie de mes thèmes de recherche.
- Responsable de l'enseignement de l'informatique à la faculté de Droit (gestion des vacataires) jusqu'en 2011 ;
- Responsable de plusieurs cours (Java, Sécurité, *etc.*) à l'ESIAG ;
- **Responsable du L1 à l'ESIAG** depuis 2013 ;
- **Rédaction de la partie "Adossement à la recherche"** pour le dossier d'école d'ingénieur (ISIP) de l'ESIAG de l'UPEC auprès de la CTI (Commission des Titres d'Ingénieur) ; Réalisation de la maquette des cours (en informatique) des 3 premières années (avec participation à la rédaction des fiches de cours).

- **Classé second en 2012 pour le concours de Pr à l'Université de Picardie (MIS)**

3 Activités d'enseignement

Mes cours sont librement accessibles à `http://www.lacl.fr/gava/actuel.htm`.

3.1 Parcours d'enseignement

3.1.1 Faculté de Droit

De septembre 2005 à septembre 2006, j'ai été 1/2 ATER à la faculté de Droit puis Maître de Conférence de 2006 à 2011. La faculté venait de s'installer dans ses nouveaux locaux et une nouvelle maquette des cours venait d'être votée. J'ai donc mis en place ces nouveaux cours. Ils correspondaient essentiellement au C2i niveau 1 qu'un grand nombre d'étudiants a ensuite passé et réussi (le passage du C2i ne dépend pas de la faculté mais directement de l'université).

L'enseignement de l'Informatique se faisaient sur trois semestre. (1) Le premier, en licence, correspondait à la partie « théorique » du C2i. Il s'agissait d'introduire les notions essentielles de l'Informatique : architecture, réseaux, sécurité, système d'exploitation, logiciel libre, déontologie, *etc.* ; (2) Ensuite, au premier semestre de M1, nous abordions les formules Excel et les SGBD (base de données) c'est-à-dire la modélisation (modèle Entité-Association) puis le modèle relationnel et enfin les requêtes SQL (le tout généralement en Access). Cela permettait aux étudiants de s'initier à la programmation (syntaxe rigoureuse et logique) ainsi que de s'abstraire pour mettre en avant l'essentiel d'un problème donné ; (3) Enfin, des TD terminaient cet enseignement par de la bureautique standard (Word, publipostage) et la mise en application d'Excel et des formules SQL.

Du fait de ma section CNU, il n'était pas possible pour moi d'avoir d'autre responsabilité en faculté de Droit que celle des cours d'Informatique et de la gestion, chaque année, des nombreux vacataires pour les groupes de TD. Michael Guedj, m'a remplacé en tant qu'ATER à la faculté de Droit et maintenant, Julien Tesson, nouvellement nommé Maître de Conférences, a repris le poste d'Informatique à la faculté de Droit.

3.1.2 École Supérieure d'Informatique Appliquée à la Gestion (ESIAG)

Je suis maintenant rattaché à l'ESIAG (formation MIAGE de la faculté des Sciences Économiques et de Gestion). J'y étais précédemment moniteur (cours de SGBD ; initiation à l'algorithmique ; et langage, complexité et calculabilité). J'y avais aussi effectué quelques vacations lorsque j'étais MCF en faculté de Droit.

Nouvellement arrivé à l'ESIAG, où des collègues y sont en places depusi des annnées, je n'ai pu avoir de lourdes responsabilités. Néanmoins, j'ai participé à l'écriture d'un projet de « prépa-intégrée » (L1/L2) qui complète le désir de l'ESIAG de participer à la mise en place d'une école d'ingénieurs au sein de l'UPEC. Ce cursus viserait soit l'entrée dans cette hypothétique école d'ingénieur, soit dans l'actuel cursus de l'ESIAG en formation initiale (FI) ou en apprentissage (FA). Actuellement, le projet prévoit des cours assez classiques, comme initiation à l'algorithmique, à la programmation en Java/IHM, électronique, algèbre, analyse, statistique, anglais, gestion et Droit. Je suis aussi responsable, dans l'écriture du dossier auprès de la CPI, de la partie "Adossement à la recherche" qui consiste à montrer les liens entre notre école (et ses cours) et les laboratoires de l'UPEC. L'écriture est en cours avec un dossier pour juillet 2014 et une ouverture de l'école pour septembre 2015 au plus tôt.

Pour l'heure, je suis essentiellement **responsable de l'année de L1** (un groupe en FI) et en charge des cours de Java (« Java avancé-IHM », avec projet en L3 et « Calculs distribués en Java » en M1) et des cours de « Sécurité » en M2. Comme tous mes collègues, je participe au tutorat des étudiants qui consiste en des rendez-vous trimestriels et des visites en entreprise lors des stages.

3.1.3 M2 SSI (Sécurité des Systèmes Informatiques)

Depuis la création du M2 SSI en faculté des Sciences, je participe à l'enseignement. J'ai d'abord pris en charge le cours de « Programmation et Sûreté des Systèmes répartis » où j'introduisais le modèle BSP, la programmation en C+MPI et des sémantiques opérationnelles d'un micro-langage impératif parallèle. La sûreté était introduite par les barrières BSP contre les inter-blocages et les sémantiques pour l'analyse formelle de programmes. Je partageais initialement le cours avec Fabrice Mourlin (MCF à l'ESIAG) qui lui faisait un cours sur les bibliothèques de programmation multi-agents en Java. Cette partie a ensuite été effectuée par Olivier Michel (Pr. à l'ESIAG) mais cette fois-ci sur les approches non-conventionnelles des langages de programmation (autonomic et spacial computing).

J'ai ensuite laissé entièrement ce cours pour reprendre celui sur la sécurité des systèmes d'exploitations (SESE) où nous présentions et étudions (notamment comment s'en prévenir) divers attaques systèmes depuis des programmes par exemples attaques par buffer/integer-overflow, concurrence, injection SQL *etc.* Ce cours comporte des TP qui sont régulièrement assurés par un intervenant extérieur (membre d'une SSII spécialisé en sécurité).

Dans ce master, je participe aussi régulièrement à l'encadrement des stagiaires en entreprise et aux soutenances.

En tant que tuteur d'étudiants à l'ESIAG, je participe au jury de soutenance de stages en entreprise pour des étudiants en L3, M1 et M2. Je participe aussi aux soutenances de stages en entreprise pour des étudiants du M2 Pro Sécurité (SSI).

3.2 Synthèse et résumé des enseignements

Dans les tableaux des Figures 1 et 2, je donne les listes des enseignements que j'ai effectuées où les volumes horaires sont donnés en heures équivalents-TD. Soit environs : 160h en L1, 187h en L2, 600h en L3, 1435h en M1 et **315h en M2**. Notons qu'à l'ESIAG, les cours sont sous la forme cours/TD de 3h30 et donc comportent aussi beaucoup de TP notamment pour les cours de programmation.

Initiation aux technologies de l'information Ce cours a pour objectif d'étendre la culture informatique des étudiants juristes en leur présentant brièvement l'architecture d'un ordinateur, le système d'exploitation, les réseaux, Internet ainsi qu'en faisant régulièrement mention des problèmes de sûreté et de sécurité informatique. Les étudiants disposeront alors des notions théoriques pour passer le « Certificat informatique et internet » (C2i) de niveau 1.

Excel, SGBD, Sécurité Ce cours expose les premières notions indispensables pour l'utilisation d'un SGBD par l'étude du modèle entité-association, de l'analyse des données, du modèle relationnel et du langage SQL. L'accent est mis (puisque le cours est dédié aux étudiants juristes et non informaticiens) sur l'utilisation du SGBD Access de la suite Office. Des problèmes de sécurité dans les SGBD et Internet sont aussi présentés. Un rappel est fait sur le Droit en Informatique notamment dans le cas des SGBD. Nous présentons aussi les formules en Excel.

Internet et Bureautique Ces TP ont pour objectif d'assurer aux étudiants juristes, par une présentation systématique des notions mises en jeu, la maîtrise des logiciels d'Internet et de la Bureautique, notamment pour la mise en forme de documents structurés (notions de style et de modèle en Word), de feuilles de calculs (en Excel), publipostage (powerpoint), d'un petit SGBD (Access) et de présentations sur vidéo-projecteurs. Ces enseignements visent à les familiariser avec l'outil informatique en général. Les étudiants disposeront alors de suffisamment de pratique pour passer le « Certificat informatique et internet » (C2i) de niveau 1.

Introduction à l'algorithmique Cet enseignement est une introduction à la conception d'algorithmes et à l'analyse de leur efficacité (cours, cette année, similaires pour les 2 niveaux suite à la réforme LMD). Sont introduits les concepts de base de l'algorithmique, tels que la complexité et les structures de données classiques (listes et arbres). Les algorithmes sont implantés en ADA (introduit aussi lors de ce cours).

Algorithmique avancée Cet enseignement est une introduction aux algorithmes classiques sur les arbres (équilibrés, rouge-noir) et de leurs complexités. Des cas d'utilisation sont aussi donnés (en logique, notamment).

Année	Matière	Cursus	Bac+	Nature	Vol. (H-TD)
2009-10					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Excel, SGBD et Sécurité	Droit	4	Cours	22h30
	Internet et Bureautique	Droit	4	TP	180h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours et TP	15h
2008-09					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Excel, SGBD et Sécurité	Droit	4	Cours	22h30
	Internet et Bureautique	Droit	4	TP	195h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours et TP	15h
2007-08					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Excel, SGBD et Sécurité	Droit	4	Cours	22h30
	Internet et Bureautique	Droit	4	TP	225h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours et TP	12h
2006-07					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Excel, SGBD et Sécurité	Droit	4	Cours	22h30
	Internet et Bureautique	Droit	4	TP	165h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours et TP	13h30
	Sémantiques des langages de programmation	Sciences	4	TD	21h
	Programmation Orientée Objet (Init Java)	ESIAG	2	Cours et TD	48h
2005-06					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Internet et Bureautique	Droit	4	TP	90h
	Initiation à l'algorithmique	Sciences	2	TD et TP	35h
	Algorithmique avancé	Sciences	3	TD et TP	35h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours	15h
	Sémantiques des langages de programmation	Sciences	4	TD	19h30
2004-05					
	Initiation aux bases de données	ESIAG	2	Cours et TD	48h
	Architectures parallèles	Sciences	4	TD et TP	21h
2003-04					
	Langage, complexité et calculabilité	ESIAG	3	Cours et TD	54h
	Langages pour la programmation concurrente	Sciences	4	TD	10h30
2002-03					
	Introduction à la Programmation en ADA	ESIAG	2	Cours et TD	48h
	Langages pour la programmation concurrente	Sciences	4	TD	10h30
	Architectures parallèles	Sciences	4	TD et TP	21h
1999-01					
	Tutorat d'Informatique	DEUG MIAS	1	Tutorat	60h

FIGURE 1 – Partie 1 de la synthèse du service effectué ; années 1999 à 2010.

Programmation et sûreté des systèmes répartis Ce cours introduit la programmation C+MPI ainsi que BSP (modèle de coût et sa programmation en C+BSPlib ou BSML), les agents mobiles et les problèmes de sûreté et de sécurité des machines parallèles et Grid. Des rappels de sémantiques sont donnés. Sont aussi présentés les domaines de recherche du thème « programmation parallèle de haut niveau » de l'équipe « Systèmes Communiquants ».

Sémantiques des langages de programmation Il s'agit d'un cours de sémantique des langages en deux parties. La première partie est consacrée aux sémantiques opérationnelles (naturelles, SOS, à « petits pas » et systèmes de réécriture) et aux systèmes de types (types simples, pas de polymorphisme) d'un mini-ML. Des preuves d'équivalences entre les sémantiques et des preuves de préservation du typage sont présentées. La seconde partie présente les mêmes sémantiques mais pour un mini-ADA.

Initiation aux systèmes de gestion des bases de données Ce cours expose les premières notions indispensables pour l'utilisation des SGBD par l'étude du modèle entité-association, l'analyse des données, du modèle relationnel et du langage SQL.

