# DOSSIER DE CANDIDATURE

Christian Rinderknecht

## Contact

**Christian Rinderknecht**
4 Rue de Maguelone
Montpellier, France

http://crinderknecht.free.fr
06.14.99.23.55
rinderknecht@free.fr

## Table des matières

1 Présentation générale . . . . . 1
2 Curriculum vitæ . . . . . 2
3 Publications et manuscrits . . . . . 4
4 Activités d'enseignement . . . . . 6
5 Responsabilités administratives . . . . . 8
6 Activités de recherche . . . . . 9
7 Résumé des travaux de recherche . . . . . 10
8 Expérience industrielle . . . . . 14
9 Projets de recherche . . . . . 15

## Mots clés

Langages de données, langages de programmation, programmation fonctionnelle, compilateurs, analyse statique, génie logiciel, preuves formelles, méthodes formelles, test, protocoles de télécommunication.

# 1 Présentation générale

J'ai soutenu ma thèse de doctorat en informatique à l'université Pierre et Marie Curie (Paris 6) en décembre *1998*. Ma recherche doctorale a été effectuée à l'INRIA, site de Rocquencourt, au sein du projet Cristal (maintenant Gallium), spécialisé dans la conception, la formalisation et la réalisation de langages de programmation et de systèmes, en particulier le langage fonctionnel OCaml.