Architectures et/ou Algorithmes parallèles Ce cours est une introduction au parallélisme : architectures des machines parallèles et des réseaux, modèles et algorithmes PRAM (théroème de Brent), techniques de parallélisation automatique (nid

Année	Matière	Cursus	Bac+	Nature	Vol. (H-TD)
2014-15					
	Algorithmes parallèles	Sciences (FI)	4	Cours et TD	24h
	Sécurité des Systèmes d'exploitations	Sciences (FI et FA)	5	Cours et TP	15h
	Ingénierie du logiciel	ESIAG (FI)	5	Cours et TP	15h
	Sécurité	ESIAG (FI)	5	Cours et TD	25h
	Java distribué	ESIAG (FA-A)	4	Cours et TD	49h
	Java distribué	ESIAG (FA-B)	4	Cours et TD	49h
	Java distribué	ESIAG (FA-C)	4	Cours et TD	49h
	Parser/Lexer en Java	ESIAG (FI)	2	Cours et TD	7h
	Internet et Bureautique	Droit	4	TP	24h
2013-14					
	Sécurité des Systèmes d'exploitations	Sciences (FI et FA)	5	Cours et TP	15h
	Ingénierie du logiciel	ESIAG (FI)	5	Cours et TP	15h
	Sécurité	ESIAG (FI)	5	Cours et TD	25h
	Langage, complexité et calculabilité	ESIAG (FA)	4	Cours et TD	54h
	Projet Java	ESIAG (FI et FA)	3	Projet	3*5h
	Programmation fonctionnelle	ESIAG (F1)	1	Cours et TD	54h
2012-13					
	Sécurité des Systèmes d'exploitations	Sciences (FI et FA)	5	Cours et TP	15h
	Sécurité	ESIAG (FI)	5	Cours et TD	25h
	IHM et Java avancé	ESIAG (FI)	3	Cours et TD	49h
	IHM et Java avancé	ESIAG (FA 2/3)	3	Cours et TD	49h
	IHM et Java avancé	ESIAG (FA BI)	3	Cours et TD	49h
	Projet Java	ESIAG (FI et FA)	3	Projet	3*5h
2011-12					
	Sécurité des Systèmes d'exploitations	Sciences (FI et FA)	5	Cours et TP	15h
	Sécurité	ESIAG (FI)	5	Cours et TD	25h
	Sécurité des Systèmes d'exploitations	ESIGETEL	5	Cours et TP	22h
	Génie logiciel en Ada	ESIAG (FI)	1	Cours et TD	49h
	IHM et Java avancé	ESIAG (FI)	3	Cours et TD	49h
	IHM et Java avancé	ESIAG (FA)	3	Cours et TD	49h
	Projet Java	ESIAG (FI et FA)	3	Projet	2*5h
2010-11					
	Initiation aux technologies de l'information	Droit	3	Cours	22h30
	Excel, SGBD et Sécurité	Droit	4	Cours	22h30
	Internet et Bureautique	Droit	4	TP	120h
	Programmation et sûreté des systèmes répartis	Sciences	5	Cours et TP	15h
	Sécurité des Systèmes d'exploitations	Sciences	5	Cours et TP	15h
	Initiation à l'algorithmique en Ada	ESIAG (FA)	3	Cours et TD	49h
	Génie logiciel en Ada	ESIAG (FI)	3	Cours et TD	49h
	Projet ADA	ESIAG (FI)	2	Projet	5h

FIGURE 2 – Partie 2 de la synthèse du service effectué ; années 2010 à 2015.

de boucles), programmation d'algorithmes parallèles et modèle BSP, programmation parallèle BSML et MPI (une comparaison est alors effectuée) ; cet enseignement comprend un cours, des TD et des TP (avec des exemples d'exécutions de programmes parallèles sur la grappe du LACL).

Programmation Orientée Objet (Init Java) Dans ce cours est introduit la notion d'objet et d'héritage pour la programmation en Java. Une première utilisation de l'API Java (fichiers notamment) permet d'écrire des premières applications.

Langage, Complexité et Calculabilité Cet enseignement est une introduction à la conception des langages et à la calculabilité : expressions rationnelles, langages formels, automates, automates à piles et machines de Turing. Ce cours introduit aussi brièvement les problèmes de complexité. Des exemples de cas d'utilisation des outils Lex et Yacc sont également donnés.

Langages pour la programmation concurrente Il s'agit d'un cours de sémantique de la concurrence en deux parties. La première partie est consacrée aux sémantiques opérationnelles, dénotationnelles et axiomatiques des algèbres de processus comme CCS. La seconde partie présente des sémantiques d'évaluation des réseaux de Petri et des M-nets (réseaux de Petri de haut niveau).

Programmation en ADA Ce cours est le premier sur la programmation (impérative) suivi par les étudiants. On y aborde des notions concernant les structures de programmes et de données (affectations, structures de contrôle, procédures, tableaux

etc.) ainsi que des bases sur les paquetages et les exceptions.

IHM et Java avancé Ce cours présente l'API AWT/Swing Java pour la programmation d'IHM (Interface Homme Machine) en Java. Nous présentons la programmation événementielle, les fenêtres, les composants et conteneurs. Sont aussi introduites les notions d'interface, de classe abstraite, de classe anonyme et la généricité. Un projet (par binôme, tout au lond du deuxième semestre) est donné aux étudiants avec soutenance à la fin de l'année.

Génie logiciel en Ada Ce cours présente les notions de paquetage (avec notamment piles, files, arbres) et de généricité en Ada. Cela permet aux étudiants de mieux concevoir leurs programmes par séparation des tâches et d'avoir un peu d'abstraction sur leurs codes (qu'est-ce que l'API m'autorise à faire ? *etc.*) Les notions d'interface et d'implantation sont bien évidemment au cœur de cet enseignement. Un projet (par binôme, tout au lond du deuxième semestre) est donné aux étudiants avec soutenance à la fin de l'année.

Sécurité Ce cours est une introduction à la problématique de la sécurité. Y sont abordés les normes de sécurité, les algorithmes de cryptage et de génération de nonces, les protocoles de sécurité (identification) ainsi que leurs attaques (*man-in-the-middle*, rejoue, *etc.*), des attaques de logiciels (injection de code SQL, buffer overflow, bug de format *etc.*) Nous présentons aussi des moyens pour remédier à ces problèmes.

Sécurité des systèmes d'exploitations Dans ce cours du M2 SSI (sécurité), nous présentons les attaques systèmes (buffer overflow, bug de format, heap-overflow, attaque noyau, détournement des appels systèmes *etc.*) ainsi que des attaques de logiciels (injection de code SQL, attaque sur cookies, attaque par concurrence, bug de format, *etc.*) Nous présentons aussi des moyens pour détecter et remédier à ces problèmes.

Programmation fonctionnelle Cours d'initiation à la programmation fonctionnelle en OCaml en L1. Dans ce cours, nous présentons la notion de fonction récursive, les types de base, le polymorphisme ML, les fonctions d'ordres supérieurs, la gestion de listes, de tableaux (avec un peu de code impératif) et les structures de données récursives (essentiellement les arbres). Notons que ce cours d'accompagne d'un cours à la programmation impérative, aussi en L1.

Ingénierie du logiciel Dans ce cours, nous présentons comment gérer du code et un projet (diagrammes de Gantt, Graphe de Pert, méthodes COCOMO, *etc.*) notamment avec l'application à la gestion des meta-annotations Java (API d'apt) et à la correction des codes en logique de Dijkstra/Hoare avec JML (outils Krakatoa) ; création de Makefile ; gestion de projet avec ANT et Maven ; gestion concurrente du code source avec CVS et SVN.

Calculs distribués en Java Dans un premier temps, sont présentés lors ce cours de M1 ESIAG, les threads en Java ainsi que l'API "Concurrence" (pb des interblocages, sémaphores, futures, collections concurrentes, *etc.*. Puis nous abordons l'utilisation de la bibliothèque introspection, de reflexion ainsi que les "classes loader". Ce cours se continue avec l'utilisation d'objets distants en RMI et de JDBC (accès BD en Java) . Enfin, nous terminons par la programmation de clients/serveurs en TPC/IP et UDP en Java.

Analyse de fichier en Java Dans ce cours de L2, nous mettons en oeuvres les connaissances sur les automates et les grammaires pour faire de l'analyse de fichiers avec JFlex (lexer) et Jacc (parser).

3.3 Projet

3.3.1 Responsabilité

Mon expérience à la ESIAG-MIAGE de Créteil (qui dépend de l'UFR des Sciences Economiques et Sociales), me fait dire que réduire la charge de travail d'une responsabilité est possible par le partage de telles responsabilités, par exemple en déléguant la gestion des stages des apprentis (la MIAGE étant entièrement en Alternance en M1 et M2) ou la répartition des enseignements (le « qui fait quel cours »). Dans notre MIAGE, c'est ce qui est fait (en plus des responsabilités par année et par section CNU).

Nous avons aussi mis en place un système de tutorat qui fait que chaque enseignant est officiellement tuteur de 5 à 6 étudiants, tout au long de leurs cursus (sauf demande explicite de changement de tuteur de la part de l'étudiant ou de l'enseignant). Cela permet un suivi individualisé avec un entretien par semestre et la présence du tuteur lors des soutenances de stages. Ce suivi permet de repérer « rapidement » les étudiants qui décrochent et éventuellement de leur trouver une solution. L'avantage est que l'étudiant n'a qu'une personne à contacter en priorité et lui évite d'aller de bureau en bureau pour trouver ce qui lui manque. Ce travail de gestion du tutorat nécessite aussi une responsabilité.

Je me porte donc volontaire pour aider dans l'une ou plusieurs de ces responsabilités dans l'année (ou par la suite) : formation à distance, gestion du tutorat, répartition des cours, stages ou apprentis, *etc.*

Aussi par mon expérience à l'UFR de Sciences et de par ma formation d'informaticien, je peux aussi assurer des cours et/ou prendre des responsabilités dans des spécialités autres que MIAGE.

3.3.2 Présentation de nouveaux cours

La diversité des matières que j'ai enseignées, aussi bien dans les disciplines scientifiques que dans des formations hors de l'UFR de Sciences, montre que je n'ai jamais hésité à m'investir dans des domaines dont je n'étais pas spécialiste. Par cette

expérience, comme par ma bonne formation en Informatique, je me sens tout à fait prêt à m'adapter à n'importe quelle situation et donc à assurer avec plaisir de nouveaux enseignements (avec naturellement des préférences pour des cours que j'ai déjà effectués ou proches de mon domaine de recherche).

Néanmoins, je présente ci-dessous 4 cours qui à ma connaissance n'existent pas tels quels (ou très peu existant) et qui pourraient un jour être mis en place.

Cours de programmation du Big-data L'idée de ce cours est de sensibiliser les étudiants aux problématiques du Big-data (`http://big-data-fr.com/`) et aussi à les former pour programmer des applications liées au grand nombre de données. Des formations complètes (entièrement dédiées) sur le Big-data n'existent pas vraiment et il existe encore peu de cours sur cette "nouvelle" problématique. Je n'ai jamais fait un cours sur ce domaine, mais c'est bien évidemment intéressant et très motivant.

Dans ce cours, nous pourrions introduire les problèmes d'éthique (notamment sur les outils dits de "Social Media Analytical" comme ceux utilisés dans les campagnes electorales américaines), la publicité sur l'internet et l'écosystème du Big Data dans son ensemble. En lien avec des cours de maths et de gestion, des rappels sur le Machine-Learning et le Graph-mining. Et montrer les applications comme dans les compétitions ouvertes (`http://www.kaggle.com/competitions`) ou sur l'utilisation du Big-data en science (détection/simulation de propagation de maladies, optimisation des réseaux de transport, *etc.*). Quelques principes de sécurité pourraient aussi être abordés comme la détection d'attaque dans les réseaux.

D'un point de vue plus informatique, ce cours pourraient être l'occasion de présenter les outils et algorithmes du Big-data comme les fameux Hadoop/MapReduce [1] mais aussi MapReduce et la manipulation de grands graphes avec Google's Pregel/Giraph (`https://github.com/aching/Giraph` [2], Hama, *etc.* Cette partie du cours prendrait donc le point de vue d'utiliser des langages de programmation dédiés à la manipulation des grands graphes (indexation/recherche de documents , Ranking et filtrage collaboratif, exploration des réseaux sociaux, *etc.*) et aux données (NoSql, *etc.*). Ce cours peut s'apparenter à un cours classique de calcul distribué/parallèle mais il est en réalité bien dédié à la manipulation d'un grand nombre de données et non à du calcul numérique de type HPC (simulation, FFT, calcul matriciel, etc.).

Enfin, après avoir montré comment programmer des applications en Big-data, nous pourrions monter des outils généralistes dédiés à la visualisation comme Tulipe `http://tulip.labri.fr/TulipDrupal/` ou plus spécialisés.

L'utilisation de données libres françaises `http://www.data.gouv.fr/` serait l'occasion de montrer le potentiel de telles appplications.

Cours de gestion et calcul « green » (M1 ou M2) L'idée serait de faire un cours du genre « Green-IT et normes de qualité ». Je n'ai jamais fait un cours sur ce domaine, mais cela me semble intéressant et motivant. Le livre « Green IT ; les meilleures pratiques pour une informatique verte » (de Adrien Porcheron, Christophe Corne, Pénélope Guy) me semble être une bonne base de travail pour monter un tel cours.

La gestion des déchets informatiques ainsi que l'écriture de programmes ou la configuration des architectures (logiciel OS, le web-developpement durable) et matériel (comme les systèmes de ventilation) afin de réduire la consommation électrique (bilan carbone), semblent prometteuses en terme de métiers d'avenir. On trouve aussi la gestion du personnel (déplacement contre télé-travail), la récupération de la chaleur (cas classiques des super-calculateurs et des centre de cloud-computing, Amazon, ou comme les centres de Google dans le nord des USA), le problème de la gestion des stocks du matériel (durée de vie des composants et question sur l'intérêt réel de changer si régulièrement de machines), *etc.*

Lors de mes cours d'introduction à l'informatique en Droit, j'avais essayé de sensibiliser les étudiants sur la consommation des machines et des réseaux, par exemple, dans le cas de l'envoi de gros emails « humouristiques » (la pièce jointe) mais inutiles. Mais un cours entier sur cette problématique du calcul vert est vraiment passionnant.

Pour ce qui est des normes de qualité, je pense aussi naturellement au « Common Criteria certification », notamment les normes EAL (1 à 7) `http://www.commoncriteriaportal.org/` qui est un argument pour le génie logiciel (qualité du code par le test voir les outils formels).

Cours de vérification et de sécurité en M1 ou M2 Suite à mon travail de recherche sur la vérification déductive de programmes BSP et en continuité avec mon cours de sécurité, j'aimerais mettre en place un cours de vérification non pas basé sur les éléments théoriques de la vérification (bien qu'un minimum soit nécessaire pour comprendre les techniques) mais sur la vérification concrète de problèmes. En effet, de nombreuses failles de sécurité ont pour origine une erreur de programmation ou un comportement non attendu du programme. Erreurs qui pourraient être détectées avec des outils formels. L'introduction d'assertions logiques dans les programmes me paraît être une bonne base de travail, en Java comme en C. Ensuite, la vérification par model-checking ou par génération d'obligations de preuves (VCG) pourraient être étudiée pour montrer les avantages et les limites.

Différents outils comme F-soft (`http://www.nec-labs.com/research/system/systems_SAV-website/index.php`) pour le model-checking de programmes C ou encore Frama-C (`http://frama-c.com/`) en tant que VCG (et ses autres outils d'analyses) pourraient être des exemples concrets d'utilisation (les deux ont leurs propres langages d'assertions en C). En Java, prenant en

1. MapReduce est basé sur le paradigme des *skeletons* que j'étudie depuis quelques années
2. Outils basés sur le modèle d'execution BSP qui est la base de mon travail de recherche

compte le langage standard d'assertion logique (JML), nous trouvons PathFinder (`http://javapathfinder.sourceforge.net/`) pour le model-checking et Key (`http://www.key-project.org/`) en tant que VCG.

Le cours se ferait plutôt sur des exemples concrets où des erreurs entraînent des failles de sécurité. Si ce cours fonctionne correctement, il pourrait être intéressant de l'étendre à la vérification de programmes concurrents ou parallèles (distribués). Des outils comme MPI-SPIN (`http://vsl.cis.udel.edu/mpi-spin/`) ou ISP (`http://www.cs.utah.edu/fv/ISP-Release`) pourraient être utilisés. Bien qu'il existe déjà beaucoup de cours de vérification, encore trop peu se basent sur un mixe des méthodes à utiliser et sur des erreurs concrètes de sécurité.

Cours de programmation parallèle dès la L1 ou L2 Du fait de mon parcours de recherche en programmation fonctionnelle parallèle, j'aimerais réussir à mettre en place, dès le L1 ou le L2, un cours d'initiation à la programmation et à l'algorithmique qui soit dès le départ basé sur du parallélisme.

En effet, après plusieurs années d'expérience de cours en M1/M2 sur les architectures parallèles ou la programmation parallèle, je pense qu'expliquer un modèle simple de parallélisme qu'est BSP (décrit dans la partie recherche) n'est pas plus compliqué qu'expliquer ce qu'est un objet (héritage, polymorphisme, *etc.*) ou les pointeurs en C.

Ce cours permettrait d'avoir au plus tôt des esprits habitués à une programmation parallèle structurée. J'y vois plusieurs avantages : (1) On pourrait prendre en compte, au plus tôt, les clusters et les architectures multi-cœurs lors des TP ; (2) Alors que le parallélisme n'est généralement introduit qu'en M1 ou M2 avec notamment une confusion avec les réseaux et la concurrence, nous pourrions faire réfléchir les étudiants sur les coûts des communications et comment structurer leurs programmes afin d'éventuellement les optimiser par la suite ; (3) Le parallélisme introduit naturellement des problèmes de combinatoire (notamment le placement des données sur les processeurs) ce qui complète bien un cours d'introduction à l'algorithmique ; (4) Par la suite, les cours d'algorithmiques avancées (*e.g.*, algorithmes sur les graphes) seraient facilement complétables, lors de TD, à la parallélisation des dits algorithmes.

Ce cours pourrait se faire sur un langage de programmation de haut-niveau (Java ou OCaml) où le fonctionnement des communications est caché (*via* une *serialization* des valeurs à envoyer). Ce type de cours sera d'autant plus facile à mettre en place que des étudiants auront bientôt déjà eu des cours de programmation au lycée (l'attendu cours d'Informatique en série S).

4 Activités de recherche

4.1 Thématiques de recherche

J'effectue mes travaux de recherche dans l'équipe « Spécification et Vérification de Systèmes » (dirigée par Catalin Dima) dans le thème « Programmation parallèle de haut-niveau » dirigé par Gaétan Hains et axé sur la conception de langages de haut-niveau pour la programmation parallèle, leur vérification (sûreté et preuves de correction des programmes [56, 58]) et l'utilisation du calcul haute-performance dans le model-checking, les protocoles de sécurité, le big-data ou le cloud-computing [C6].

Mes travaux se sont donc orientés vers la sécurité au sens large que lui donne l'ACI Sécurité et Informatique, qui a pour objectif « de fortement dynamiser la recherche sur l'ensemble des aspects de la sécurité et de la sûreté des systèmes informatiques, des systèmes informatisés et des systèmes d'informations ». Cette orientation se fait donc naturellement dans la continuité de mes travaux de thèse sur la programmation parallèle de « haut-niveau », le *meta-* et *grid-computing* [C5] [R7].

Mon travail de recherche est bien évidement plus longuement détaillé dans mon mémoire d'HDR :

`http://www.lacl.fr/gava/papers/hdr_gava.pdf`

4.2 Valorisation et animation de la recherche

4.2.1 Publications

Toutes mes publications sont disponibles sur ma page web. Les résultats ainsi que la liste complète de mes publications sont donnés par la suite. Mes publications sont aussi sur des pages de référencement :

- DBLP : `http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/g/Gava:Fr=eacute=d=eacute=ric.html`
- ACM : `http://dl.acm.org/author_page.cfm?id=81384615456&coll=DL&dl=GUIDE&CFID=100826674&CFTOKEN=24807858`
- Microsoft academy : `http://academic.research.microsoft.com/Author/775929/frederic-gava`
- Google Scholar : `http://scholar.google.fr/citations?user=6GviDPwAAAAJ&hl=fr`
- Research Gate : `https://www.researchgate.net/profile/Frederic_Gava`

4.2.2 Projets

- avec des collègues du LACL (J. Tesson), du LIFO (F. Loulergue, W. Bousdira et J.-M. Couvreur), de l'IBISC (F. Pommereau et F. Belardinelli), de l'IRIT (J.-G. Smaus) et du LRDE (A. Duret-Lutz), nous avons déposé cette année le projet TORTUES auprès de l'ANR, projet dont l'objectif est la vérification d'algorithmes de model-checking distribués pour l'extraction d'un model-checker certifié avec application pour l'analyse de gros modèle (big-data).
- J'étais membre du projet SPREADS (`http://spreads.fr/`), 2007-2010, financé par l'ANR Sécurité, dont l'objectif est d'étudier, définir et vérifier des systèmes de sauvegardes de données en utilisant la technologie « pair-à-pair » (P2P) ;
- J'étais responsable du projet VEHICULAIRE (`http://www.lacl.fr/gava/vehiculaire/vehiculaire.html`), 2006-2010, financé par la fondation « Digiteo, triangle de la physique » de la région Île-de-France, qui a pour but la conception d'un nouveau logiciel de *model-cheching* parallèle pour une classe de haut-niveau de réseau de Petri ;
- J'ai été membre du projet de l'ACI « Jeunes Chercheurs » 2004-2008 « Programmation parallèle certifiée » (PROPAC, page web à `http://wwwpropac.free.fr`) dont l'objectif est de fournir un environnement de programmation permettant la production et l'exécution de programmes parallèles certifiés ;
- J'ai été membre du projet de l'ACI Grid 2001-2004 « Coordination et répartition des applications multiprocesseurs en Objective Caml » (CARAML, page web à `http://www.caraml.org`) qui visait au développement de bibliothèques pour le calcul haute-performance et globalisé autour d'Objective Caml.

4.2.3 Logiciels

Plusieurs bibliothèques ont été développées dans le cadre de mes recherches, et sont librement téléchargeables :

La **BSMLlib** (`http://bsmllib.free.fr`) pour le langage OCaml. Elle autorise la programmation parallèle fonctionnelle suivant le modèle BSP. Une nouvelle version, facilement extensible grâce à l'utilisation des modules de OCaml, est prévue pour cette année (3 implantations sont déjà disponibles : MPI, PUB et TCP/IP). Elle comportera toutes les dernières améliorations de notre recherche : nouvelles primitives, opérations de composition parallèles (superposition et juxtaposition), exceptions et entrées/sorties parallèles, structures de données parallèles, outils de prévision des performances ainsi qu'une nouvelle bibliothèque standard. Cette bibliothèque est utilisée pour l'enseignement du parallélisme dans les universités d'Orléans et Paris-Est Créteil. Licence GPL.

MSPML (`http://mspml.free.fr`) pour le langage OCaml permet la programmation parallèle suivant le modèle MPM. Une nouvelle version est prévue pour cette année. Elle comportera aussi une implantation modulaire et une opération de composition parallèle.

La **DMMLlib** (`http://dmmllib.free.fr`) pour le langage OCaml permet la programmation fonctionnelle de méta-ordinateurs (grappe de machines parallèles). Il existe une version de travail.

La **Certified BSMLlib,** un sous-ensemble important de la bibliothèque standard de la BSMLlib mais certifié par l'assistant de preuves Coq. Les fonctions parallèles sont spécifiées, puis construites de manière certifiée par les tactiques de Coq et enfin extraites en code OCaml pouvant utiliser les primitives de la BSMLlib.

BSP-WHY Outil VCG pour la vérification déductive d'algorithmes BSP, en licence GPL : `http://lacl.univ-paris12.fr/fortin/BSP-why/index.html`.

4.2.4 Rapports de lecture

— Revues :
  - SCPE (« Scalable Computing : Practice and Experience »), SWPS Publisher (anciennement appelée PDCP, « Parallel and Distributed Computing Practices », NOVA Sciences Publisher) ;
  - CLSS (« Computer Languages, Systems and Structures »), Elsevier Publisher ;
  - HPCN (« High Performance Computing and Networking »), InderScience Publishers ;
  - TPDS (« Transactions on Parallel and Distributed Systems »), IEEE Publishers ;
  - TSI (« Technique et Science Informatiques »), Hermes Lavoisier Publishers ;
  - PPL (« Parallel Processing Letters »), World Scientific Publisher ;
  - Concurrency and Computation : Practice and Experience, Wiley Publisher.
  - JPP (« Journal of Parallel Programming »), Springer.

— Conférences :
  - ICCS (« International Conference on Computational Science »), 2007-2010 ;
  - HPCS (« High Performance Computing and Simulation »), 2011, 2012 et 2013 ;
  - PDCN (« Parallel and Distributed Computing and Networks »), 2011.
  - Symposium TFP (« Trends in Functional Programming ») 2005 ;

— Workshops
  - PaPP (« aPplications of declArative and object-oriented Parallel Programming ») 2004–2006 affilié à ICCS ;
  - APDCM (« Advances in Parallel and Distributed Computational Models ») 2009 affilié à IPDPS (« Parallel and Distributed Processing Symposium ») ;

4.2.5 Membre de Comités de programme

— Conférence ICCS 2011-15 ;
— Workshop HLPP (« High-level Parallel Programming and Applications ») 2013 ;
— Workshops PaPP 2007, 2008, 2011 et 2012 ;
— Workshops WLPP (« Language-Based Parallel Programming ») 2007, 2009, 2011, 2013 et 2015 ; affilié à la conférence PPAM (« Parallel Processing and Applied Mathematics ») ;
— Workshop HPDFA (« High-Performance and Distributed Computing for Financial Applications ») 2010.

4.2.6 Organisateur et Éditeur invité

J'ai été chair et co-organisateur (avec Dr. Anne Benoit, ENS Lyon) des workshops PAPP 2009 et 2010 affiliés à la conférence ICCS. J'ai été co-organisateur (avec Pr. Gaétan Hains, LACL et Pr. Kevin Hammond, University of St. Andrews) du workshop HLPGPU (« High-level programming for heterogeneous and hierarchical parallel systems ») 2011 affilié à la conférence HiPEAC (« High-Performance and Embedded Architectures and Compilers »). J'ai été co-chair et co-organisateur (avec Dr. Julien Tesson, Université technologique de Kochi) du workshop FraDeCoP 2012 (« Frameworks for the Development of Correct (parallel) Programs ») qui a eu lieu le 15 mai à l'UPEC. J'ai aussi été co-chair et co-organisateur (avec Dr. Abdelhamid Mellouk) d'une journée de séminaires communs entre le LACL et le LISSI (2 laboratoires de l'UPEC).

Co-éditeur (avec Pr. Gaétan Hains, LACL et Pr. Kevin Hammond, University of St. Andrews) invité à la revue Parallel Processing Letters (PPL, Volume 22, Number 2, June 2012) pour une sélection de papiers du workshop HLPGPU 2011.

4.2.7 Encadrements

Stages de M2

Imane Rahmoun :, « Preuves formelles d'algorithmes séquentiels et distribués de model-checking (NDFS et Tarjan). », master UPMC, 2014, 50% de l'encadrement avec Julien Tesson (LACL).

Mohamed Menasria :, « Extension et traduction d'un langage de description des protocoles de sécurité vers une algèbre de réseaux de Petri », master SSI, UPEC, 2014, 50% de l'encadrement avec le Pr Franck Pommereau (IBISC).

Valentin Samir :, « Conception d'un algorithme parallèle de model-checking de protocoles de sécurité pour architectures hybrides (grappes de multi-coeurs) », master recherche MPRI (venant de l'ENS Cachan), 2013 , 50% de l'encadrement avec le Pr Franck Pommereau.

Souad Kherroubi :, « Traduction d'un langage de description des protocoles de sécurité vers une algèbre de réseaux de Petri », master recherche NSI, Orsay, 2013, 50% de l'encadrement avec le Pr Franck Pommereau.

Arthur Hidalgo : « Preuve formelle d'algorithmes parallèles pour la génération de l'espace d'état et la vérification de protocoles de sécurité », master recherche NSI, Orsay, 2012, 100% de l'encadrement.

Jean Fortin : « Optimisation Certifiée de code BSP », master recherche MPRI (venant de l'ENS Lyon), 2008, 100% de l'encadrement.

Ilias Garnier : « Nouvelle implantation de la primitive BSML de superposition et application à l'implantation de squelettes algorithmiques », master recherche MPRI Paris VII, 2008, 100% de l'encadrement.

Otmane Bouziani : « Implantation d'une bibliothèque de programmation fonctionnelle BSP dans un environnement de méta-computing », master recherche SSI, Paris XII, 206, 100% de l'encadrement.

David Billiet : « BSML : Implémentation modulaire et prévision de performances », master recherche NSI, 2004, Orsay, 50% de l'encadrement avec le Pr. Frédédric Loulergue.

Thèses

Victor Allombert : 100% de l'encadrement. Titre « Multi-BSML, development of a functional language for multi and hierarchical parallel programming : semantics, typing and tool for deductive verification ». Soutenance prévue en 2016/17.

Jean Fortin : 100% de l'encadrement. Titre : « BSP-Why : a Tool for Deductive Verification of BSP Programs » ; soutenance le 14 octobre 2013 avec le jury suivant :

- Pr. Roberto di Cosmo, président, Université de Paris 7
- Dr. Hab. Jean-Christophe Filliâtre, rapporteur, Université de Paris-Sud/CNRS
- Dr. Alan Stewart, rapporteur, Queen's University of Belfast
- Pr Sandrine Blazy, rapporteur, Université de Rennes
- Pr. Jan-Georg Smaus, Université de Toulouse (Paul Sabatier)
- Pr. Bart Jacobs, Katholieke University of Leuven
- Pr. Elisabeth Pelz, Université de Paris-Est
- Dr. Hab. Frédéric Gava, directeur, Université de Paris-Est

Michael Guedj : co-encadrement avec le Pr. Franck Pommereau (IBISC Evry) ; 75% de l'encadrement. Titre : « BSP Algorithms for LTL & CTL* Model Checking of Security Protocols » ; soutenance le 11 octobre 2012 avec le jury suivant :

- Pr. Catalin Dima, président, Université de Paris-Est
- Pr. Frédéic Loulergue, rapporteur, Université d'Orléans
- Pr. Jean-François Pradat-Peyre, rapporteur, Université de Paris VI
- Pr. Laure Petrucci, Université de Paris-Nord
- Pr. Hanna Klaudel, Université d'Evry
- Pr. Gaétan Hains, Université de Paris-Est
- Pr. Franck Pommereau, co-directeur, Université d'Evry
- Dr. Frédéric Gava, co-directeur, Université de Paris-Est

Louis Gesbert : co-encadrement avec le Pr. Frédéric Loulergue (LIFO Orléans), 50% de l'encadrement, Titre : « Développement systématique et sûreté d'exécution en programmation parallèle structurée » ; soutenance le 5 mars 2009 avec le jury suivant :

- Pr. Olivier Michel, président, Université de Paris-Est
- Pr. Emmanuel Chailloux, rapporteur, Université de Paris VII
- Pr. Jocelyn Sérot, rapporteur, Université de Clermond-Ferant
- Pr. Zhenjiang Hu, National Institute of Informatics, Tokyo
- Pr. Frédéric Loulergue, directeur, Université d'Orléans
- Dr. Frédéric Gava, co-directeur, Université de Paris-Est

4.3 Résumé de mes travaux

Les besoins en puissance de calcul dans quelque domaine que ce soit (simulation, modélisation, vérification, traitement d'images, fouille de données, Big-Data *etc.* [53]) vont croissant et le parallélisme, partant du vieil adage selon lequel « l'union

fait la force », est une tentative de réponse toujours d'actualité. La conception d'algorithmes et de langages adaptés est donc un sujet de recherche actif, nonobstant la fréquente utilisation de la programmation concurrente [7, 50, 62].

Mon travail de recherche se décompose en trois axes distincts, mais qui comportent une trame commune : une forme de parallélisme structuré appelé BSP [20, 75] qui décrit une machine parallèle comme étant plusieurs unités de calcul communicantes et fortement synchrones. Ce modèle permet notamment une études des coûts des algorithmes ainsi que leur simplification.

Mon premier axe de recherche se porte sur la sémantique, l'implantation et l'extension d'un langage de programmation fonctionnelle BSP appelé BSML. J'étudie aussi l'adaptation et l'utilisation de ce modèle BSP et du langage BSML pour le meta-computing [R7] [R9], les machines parallèles hiérarchiques [C5] et l'implantation de patrons (squelettes/*skeletons* à la MapReduce) algorithmiques pour une programmation parallèle de haut-niveau [N1] [R3]. Le second axe est la création et l'étude formelle d'outils pour la vérification de programmes BSP, fonctionnels [R10] ou non [W4] notament pour l'application aux algorithmes de graphes (Big-Data). Mon dernier et nouvel axe est l'étude et la construction sûre d'algorithmes BSP pour la vérification de type model-checking [C1] de protocoles de sécurité [W2] [C4] [C2] [R2]. En ce sens, j'essaie d'utiliser les résultats des deux premiers axes afin de prouver les algorithmes et de les implanter de manière sûre [C3].

4.3.1 Conception et étude d'un langage de programmation fonctionnelle BSP

En collaborations avec le Pr. Frédéric Loulergue du LIFO (Orléans) et dans la continuité de ma thèse, je travaille sur une extension parallèle appelée BSML du langage « fonctionnel » (de haut-niveau) OCaml. Cette partie de mon travail de recherche se porte sur la conception, la sémantique et les bibliothèques applicatives de BSML. Nous avons co-encadré la thèse de Louis Gesbert [45] (soutenue en 2008 avec financement projet ANR Propac 2004-2007).

Pendant ma thèse [M2], j'ai dans un premier temps effectué une étude sémantique du langage BSML. Diverses sémantiques (grands-pas, petits-pas, machine-abstraite, *etc.*) ont été étudiées pour mettre en exergue différentes propriétés du langage : la confluence et la preuve de programmes pour la sémantique à grands-pas, les coûts BSP pour la sémantique à petits-pas et la compilation sûre pour la machine abstraite [C24] [C20]. Cela nous a permis d'obtenir une nouvelle implantation plus sûre et plus modulaire [C16] : grâce à cette étude sémantique, nous sommes à même de déduire l'implantation de tout le langage BSML à partir d'un petit module dont l'implantation peut être effectuée avec TCP/IP, MPI ou les bibliothèques de calculs BSP que sont PUB et BSPlib. Notons que ce travail sémantique a été étendu et en partie réalisé en Coq dans [23]. La sémantique à petits-pas nous a aussi permis d'étudier le typage du langage BSML afin de garantir (classique) la sûreté d'exécution des programmes. Un premier travail a été fait dans [N5], [C23] et [R8] puis après les extensions décrites ci-dessous et en se basant sur la nouvelle syntaxe du langage BSML [W3] (qui rend les programmes plus lisibles), un nouveau système de types a été défini dans [W1]. J'ai aussi travaillé sur une extension (avec des primitives) du modèle BSP pour les algorithmes avec utilisation d'une mémoire externe [33] (I/O) [R6].

J'ai ensuite étudié différentes extensions de BSML. En effet, le langage BSML était à l'origine défini comme un langage de programmation purement fonctionnelle en tant qu'extension BSP du langage ML. Mais ce langage ML contient différents traits impératifs dont il fallait étudier leurs interactions avec le parallélisme explicite de BSML — celui-ci étant implanté comme une bibliothèque d'une version du langage ML qu'est OCaml. Les (classiques) références ML constituent cette première extension dont j'ai étudié la sémantique en BSML dans [C19]. La deuxième extension concerne les entrées-sorties (E/S) vers les fichiers de la machine parallèle (extension que l'on trouve dans MPI-2). J'ai étudié l'implantation de nouvelles primitives d'E/S pour BSML dans [C17] ainsi que la sémantique et le modèle de coût dans [R6]. Nous avons aussi étudié une nouvelle primitive de BSML qui permet de composer les programmes ou de programmer plus aisément les algorithmes diviser-pour-régner parallèles[79]. La sémantique a été étudiée dans ma thèse et a permis une implantation à base de *threads* [C13]. Grâce à cette primitive nous pouvons aussi aisément simuler la précédente primitive de composition parallèle du langage BSML [N3]. Une nouvelle implantation plus efficace à base d'une transformation CPS [74, 59] a été faite dans [W5] et [R3]. Celle-ci a permis l'implantation de patrons algorithmiques de flots en simulant le flot comme une composition de micro-programmes BSML. Une autre extension est le filtrage de valeurs (notamment parallèles). Ceci permet d'augmenter fortement l'expressivité du langage BSML en rendant les programmes plus naturels à lire (notamment les cas d'algorithmes d'équilibrage de charge). Ce travail a été initié dans [C21] puis complété dans la thèse de Louis Gesbert (que j'ai co-dirigée avec Frédéric Loulergue, financement projet ACI JC Propac) et [W3]. Enfin, nous avons étudié une dernière extension non fonctionnelle du langage ML que sont les exceptions (une structure de contrôle nécessaire à tous les langages modernes). Les exceptions permettent d'interrompre le contrôle normal d'un programme notamment celui d'un nœud de calcul d'une machine parallèle. Ceci peut rendre les programmes BSML non sûrs car une seule machine peut ne plus se synchroniser avec les autres nœuds. L'implantation des exceptions parallèles a été étudiée dans [N2], [C14] et [R4].

Avec toutes ces primitives, j'ai implanté des structures parallèles de données complexes en BSML [R5] : ensembles, dictionnaires, piles, *etc.* avec un équilibrage des charges dynamique et semi-automatique. Une comparaison d'implantation d'algorithmes matriciels entre BSML et C+MPI a aussi été faite dans [C8] ainsi qu'une implantation de patrons algorithmiques (à la MapReduce) data-parallèles dans [N1]. Une étude de cas sur la comparaison entre les coûts et les performances réelles (sur la grappe du LACL) de petits programmes BSML typiques a été effectuée dans [C12]. Toutes ces études ont montré que le langage BSML rend possible une programmation simple, élégante et autorisant tout de même la génération de programmes performants.

Avec Louis Gesbert, nous avons notamment défini un nouveau système de typage des programmes BSML [W1] qui rend

sûre l'exécution des programmes BSML, implanté un dispositif original pour rattraper les exceptions [R4] dans ce langage de programmation parallèle et enfin redéfini une syntaxe qui rend la programmation plus aisée. Nous développons des exemples non-triviaux de notre langage. Par exemple, avec Sovanna Tan (LACL), nous avons implanté une bibliothèque [2] de patrons (à la MapReduce) algorithmiques [N1] et d'opérations [43] matricielles [C8]. J'ai aussi implanté en BSML des structures de données standard [R5] et prouvé expérimentalement que notre langage conserve la bonne propriété des programmes BSP qu'est la prédiction des temps d'exécution [C12]. Durant ma thèse, j'ai aussi travaillé sur la preuve de programmes [R10], travail qui a été étendu par Tesson *et al.* [46, 78]. Notre travail sur BSML a inspiré plusieurs bibliothèques de calcul BSP [36, 55, 67].

Enfin, nous avons étudié l'extension de BSML dans le cadre du *grid-computing* et notamment le *meta-computing* (coût, sémantique et implantation) : quand plusieurs grappes de calculs sont utilisées conjointement en tant qu'une seule machine de calcul. Pour cela, nous avons proposé une extension de BSML en deux niveaux, le premier étant un langage de coordination proche de BSML mais plus asynchrone que nous avions précédemment étudié dans [C15], [C22] et [R9]. Le second niveau étant BSML. Cette étude a été faite dans [C18] et [R7]. Elle a servi de base à un nouveau langage appelé SGL pour les machines hiérarchiques (plus que deux niveaux de parallélisme), dans le cadre d'une thèse CIFRE encadré par Gaétan Hains, où j'ai participé à l'étude de cas de l'implantation de patrons algorithmiques data-parallèles [C5] à la MapReduce. Nous essayons d'appliquer ces outils pour l'analyse de grande données (Big-Data), notamment l'analyse de requêtes pour grand fichiers XML [52].

4.3.2 Vérification de programmes BSP

Dans ma thèse, j'ai étudié la vérification formelle (dans le sens « machine-checked », aussi appelée « mechanized » [56, 58, 21]) de programmes BSML [R10]. Il s'agissait d'axiomatiser les primitives BSML dans l'assistant de preuves Coq (cette axiomatisation se basait sur l'étude sémantique de BSML susmentionnée) et ensuite d'effectuer des preuves constructives de spécifications de calculs BSP (notamment calculs des préfixes et tris de données). Grâce à l'extraction de Coq (basée sur l'isomorphisme de Curry-Howard), des programmes BSML certifiés peuvent être générés. Ces programmes ont la garantie d'effectuer correctement les calculs. Les bases de ce travail ainsi que la preuve mécanisée de premiers programmes ont été fait dans [R10] puis une bibliothèque de programmes certifiés a été réalisée dans [N4] et [A1]. Tesson *et al.* ont continué et amélioré ce travail dans [46, 78].

Dans la cadre de la thèse de Jean Fortin (soutenance en 2013 et financement ENS-Lyon), nous travaillons sur la sémantique formelle (« mechanized ») en Coq d'un langage de spécification et de programmation d'algorithmes BSP [C10] [C11]. Nous avons aussi prouvé formellement une optimisation de code [C9]. Le travail a été amélioré pour la thèse de Jean Fortin, notamment avec une nouvelle primitive de composition parallèle. Cela permet d'étudier tous les programmes MPI n'utilisant que des opérations collectives [25, 48].

Nous en avons aussi déduit un outil appelé BSP-WHY [W4] qui est une extension du célèbre outil WHY [41, 22] du LRI (Orsay) pour la vérification de programmes à l'aide de la logique de Hoare. BSP-WHY nous permet donc de vérifier (prouver des propriétés du programme ou sa correction) des programmes BSP annotés dans une extension de la logique de Hoare. Pour finir, BSP-WHY a été amélioré avec la primitive de composition parallèle [R1].

BSP-WHY fonctionne par une « séquentialisation » du code BSP-WHY en du code purement séquentiel WHY [39]. Nous nous basons sur le fait que dans les phases de calcul des super-étapes BSP, les calculs sont réellement indépendants et donc peuvent être exécutés séquentiellement [57]. Et les communications sont simulées par des copies mémoires. La première et principale difficulté est d'extraire correctement les phases de purs calculs. En effet, les exceptions comme les primitives de synchronisations peuvent apparaître « n'importe où », ce qui induit plusieurs passes d'analyses syntaxiques notamment pour la détection de codes parallèles pouvant être exécuté du fait de la présence d'exceptions. La seconde difficulté provient des variables qui interviennent dans plusieurs blocs de calculs et qu'il faut donc gérer sous la forme de tableaux (une valeur par processeur). Enfin, avec la primitive de composition parallèle, des sous-groupes distincts de processeurs pouvant se synchroniser [76], il faut gérer ces sous-groupes afin de ne pas chevaucher des calculs *a priori* indépendants.

Nous avons aussi appliqué cet outils à la vérification de d'algorithmes de state-spaces [C1] et notamment d'algorithmes (décrits ci-dessus) BSP et optimisés pour les protocoles de sécurités [C3]. Nous recherchons à l'appliquer à des algorithmes de grand graphes dans le cadre du Big-Data.

4.3.3 Algorithmes BSP pour le model-checking de protocoles de sécurité

Avec le Pr. Franck Pommereau du IBISC (Evry), nous avons co-encadré la thèse de Michael Guedj [51] (soutenance en octobre 2012 et financement fondation Digiteo) qui porte sur le design et l'implantation en BSP-Python [55] d'algorithmes BSP pour le model-checking de protocoles de sécurité [65, 5, 14, 70]. La modélisation des protocoles a été effectuée avec une algèbre de réseaux de Petri de haut-niveau définie dans l'outil SNAKE (et principalement développée en Python par Franck Pommereau) de l'IBISC . Nous utilisons des propriétés de structuration des protocoles pour optimiser les communications et l'utilisation mémoire par rapport à l'algorithme le plus couramment utilisé dans le cadre du model-checking parallèle (utilisation d'une fonction de partition des états sur les processeurs et chaque processeur n'est responsable que des états qui lui « appartiennent » suivant cette partition) [W2].

En effet, lors de la vérification formelle d'un protocole [31], la plus grande partie du travail est générée par l'attaquant. Distribuer et équilibrer sur les processeurs cette partie du calcul permet d'éviter des attentes entre les processeurs et de trop nombreuses communications (les fameuses « cross-transitions » [42]). Notre méthode de génération de l'espace des états (fondement du model-checking) se base sur la création d'un histogramme global de classes d'états, permettant de distribuer dynamiquement [61] (à chaque super-étape BSP) les états par classes afin d'équilibrer au mieux les calculs et les communications [W2]. Nous avons des premiers résultats sur le calcul de l'espace des états pour des protocoles de sécurité classiques et néanmoins non-triviaux [C7].

Nous avons ensuite adapté ce travail pour la vérification de propriétés de sécurité qui sont exprimées dans les logiques temporelles que sont LTL et CTL*. Nos algorithmes BSP se basent sur le travail de [18] : les formules LTL sont décomposées pour générer un graphe d'assertions logiques (une « proof-structure ») où l'on recherche des composants fortement connexes (SCC) non-valides à l'aide d'une adaptation de l'algorithme de Tarjan. Cet travail est étendu à CTL* en décomposant la formule et en appelant récursivement la vérification LTL (des « sessions LTL »).

En se basant sur notre partitionnement de l'espace des états, nous avons remarqué que les SCC étaient forcément purement locales aux processeurs mais aussi aux super-étapes [C4]. La vérification LTL devient alors aisée en exécutant en chaque processeur l'algorithme de Tarjan. Enfin, en cas d'une SCC qui invalide la formule (et donc trouve une erreur du protocole), nous avons développé un algorithme BSP pour reconstituer la trace de l'erreur depuis l'état initial. Pour la vérification CTL* [C2] [R2], nous avons remarqué qu'en validant temporairement une assertion (même si elle doit être plus tard invalidée) tout en continuant la construction de la proof-structure (même si cette partie du calcul peut être inutile) d'une session LTL, nous pouvions exécuter toutes les sessions LTL (qui correspondent à la construction de proof-structures) en « même temps » c'est-à-dire en faisant la recherche de SCC en largeur. Et ce, avec toujours le même nombre de super-étapes que pour la génération de l'espace des états. Finalement, si une session LTL « fille » découvre que l'assertion n'était pas en fait valide, il est alors aisé de remonter ce résultat à la session LTL appelante et ensuite de l'arrêter. Cette même session LTL pourra alors ensuite remonter ou non son résultat.

Pour terminer, nous avons utilisé l'outil BSP-WHY susmentionné pour vérifier [73, 68] nos algorithmes BSP [C3] d'espace des états (standards et ceux dédiés aux protocoles de sécurité). Nous avons donc déterminé les invariants de boucles (parallèles) et en rajoutant quelques assertions techniques, nous avons obtenu que toutes les obligations de preuves générées soient entièrement prouvées par des prouveurs automatiques (ceux supportés par WHY).

4.4 Perspectives « immédiates » de recherche

Nous donnons ici quelques pistes qui pourraient donner lieu à des stages de M2 et à des sujets de thèses (ou plus ?).

4.4.1 Programmation fonctionnelle BSP

Il n'y a, pour l'instant, que de petits exemples de l'utilisation de nos bibliothèques, en plus des bibliothèques standards qui fournissent déjà un certain nombre de cas d'utilisation. Toutefois, il est nécessaire d'avoir plus d'applications pour mener des expériences plus importantes. L'implantation d'utilitaires dans des domaines autres que purement informatiques, comme par exemple ceux décrits dans [20] (transformée de Fourier rapide, résolution de systèmes linéaires, inversion de matrices creuses *etc*.) ou bien la bio-informatique avec par exemple un algorithme BSP (avec entrées/sorties) de recherche de motifs de [38], permettrait de valider expérimentalement le langage BSML et créerait sûrement de nouveaux besoins. Plusieurs directions peuvent être suivies. La première est la conception et l'implantation d'autres structures de données parallèles telles que les listes ou les files de priorités parallèles [44]. Une seconde direction serait d'étendre la bibliothèque OCamlgraph (manipulation de graphes en OCaml) afin de pouvoir implanter, de manière modulaire, des algorithmes BSP sur les graphes [26].

Actuellement, le langage BSML souffre d'un sérieux défaut : en tant qu'extension parallèle de OCaml, il n'est implanté que sous la forme d'une bibliothèque applicative. Ainsi donc, seul le typage « à la ML » [86] est possible. Malheureusement, le parallélisme introduit des erreurs subtiles qui ne sont pas prises en compte par le typage ML (dédié aux programmes séquentiels). Le typage permet de garantir la bonne exécution des programmes et il est dommage qu'elle ne soit plus garantie pour les programmes parallèles. Lors de la thèse de Louis Gesbert (co-encadré par Frédéric Loulergue et moi-même), un système de types, ainsi qu'un algorithme de typage avaient été définis et prouvés corrects [W1]. Mais l'implantation n'avait été faite que pour un micro-langage, pour tester sa faisabilité. Il faudrait implanter ce typage pour le plus grand sous-ensemble possible de OCaml : il est clair que certaines constructions du langage (notamment les dernières comme les modules du permier ordre) ne pourront être implantées aussi vite que le typage de OCaml lui-même. Mais un large sous-ensemble typant (disons « Caml-light »+modules) serait un bon début. Nous avons fait une demande de post-doc à ce sujet auprès de notre université de tutelle.

La tolérance aux pannes pour le modèle BSP a déjà été traitée dans [54]. La documentation de la dernière version de l'Oxford BSPlib indiquait que la version suivante devait contenir la migration de processus appliquée à l'équilibrage de charges et à la tolérance aux pannes. Mais cette version n'a jamais été diffusée. L'actuelle implantation de la bibliothèque PUB propose une version basée sur TCP/IP permettant de faire migrer des processus, mais la tolérance aux pannes n'est pas évoquée. Bien entendu, il est possible d'avoir actuellement une tolérance aux pannes en BSML par le biais de notre implantation MPI et en utilisant MPICH-V [24]. La structuration des programmes BSP autorise toutefois d'avoir des solutions plus simples et sûrement plus efficaces. Il serait intéressant de pouvoir implanter une telle fonctionnalité.

4.4.2 Preuves de programmes parallèles

Avec l'outil BSP-WHY développé par Jean Fortin sous ma direction, nous sommes à même de prouver la correction de programmes (impératifs) BSP (avec sous-synchronisation). Mais pour l'instant, nous ne prenons en compte qu'une extension parallèle de WhyML qui est un mini langage algorithmique. Il faudrait maintenant étendre cet outil pour de vrais langages comme Why [40] l'a été pour C (plugging Jessie pour frama-C) et Java (l'outil Krakatoa [64]). Cela permettrait de prouver correct de vrais programmes de calculs scientifiques. Et puis naturellement, tester l'outil sur des exemples plus conséquents serait pertinent pour sa diffusion et son utilisation par de tierces personnes [3].

Lors de ma thèse, j'avais plutôt travaillé sur la preuve de programmes parallèles purement fonctionnels à l'aide de l'assistant de preuves Coq. Il semble pertinent de tenter une fusion entre cette approche et celle de l'outil BSP-WHY. De plus, les patrons algorithmiques (type data-parallèles comme MapReduce [32] ou en flot [47, 49]) sont très souvent utilisés dans le calcul haute-performance. Adapter nos approches à ce paradigme permettrait sûrement la création d'outils innovants pour la preuve de programmes par patrons [69].

Une direction complémentaire est la description d'analyses des performances d'un programme. L'analyse des performances d'un programme, communément appelée analyse de coûts, permet une estimation des programmes donnée en terme d'une métrique désirée telle que le temps ou l'espace : les prévisions de l'analyse sont alors une borne supérieure des évaluations du programme. C'est une certification de la bonne exécution dudit programme sur une machine réelle disposant des ressources nécessaires [85]. En effet, dans le contexte de grid/cloud-computing [6], une architecture parallèle peut refuser (ou accepter) l'exécution de tel ou tel programme (« transporter » par un agent mobile, par exemple) en fonction de sa formule de coût et des paramètres réels de la machine. Les programmes ne sont exécutés que si leurs «certificats de coûts» sont acceptables. Dans un projet comme Grid'5000, où plusieurs grappes sont accessibles et partagées par les utilisateurs [4], cet ordonnancement des processus parallèles permettrai d'éviter des programmes trop coûteux (trop longs à exécuter) sur certaines grappes. On trouve dans la littérature des analyses de coût de programmes séquentiels [84, 83] (même pour des programmes certifiés [16]). On trouve ce genre d'analyses pour les langages à patrons algorithmiques [30] mais pratiquement pas pour la programmation parallèle structurée et explicite [71, 87, 71, 28].

Les performances (coûts) des programmes parallèles qui pourraient être vérifiés par notre outil BSP-WHY. Il faudrait alors définir une extension de notre outil pour analyser automatiquement les programmes afin de faciliter l'insertion d'annotations de coûts comme dans [60, 9, 8]. Ce travail sera aussi à faire pour notre langage BSML. Ces analyses pourront soit se faire sur des programmes parallèles quelconques (et permettre au programmeur de rapidement constater la bonne réalisation de ses programmes), soit sur des programmes extraits par Coq (afin de certifier le programme et son coût, c'est-à-dire vérifier que le programme est correct vis-à-vis de la spécification et correct vis-à-vis de la complexité algorithmique).

4.4.3 Protocoles de sécurité

Dans le cadre de la thèse de Michael Guedj, nous avons développé des algorithmes parallèles pour la vérification (de type model-checking) de protocoles de sécurité. Pour modéliser nos protocoles, nous utilisions une algèbre de réseaux de Petri de haut-niveau appelé ABCD. Le principal problème est que nous n'avons qu'un nombre limité d'exemples de protocoles en ABCD, vu qu'il faut, pour l'instant, encoder manuellement les protocoles dans cette algèbre. De plus, encoder dans cette algèbre n'est pas « user-friendly ». Nous souhaitons donc avoir plus d'exemples, notamment tous ceux provenant des exemples d'autres outils (afin de pouvoir se comparer), comme entre autres ceux de SPORE ou AVISPA [4]. Mais aussi, à l'avenir, de pouvoir écrire plus facilement nos exemples de protocoles et surtout les propriétés logiques à vérifier (dans le sens model-checking) sur ceux-ci.

Nous souhaitons aussi pouvoir facilement modéliser des protocoles avec boucle (un agent renvoie son message jusqu'à une certaines condition d'acceptation des autres agents ou envoie des messages à « n » autres agents), de définir des canaux sûrs (pas d'attaquant sur ce réseau) ou avec un attaquant passif (l'attaquant ne peut que lire les messages) mais aussi de définir des réseaux mobiles *ad hoc*, *etc.* Il nous faudrait donc avoir un traducteur de format de définition de protocoles vers cette algèbre. Et éventuellement étendre ce travail pour le protocole non trivial de sauvegarde de fichiers en P2P crypté [72] du projet SPREAD, d'e-vote et de e-gouvernance [34] ou pour le routage dans les réseaux ad hoc (mobiles et volatiles) [3], les protocoles multi-parties [17], *etc.*

Tout ce travail est en cours avec un stagiaires de M2 que je co-encadre par Franck Pommereau : un stage pour la transformation de fichiers AVISPA en algèbre de Réseau de Petri (ABCD) et un autre stage sur l'adapatation de nos algorithmes pour les protocoles non-standards comme ceux pour réseaux mobiles.

4.4.4 Preuves d'algorithmes distribués de modèle checking

Avec l'aide de l'outil BSP-WHY, nous avons prouver différents algorithmes [77] (parallèles) du calcul de l'espace d'états (notamment des protocoles de sécurité), calcul qui est à la base du model-checking. Nous pensons étendre ce travail aux algo-

3. Un étudiant de M2 a utilisé l'outil mais sous notre direction et donc pas de manière vraiment autonome.

4. Notons que dans le cas de plusieurs programmes parallèles de différents utilisateurs (donc « indépendants »), l'utilisation des primitives de composition parallèles permettrai de faire partager les ressources d'une même grappe à ces programmes et ainsi ooptimiser l'utilisation du réseau ; ceci implique, par contre, un ordonnancement plus subtile des tâches

rithmes de vérifications de formules temporelles (comme LTL [35] ou CTL* [80]) et de pouvoir aussi y inclure des techniques de réductions : pour l'instant nous utilisons des ordres partiels [81] mais des techniques comme les BDD [29] ou les symétries nous semblent prometteuses et adaptables pour la problématique des protocoles de sécurité [10]. Bien évidement, nous nous intéressons aux model-checking distribué [12, 13] pour les (nombreux) cas ou la mémoire d'un seul ordinateur n'est pas suffisante pour contenir toutes les données (cas particulier de Big-data puisque ces données sont sous la forme d'un graphe). Ce travail fait partie du projet TORTUES que j'ai soumis cette année à l'ANR.

De plus, la plupart des bugs des outils de model-cheching se retrouvent non pas dans les algorithmes mêmes mais plutôt dans leurs implantations dans les langages tels que C ou Java. Il faudra donc continuer ce travail sur les algorithmes pour l'étendre à des implantations réelles ce qui peut conduire à bien des difficultés (comme des integer/buffer overflows).

Dans ce projet, nous souhaitons utiliser BSP-WHY et les axiomatisation de primitives parallèles en Coq de BSML [R10] afin de prouver des algorithmes distribué de model-checking et en extraire des programmes BSML efficaces (étendre le travail de [C1] et [35]).

4.5 Perspectives de recherches à plus long terme

Nous donnons ici des projets plus ambitieux qui pourraient donnés aussi de très bon sujets de thèses.

4.5.1 Verification deductive d'algorithmes et programmes pour le Big-Data

Au delà de l'écriture de programmes dédiés pour le traitement du Big-data, se pose la question de la validité des résultats [27, 66] et donc de la correction desdits programmes. En effet, imaginons un programme utilisant les données de réseaux sociaux pour, par exemple, trouver des personnes souffrant d'une maladie ; si celui-ci est bugué alors il y a un risque de rater certaines personnes malades ou d'effrayer des personnes en réalité saines. Ce problème de correction se pose d'autant que les algorithmes manipulant du noSQL (graphes ou ensemble de données) sont complexes et donc sensibles aux erreurs humaines.

Des paradigmes comme MapReduce, Pregel/Giraph (et autres) se basent actuellement sur des méthodes de traitement "à la BSP" ; par exemple en itirant sur tous les arcs d'un graphe orienté puis en communiquant aux noeuds les résultats temporaires et ce jusqu'à obtenir un point fixe (célèbre exemple du page ranking). Ce sont donc des algorithmes qui conviendraient à notre outil BSP-WHY. Mais pour cela, il faut extraire ces algorithmes depuis les programmes Java en MapReduce, Pregel, Hama, Giraph, *etc.* comme cela a été fait par exemple en Java séquentiel par Krakatoa [39]. Ce travail est souvent technique et comme tout outil dédié pour la preuve, très sensible à la sémantique des programmes et donc au moindre changement de ladite sémantique (opérationnelle) ou des ambigüités dans le langage.

De plus, les algorithmes du Big-Data ne se contentent pas d'être juste des algorithmes de graphes ; ils utilisent aussi massivement des propriétés de statisque. Or, à notre connaissance, ce genre de propriétés n'a jamais été étudié dans des outils de preuves (assistante de preuves ou prouveur automatiques) tels que Coq ou Z3, outil qui servent massivement à la vérification des programmes puisque les propriétés des programmes sont extraites dans les prouveurs (pour être prouvées). Il faudrait donc formaliser ces propriétés mathématiques dans un assistant de preuves, chose qui est très chronophage et très technique aussi.

4.5.2 Bibliothèques auto-optimisantes de patrons algorithmiques

Un problème majeur avec les patrons algorithmiques est l'inefficacité introduite lors de la composition de plusieurs patrons. Ceux-ci ajoutent des communications qui pourraient être évitées par le passage de données entre ces patrons. De nombreux travaux ont été réalisés pour éliminer de telles inefficacités en utilisant des techniques de transformation de programmes utilisant, malheureusement, un nombre pléthorique de règles [1]. Cela rend l'implantation efficace trop difficile et, plus important, la modification d'un seul patron ou l'ajout d'un nouveau patron entraînent des modifications générales de ces règles. Ce problème est partiellement réglé avec des bibliothèques de méta-programmation en C++ (utilisation des templates) [37] mais seulement sur des problèmes simples (la méthode donne néanmoins des résultats efficaces en pratique).

La conception et l'implantation d'une bibliothèque de patrons parallèles auto-stabilisants [5] en combinant la théorie de l'algorithmique constructive (ou formalisme de Bird-Meertens [19]) avec des techniques de modélisation (Markoviennes ou autres [15]) des performances des réseaux et des processus est un projet à long terme. Une première étape pourrait être la réalisation de ces patrons sur des machines parallèles (modèle BSP). Puis nous pourrions continuer avec des modèles de parallélisme à grains adaptables pour la grille et la tolérance aux pannes (une autre approche possible et intermédiaire serait le *peer-to-peer*).

4.5.3 Développement d'un BSML hiérarchique

La vue plate d'une machine parallèle comme un ensemble de machines séquentielles communicantes (comme BSP) reste utile, mais est aujourd'hui incomplète [11]. Par exemple, les GPUs ont une architecture maître-esclave et il existe des grappes de

5. On dit qu'un algorithme ou un « objet » est auto-stabilisant si quel que soit son état initial, il est capable de retrouver, de lui même, un état valide en un nombre fini d'étapes.

multi-cœurs dont les processeurs partagent les mêmes cartes réseau, ce qui peut entraîner une congestion des communications [11]. On peut aussi considérer des grappes de grappes de PCs (également connu sous le nom *meta-computing* [R7] où nous avons déjà proposé d'avoir une approche à deux niveaux : BSP sur les grappes et un autre modèle, plus asynchrone pour leurs gestion).

Partant de ce constat, les auteurs de [63] ont proposé un langage appelé SGL (pour scatte/gather language) qui est un langage basé sur des primitives maîtres-esclaves pour le modèle BSP hiérarchique de [82] : une variante multi-niveaux de BSP [82]. Ce modèle récursif représente les architectures parallèles comme des arbres de processeurs (chaque nœud d'une branche est à la fois un maître et un esclave, à l'exception des feuilles qui ne sont que des esclaves). Dans [C5], nous avons utilisé SGL pour implanter des patrons algorithmiques de données sur une grappe de multi-cœurs. Mais SGL ne semble pas approprié pour des algorithmes plus complexes tels que ceux pour le model-checking.

Une autre constatation concerne certaines primitives de BSML, notamment celles utilisées pour décrire les schémas de communications : elles sont trop difficiles à utiliser (les étudiants qui ont essayé BSML ont souvent été bloqués sur ces primitives) et elles ne permettent pas l'envoi de messages tout en continuant les calculs.

Dans les conclusions de mon mémoire d'HDR, je propose de nouvelles primitives de communications basées sur le typage de BSML (et donc moins « syntaxiques ») et deux solutions pour les architectures hiérarchiques. Elles n'ont pour l'instant été ni prouvées et ni étudiées sémantiquement (formellement).

Ce travail est commencé par la thèse de Victor Allombert que j'encadre.

5 Publications

Revues internationales (avec comité de rédaction)

[R1] J. Fortin and F. Gava. BSP-Why : a tool for deductive verification of BSP algorithms with subgroup synchronization. *Journal of Parallel Programming*. To appear. 2015.

[R2] F. Gava, M. Guedj et F. Pommereau. A BSP algorithm for on-the-fly checking CTL* formulas on security protocols. *Journal of SuperComputing*, 69(2) :(629–672, 2014. Version étendue de [C2].

[R3] I. Garnier et F. Gava CPS implementation of a BSP composition primitive with application to the implementation of algorithmic skeletons. *Parallel, Emergent and Distributed Systems*, 26(4) :(251–273), 2011. Version étendue de [W5].

[R4] L. Gesbert, F. Gava, F. Loulergue et F. Dabrowski, Bulk Synchronous Parallel ML with Exceptions. *Future Generation Computer Systems*, 26(3) :(486–490), 2010.

[R5] F. Gava. A Modular Implementation of Parallel Data Structures in BSML. *Parallel Processing Letters*, 18(1) :(39–53), 2008.

[R6] F. Gava. External Memory in Bulk-Synchronous Parallel ML. *Scalable Computing : Practice and Experience* (précèdement appelé *Parallel and Distributed Computing Practices*), 6(4) :(43–70), 2005. Version étendue et révisée de [C17].

[R7] F. Gava et F. Loulergue. A Functional Language for Departmental Metacomputing. *Parallel Processing Letters*, 15(3) :(289–304), 2005.

[R8] F. Gava et F. Loulergue, A Static Analysis for Bulk Synchronous Parallel ML to Avoid Parallel Nesting. *Future Generation Computer Systems*, 21(5) :(665–671), 2005. Version révisée de [C23].

[R9] F. Loulergue, F. Gava, M. Arapinis et F. Dabrowski, Semantics and Implementation of Minimally Synchronous Parallel ML. *International Journal of Computer and Information Science*, ACIS, 5(3) : (182–199), 2004. Version révisée et étendue de [C20].

[R10] F. Gava. Formal Proofs of Functional BSP Programs. *Parallel Processing Letters*, 13(3) :(365–376), 2003.

Conférences internationales (avec comité de sélection)

[C1] F. Gava, J. Fortin et M. Guedj. Deductive verification of state-space algorithms. Dans Einar Broch Johnsen and Luigia Petre editor *integrated Formal Methods (iFM)*, volume 7940 de LNCS, Springer. pages 124-138. 2013.

[C2] F. Gava, M. Guedj et F. Pommereau. A BSP algorithm for on-the-fly checking CTL* formulas on security protocols. Dans Hong Shen, Yingpeng Sang, Yidong Li, Depei Qian and Albert Y. Zomaya editor, *Parallel and Distributed Computing (PDCAT)*. pages 79-84. IEEE. 2012. **Best paper award**.

[C3] F. Gava, A. Hidalgo et J. Fortin. Mechanised verification of distributed state-space algorithms for security protocols. Dans Hong Shen, Yingpeng Sang, Yidong Li, Depei Qian and Albert Y. Zomaya editor, *Parallel and Distributed Computing (PDCAT)*. pages 311-316. IEEE. 2012.

[C4] F. Gava, M. Guedj et F. Pommereau. A BSP algorithm for on-the-fly checking LTL formulas on security protocols. Dans *International Symposium on Parallel and Distributed Computing (ISPDC)*. In M. Bader *et al.* editors. pages 11—18, IEEE. 2012.

[C5] F. Gava, C. Li et G. Hains : Implementation of data-parallel skeletons : a case study using a coarsed-grained hierarchical model. Dans *International Symposium on Parallel and Distributed Computing (ISPDC)*. In M. Bader *et al.* editors. pages 26—33. IEEE, 2012.

[C6] F. Loulergue, F. Gava, N. Kosmatov et M. Lemerre. Towards Verified Cloud Computing Environments. In *International Conference on High Performance Computing and Simulation (HPCS)*. In W. W. Smari and V. Zeljkovic editors. pages 91—97. IEEE, 2012.

[C7] F. Gava, M. Guedj et F. Pommereau. Performance Evaluations of a BSP Algorithm for State Space Construction of Security Protocols. Dans *Parallel, Distributed, and Network-Based Processing (PDP)* 2012. pages 170-174. 2012.

[C8] S. Tan et F. Gava. Modular Implementation of Dense Matrix Operations in a High-level BSP Language. Dans *International Conference on High Performance Computing & Simulation (HPCS)*, pages 643–649. 2010. IEEE.

[C9] J. Fortin et F. Gava. From BSP Routines to High-performance ones : Formal Verification of a Transformation Case. Dans *International Conference on Computational Science (ICCS)*. Procedia CS 1(1). pages 155–164, 2010.

[C10] F. Gava et J. Fortin. Two Formal Semantics of a Subset of the Paderborn University BSPlib. Dans D. El Baz, F. Spies, et T. Gross, editeurs, *Parallel, Distributed, and Network-Based Processing (PDP)*, numéro P3544, pages 44–51, 2009. IEEE Computer Society.

[C11] F. Gava et Jean Fortin. Formal Semantics of a Subset of the Paderborn's BSPlib. *Parallel and Distributed Computing (PDCAT)* pages 269-276. IEEE Computer Society. 2008.

[C12] F. Gava BSP Functional Programming ; Examples of a cost based methodology. Dans M. Bubak, G. D. van Albada, J. Dongarra, et P. M. A. Sloot, editeurs, *The International Conference on Computational Science (ICCS)*, Part I, volume 5101 de LNCS, Springer-Verlag. pages 375-385, 2008.

[C13] F. Gava Implementation of the Parallel Superposition in Bulk-Synchronous Parallel ML. In Y. Shi, G.D.v. Albada, J. Dongarra and P.M.A. Sloot, editors, *The International Conference on Computational Science (ICCS 2007), Part IV*, number 4487 in LNCS, pages 611–619. Springer Verlag, 2007.

[C14] L. Gesbert, F. Gava, F. Loulergue and F. Dabrowski Bulk Synchronous Parallel ML with Exceptions. In P. Kacsuk, T. Fahringer and Z. Nemeth, editors, Distributed and Parallel Systems (DAPSYS 2006), 33–42, LNCS, 2006.

[C15] F. Loulergue, R. Benheddi, F. Gava and D. Louis-Resgis In D. Grigoriev, J. Harrison and E.A. Hirsch, editors, Bulk-Synchronous Parallel ML : Semantics and Implementation of the Parallel Juxtaposition. *International Computer Science Symposium in Russia (CSR 2006)*, 475–486, number 3967 inLNCS 2006.

[C16] F. Loulergue, F. Gava et D. Billiet Bulk-Synchronous Parallel ML : Modular Implementation and Performance Prediction. In V. S. Sunderam and G. Dick van Albada and P. M. A. Sloot and J. Dongarra, editors, *The International Conference on Computational Science (ICCS 2005), Part IV*, LNCS, pages 1046–1054. Springer Verlag, 2005.

[C17] F. Gava. Parallel I/O in Bulk Synchronous Parallel ML. In M. Bubak, D. van Albada, P. Sloot, and J. Dongarra, editors, *The International Conference on Computational Science (ICCS 2004), Part III*, LNCS, pages 339–346. Springer Verlag, 2004.

[C18] F. Gava. Design of Deparmental Metacomputing ML. In M. Bubak, D. van Albada, P. Sloot, and J. Dongarra, editors, *The International Conference on Computational Science (ICCS 2004)*, LNCS, pages 50–53. Springer Verlag, 2004.

[C19] F. Gava et F. Loulergue. Semantics of a Functional Bulk Synchronous Parallel Language with Imperative Features. In G. Joubert, W. Nagel, F. Peters, and W. Walter, editors, *Parallel Computing : Software Technology, Algorithms, Architectures and Applications, Proceeding of the 10th ParCo Conference*, Dresden, 2003. North Holland/Elsevier, 2004.

[C20] F. Gava, F. Loulergue et F. Dabrowski. A Parallel Categorical Abstract Machine for Bulk Synchronous Parallel ML. In W. Dosch and R. Y. Lee, editors, *4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, pages 293–300. ACIS, 2003.

[C21] F. Dabrowski, F. Loulergue et F. Gava. Pattern Matching of Parallel Values in Bulk Synchronous Parallel ML. In W. Dosch and R. Y. Lee, editors, *4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, pages 301–308. ACIS, 2003.

[C22] M. Arapinis, F. Loulergue, F. Gava et F. Dabrowski. Semantics of Minimally Synchronous Parallel ML. In W. Dosch and R. Y. Lee, editors, *4th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, pages 260–267. ACIS, 2003.

[C23] F. Gava et F. Loulergue. A Polymorphic Type System for Bulk Synchronous Parallel ML. In V. Malyshkin, editor, *Seventh International Conference on Parallel Computing Technologies (PaCT 2003)*, number 2763 in LNCS, pages 215–229. Springer Verlag, 2003.

[C24] F. Gava et F. Loulergue. A Parallel Virtual Machine for Bulk Synchronous Parallel ML. In Peter M. A. Sloot and al., editors, *International Conference on Computational Science (ICCS 2003), Part I*, number 2657 in LNCS, pages 155–164. Springer Verlag, 2003.

Workshops internationaux (avec comité de sélection)

[W1] F. Gava , L. Gesbert, et F. Loulergue. Type System for a Safe Execution of Parallel Programs in BSML. Dans 5th *ACM SIGPLAN workshop on High-Level Parallel Programming and Applications (HLPP)*, 2011. ACM. 10 pages.

[W2] F. Gava, M. Guedj et F. Pommereau. A BSP algorithm for the state space construction of security protocols. Dans *9th International Workshop on Parallel and Distributed Methods in verifiCation (PDMC, affiliated to conference SPIN 2010)*, 2010. IEEE Computer Society. 8 pages.

[W3] W. Bousdira, F. Gava, L. Gesbert, F. Loulergue, et G. Petiot. Functional Parallel Programming with Revised Bulk Synchronous Parallel ML. Dans Koji Nakano, editeur, *2nd International Workshop on Parallel and Distributed Algorithms and Applications (PDAA)*, 2010. IEEE Computer Society. 6 pages.

[W4] J. Fortin et F. Gava. BSP-Why : an intermediate language for deductive verification of BSP programs. *Dans 4th International Workshop on High-level Parallel Programming and Applications (HLPP 2010, affiliated to conference ICFP 2010)*, 2010. ACM Press. 10 pages.

[W5] F. Gava et I. Garnier. New Implementation of a BSP Composition Primitive with Application to the Implementation of Algorithmic Skeletons. Dans *Workshop APDCM, part of IPDPS*, pages 1–8, 2009. IEEE Computer Society.

Revues nationales (avec comité de rédaction)

[A1] F. Gava. Une bibliothèque certifiée de programmes fonctionnels BSP. *Techniques et sciences informatiques*, 25(10) :(1261–1280), 2007. Version étendue et révisée de [N4].

Conférences nationales (avec comité de sélection)

[N1] F. Gava et S. Tan. Implementation et prediction des performances de squelettes data-paralleles en utilisant un langage BSP de haut-niveau. Dans S. Conchon et A. Mahboudi, éditeurs, *Journees Francophones des Langages Applicatifs (JFLA)*, Studia Informatica Universalis, pages 39–65, 2011. Hermann.

[N2] L. Gesbert, F. Gava, F. Loulergue and F. Dabrowski Bulk Synchronous Parallel ML avec exceptions. Rencontres Francophones du Parallélisme (Renpar'17), 2006.

[N3] F. Gava. Une implantation de la juxtaposition. In T. Hardin, editor, *Journées Francophones des Langages Applicatifs (JFLA 2006)*, pages 87–100, INRIA, january 2006.

[N4] F. Gava. Une bibliothèque certifiée de programmes fonctionnels BSP. In V. Ménissier Morain, editor, *Journées Francophones des Langages Applicatifs (JFLA 2004)*, pages 55–68, INRIA, january 2004.

[N5] F. Gava et F. Loulergue. Synthèse de types pour Bulk Synchronous Parallel ML. In J.-C. Filliâtre, editor, *Journées Francophones des Langages Applicatifs (JFLA 2003)*, pages 153–168, INRIA, january 2003.

Mémoires et thèses

[M1] F. Gava. BSP, bon à toutes les sauces ; Application to Functional Programming, Mechanised Verification and Security Protocols. Habilitation thesis, Université de Paris-east, 2012.

[M2] F. Gava. Approches fonctionnelles de la programmation parallèle et des méta-ordinateurs ; Sémantiques, implantations et certification. Phd thesis, Université de Paris XII–Val-de-Marne, 2005.

[M3] F. Gava. Un système de type polymorphe pour le langage BSML avec traits impératifs. Master's thesis, Université de Paris XII–Val-de-Marne, 2002.

6 Références

[1] M. Aldinucci and M. Danelutto. Stream parallel skeleton optimization. In *Proc. of the 11th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS99)*, pages 955–962, 1999.

[2] M. Alt. *Using Algorithmic Skeletons for Efficient Grid Computing with Predictable Performance*. PhD thesis, Universität Münster, 2007.

[3] Todd R. Andel, G. Back, and Alec Yasinsac. Automating the security analysis process of secure ad hoc routing protocols. *Simulation Modelling Practice and Theory*, 19(9) :2032–2049, 2011.

[4] A. Armando, R. Carbone, and L. Compagna. LTL Model Checking for Security Protocols. *Applied Non-Classical Logics*, 19(4) :403–429, 2009.

[5] Alessandro Armando, Roberto Carbone, and Luca Compagna. Ltl model checking for security protocols. *Journal of Applied Non-Classical Logics*, 19(4) :403–429, 2009.

[6] M. Armbrust, A. Fox, R. Griffith, and al. Above the Clouds : A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[7] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The Landscape of Parallel Computing Research : A View from Berkeley. Technical Report UCB/EECS-2006-183, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2006.

[8] Robert Atkey. Amortised resource analysis with separation logic. *Logical Methods in Computer Science*, 7(2), 2011.

[9] Nicolas Ayache, Roberto Amadio, and Yann Régis-Gianas. Certifying and reasoning on cost annotations in C programs. In *International Workshop on Formal Methods for Industrial Critical Systems (FMICS)*, 2012.

[10] Michael Backes and Dominique Unruh. Limits of constructive security proofs. In Josef Pieprzyk, editor, *Theory and Application of Cryptology and Information Security (ASIACRYPT)*, volume 5350 of *LNCS*, pages 290–307. Springer, 2008.

[11] Pavan Balaji, Darius Buntinas, David Goodell, William Gropp, Torsten Hoefler, Sameer Kumar, Ewing L. Lusk, Rajeev Thakur, and Jesper Larsson Träff. MPI on Millions of Cores. *Parallel Processing Letters*, 21(1) :45–60, 2011.

[12] J. Barnat, L. Brim, V. Havel, J. Havlicek, J. Kriho, M. Lenco, P. Rockai, V. Still, and J. Weiser. DiVinE 3.0 – An Explicit-State Model Checker for Multithreaded C & C++ Programs. In *Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 863–868. Springer, 2013.

[13] J. Barnat, J. Chaloupka, and J. Van De Pol. Distributed Algorithms for SCC Decomposition. *Journal of Logic and Computation*, 21(1) :23–44, 2011.

[14] David A. Basin, Sebastian Mödersheim, and Luca Viganò. An On-the-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors, *European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *LNCS*, pages 253–270. Springer, 2003.

[15] A. Benoit, M. Cole, S. Gilmore, and J. Hillston. Evaluating the performances of skeleton-based high level parallel programs. *Scalable Computing : Practice and Experience*, 6(4) :71–90, 2005.

[16] R. Benzinger. Automated higher-order complexity analysis. *Theoretical Computer Science*, 318 :79–103, 2004.

[17] Karthikeyan Bhargavan, Ricardo Corin, Pierre-Malo Deniélou, Cédric Fournet, and James J. Leifer. Cryptographic protocol synthesis and verification for multiparty sessions. In *omputer Security Foundations Symposium (CSF)*, pages 124–140. IEEE Computer Society, 2009.

[18] G. Bhat, R. Cleaveland, and O. Grumberg. Efficient On-the-Fly Model Checking for CTL*. In *Logic in Computer Science (LICS)*, pages 388–398. IEEE Computer Society, 1995.

[19] R.S. Bird. Algebraic identities for program calculation. *The Computer Journal*, 32(2) :122–126, February 1989.

[20] R. H. Bisseling. *Parallel Scientific Computation. A structured approach using* BSP *and* MPI. Oxford University Press, 2004.

[21] Sandrine Blazy and Xavier Leroy. Mechanized Semantics for the Clight Subset of the C Language. *Journal of Automated Reasoning*, 43(3) :263–288, 2009.

[22] François Bobot, Jean-Christophe Filliâtre, Claude Marché, and Andrei Paskevich. Why3 : Shepherd Your Herd of Provers. In *International Workshop on Intermediate Verification Languages (Boogie)*, 2011.

[23] Wadoud Bousdira, Louis Gesbert, and Frédéric Loulergue. Syntaxe et sémantique de Revised Bulk Synchronous Parallel ML. In S. Conchon and A. Mahboubi, editors, *Journées Francophones des Langages Applicatifs (JFLA)*, Studia Informatica Universalis, pages 117–146. Hermann, 2011.

[24] A. Bouteiller, P. Lemarinier, G. Krawezik, and F. Cappello. Coordinated Checkpoint versus Message log for Fault Tolerant MPI. *Journal of high performance computing and networking (IJHPCN)*, 2005. Inderscience Publisher.

[25] Franck Cappello, Amina Guermouche, and Marc Snir. On Communication Determinism in Parallel HPC Applications. In *Computer Communications and Networks (ICCCN)*, pages 1–8. IEEE, 2010.

[26] A. Chan, F. Dehne, and R. Taylor. Implementing and Testing CGM Graph Algorithms on PC Clusters and Shared Memory Machines. *Journal of High Performance Computing Applications*, 2005.

[27] G. Chang, C. B. Roth, C. L. Reyes, O. Pornillos, Y.-J. Chen, and A. P. Chen. Retraction. *Science*, 314(5807) :1875, 2006.

[28] Sudipta Chattopadhyay, Chong Lee Kee, Abhik Roychoudhury, Timon Kelter, Peter Marwedel, and Heiko Falk. A unified wcet analysis framework for multi-core platforms. In Marco Di Natale, editor, *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 99–108. IEEE, 2012.

[29] Gianfranco Ciardo, Yang Zhao, and Xiaoqing Jin. Parallel symbolic state-space exploration is difficult, but what is the alternative ? In Lubos Brim and Jaco van de Pol, editors, *Parallel and Distributed Methods in verifiCation (PDMC)*, volume 14 of *EPTCS*, pages 1–17, 2009.

[30] M. Cole and Y. Hayashi. Static Performance Prediction of Skeletal Programs. *Parallel Algorithms and Applications*, 17(1) :59–84, 2002.

[31] H. Comon-Lundh and V. Cortier. How to Prove Security of Communication Protocols ? A Discussion on the Soundness of Formal Models w.r.t. Computational Ones. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 29–44, 2011.

[32] J. Dean and S. Ghemawat. MapReduce : a Flexible Data Processing Tool. *Commun. ACM*, 53(1) :72–77, 2010.

[33] F. Dehne, W. Dittrich, and D. Hutchinson. Efficient external memory algorithms by simulating coarse-grained parallel algorithms. *Algorithmica*, 36 :97–122, 2003.

[34] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols : A taster. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter Y. A. Ryan, Josh Benaloh, Miroslaw Kutylowski, and Ben Adida, editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *LNCS*, pages 289–309. Springer, 2010.

[35] J. Esparza, P. Lammich, R. Neumann, T. Nipkow, A. Schimpf, and J.-G. Smaus. A Fully Verified Executable LTL Model Checker. In *Computer Aided Verification (CAV)*, 2013. to appear.

[36] J. Falcou, K. Hamidouche, and D. Etiemble. Hybrid Bulk Synchronous Parallelism Library for Clustered SMP Architectures. In *4th workshop on High-Level Parallel Programming and Applications (HLPP)*, pages 55–62. ACM, 2010.

[37] J. Falcou, J. Sérot, T. Chateau, and J.-T. Lapresté. Quaff : Efficient C++ Design for Parallel Skeletons. *Parallel Computing*, 32 :604–615, 2006.

[38] P. Ferragina and F. Luccio. String Search in Coarse-Grained Parallel Computers. *Algorithmica*, 24(3) :177–194, 1999.

[39] J.-C. Filliâtre and C. Marché. The WHY/Krakatoa/Caduceus Platform for Deductive Program Verification. In W. Damm and H. Hermanns, editors, *Computer Aided Verification (CAV)*, LNCS. Springer, 2007.

[40] J.-C. Filliâtre and C. Marché. The WHY/Krakatoa/Caduceus Platform for Deductive Program Verification. In W. Damm and H. Hermanns, editors, *Computer Aided Verification (CAV)*, LNCS. Springer-Verlag, 2007.

[41] Jean-Christophe Filliâtre. Verifying Two Lines of C with Why3 : an Exercise in Program Verification. In *Verified Software : Theories, Tools and Experiments (VSTTE)*, 2012.

[42] H. Garavel, R. Mateescu, and I. M. Smarandache. Parallel State Space Construction for Model-Checking. In M. B. Dwyer, editor, *Proceedings of SPIN*, volume 2057 of *LNCS*, pages 217–234. Springer, 2001.

[43] A. V. Gerbessiotis. Algorithmic and practical considerations for dense matrix computations on the bsp model. Technical Report PRG-TR-32-97, The University of Oxford, 1997.

[44] A. V. Gerbessiotis, C. J. Siniolakis, and A. Tiskin. Parallel priority queue and list contraction : The BSP approach. *Computing and Informatics*, 21 :59–90, 2002.

[45] L. Gesbert. *Développement systématique et sûreté d'exécution en programmation parallèle structurée*. PhD thesis, University Paris Est, LACL, 2009.

[46] Louis Gesbert, Zhenjiang Hu, Frédéric Loulergue, Kiminori Matsuzaki, and Julien Tesson. Systematic Development of Correct Bulk Synchronous Parallel Programs. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pages 334–340. IEEE, 2010.

[47] Horacio González-Vélez and Mario Leyton. A Survey of Algorithmic Skeleton Frameworks : High-level Structured Parallel Programming Enablers. *Software, Practrice & Experience*, 40(12) :1135–1160, 2010.

[48] Ganesh Gopalakrishnan, Robert M. Kirby, Stephen F. Siegel, Rajeev Thakur, William Gropp, Ewing L. Lusk, Bronis R. de Supinski, Martin Schulz, and Greg Bronevetsky. Formal Analysis of MPI-based Parallel Programs : Present and Future. *Commun. ACM*, 54(12) :82–91, 2011.

[49] S. Gorlatch. SAT : A Programming Methodology with Skeletons and Collective Operations. In F. A. Rabhi and S. Gorlatch, editors, *Patterns and Skeletons for Parallel and Distributed Computing*, pages 29–64. Springer, 2003.

[50] Sergei Gorlatch. Send-receive Considered Harmful : Myths and Realities of Message Passing. *ACM TOPLAS*, 26(1) :47–56, 2004.

[51] Michael Guedj. *BSP Algorithms for LTL & CTL* Model Checking of Security Protocols*. PhD thesis, University of Paris-Est, 2012.

[52] Gaétan Hains and Muath Alrammal. Performance and scalability of xml query processing. In Leonard Barolli, Fatos Xhafa, Salvatore Vitabile, and Minoru Uehara, editors, *Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 841–846. IEEE, 2012.

[53] B. Hendrickson. Computational Science : Emerging Opportunities and Challenges. *Journal of Physics : Conference Series*, 180(1), 2009.

[54] J. M. D. Hill, S. R. Donaldson, and T. Lanfear. Process migration and fault tolerance of BSPlib programs running on networks of workstations. In *Euro-Par'98*, 1998.

[55] Konrad Hinsen, Hans Petter Langtangen, Ola Skavhaug, and Åsmund Ødegård. Using BSP and PYTHON to Simplify Parallel Programming. *Future Generation Comp. Syst.*, 22(1-2) :123–157, 2006.

[56] C. A. R. Hoare, Jayadev Misra, Gary T. Leavens, and Natarajan Shankar. The Verified Software Initiative : A Manifesto. *ACM Comput. Surv.*, 41(4), 2009.

[57] A. Hobor and C. Gherghina. Barriers in Concurrent Separation Logic : Now With Tool Support ! *Logical Methods in Computer Science*, 8(2), 2012.

[58] Cliff B. Jones, Peter W. O'Hearn, and Jim Woodcock. Verified Software : A Grand Challenge. *IEEE Computer*, 39(4) :93–95, 2006.

[59] J. Kim and K. Yi. Interconnecting between CPS terms and non-CPS terms. In A. Sabry, editor, *SIGPLAN Workshop on Continuations*. ACM, 2001.

[60] Raimund Kirner, Jens Knoop, Adrian Prantl, Markus Schordan, and Albrecht Kadlec. Beyond loop bounds : comparing annotation languages for worst-case execution time analysis. *Software and System Modeling*, 10(3) :411–437, 2011.

[61] R. Kumar and E. G. Mercer. Load balancing parallel explicit state model checking. In *ENTCS*, volume 128, pages 19–34. Elsevier, 2005.

[62] E. A. Lee. The Problem with Threads. Technical Report UCB/EECS-2006-1, Electrical Engineering and Computer Sciences University of California at Berkeley, 2006.

[63] Chong Li and Gaétan Hains. SGL : Towards a Bridging Model for Heterogeneous Hierarchical Platforms. *IJHPCN*, 7(2) :139–151, 2012.

[64] Claude Marché, Christine Paulin-Mohring, and Xavier Urbain. The KRAKATOA tool for certificationof Java/JavaCard programs annotated in JML. *J. Log. Algebr. Program.*, 58(1-2) :89–106, 2004.

[65] Catherine Meadows. Formal methods for cryptographic protocol analysis : emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1) :44–54, 2003.

[66] Z. Merali. Computational science : Error, why scientific programming does not compute. *Nature*, 467(7317) :775–777, 2010.

[67] Q. Miller. BSP in a Lazy Functional Context. In *Trends in Functional Programming*, volume 3. Intellect Books, 2002.

[68] Kedar S. Namjoshi. Certifying Model Checkers. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *Computer Aided Verification (CAV)*, volume 2102 of *LNCS*, pages 2–13. Springer, 2001.

[69] Kosuke Ono, Yoichi Hirai, Yoshinori Tanabe, Natsuko Noda, and Masami Hagiya. Using coq in specification and program extraction of hadoop mapreduce applications. In Gilles Barthe, Alberto Pardo, and Gerardo Schneider, editors, *Software Engineering and Formal Methods (SEFM)*, volume 7041 of *LNCS*, pages 350–365. Springer, 2011.

[70] R. Patel, B. Borisaniya, A. Patel, D. Patel, M. Rajarajan, and A. Zisman. Comparative analysis of formal model checking tools for security protocol verification. In *CNSA*, volume 89 of *Communications in Computer and Information Science*, pages 152–163. Springer, 2010.

[71] Christine Rochange, Armelle Bonenfant, Pascal Sainrat, Mike Gerdes, Julian Wolf, Theo Ungerer, Zlatko Petrov, and Frantisek Mikulu. Wcet analysis of a parallel 3d multigrid solver executed on the merasa multi-core. In Björn Lisper, editor, *Worst-Case Execution Time Analysis (WCET)*, volume 15 of *OASICS*, pages 90–100. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2010.

[72] S. Sanjabi and F. Pommereau. Modelling, Verification, and Formal Analysis of Security Properties in a P2P System. In *Collaboration and Security (COLSEC'10)*, pages 543–548. IEEE, 2010.

[73] Natarajan Shankar. Trust and Automation in Verification Tools. In Sung Deok Cha, Jin-Young Choi, Moonzoo Kim, Insup Lee, and Mahesh Viswanathan, editors, *Automated Technology for Verification and Analysis (ATVA)*, volume 5311 of *LNCS*, pages 4–17. Springer, 2008.

[74] O. Shivers. Continuations and threads : Expressing machine concurrency directly in advanced languages. In *Second ACM SIGPLAN Workshop on Continuations*, 1997.

[75] D. B. Skillicorn, J. M. D. Hill, and W. F. McColl. Questions and Answers about BSP. *Scientific Programming*, 6(3) :249–274, 1997.

[76] A. Stewart. A Programming Model for BSP with Partitioned Synchronisation. *Formal Asp. Comput.*, 23(4) :421–432, 2011.

[77] Jun Sun, Yang Liu, and Bin Cheng. Model Checking a Model Checker : A Code Contract Combined Approach. In Jin Song Dong and Huibiao Zhu, editors, *Formal Engineering Methods (ICFEM)*, volume 6447 of *LNCS*, pages 518–533. Springer, 2010.

[78] Julien Tesson and Frédéric Loulergue. A Verified Bulk Synchronous Parallel ML Heat Diffusion Simulation. In *International Conference on Computational Science (ICCS)*, Procedia Computer Science, pages 36–45. Elsevier, 2011.

[79] A. Tiskin. A New Way to Divide and Conquer. *Parallel Processing Letters*, (4), 2001.

[80] Ming-Hsien Tsai and Bow-Yaw Wang. Formalization of CTL* in the Calculus of Inductive Constructions. In Mitsu Okada and Ichiro Satoh, editors, *Asian computing science conference on Advances in computer science : secure software and related issues (ASIAN)*, volume 4435 of *LNCS*, pages 316–330. Springer-Verlag, 2007.

[81] E. Turner, M. Butler, and M. Leuschel. A Refinement-based Correctness Proof of Symmetry Reduced Model-checking. In *Abstract State Machines, Alloy, B and Z*, LNCS, pages 231–244. Springer, 2010.

[82] L. G. Valiant. A bridging model for multi-core computing. In *European Symposium on Algorithms (ESA)*, pages 13–28. Springer-Verlag, 2008.

[83] E. van der Weegen and J. McKinna. A Machine-checked Proof of the Average-case Complexity of Quicksort in Coq. In Stefano Berardi, Ferruccio Damiani, and Ugo de'Liguoro, editors, *Types for Proofs and Programs, International Conference (TYPES 2008)*, LNCS 5497, pages 256–271. Springer, 2008.

[84] P. B. Vasconcelos and K. Hammond. Inferring Cost Equations for Recursive, Polymorphic and Higher-Order Functional Programs. In *IFL'02*, LNCS, pages 110–125. Springer Verlag, 2003.

[85] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David B. Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter P. Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.

[86] Andrew K. Wright and Matthias Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115 :38–94, 1994.

[87] Wolf Zimrnermann. Automatic worst case complexity. analysis of parallel programs. Technical Report TR-90-U66, Berkeley university (ICSI), 1990.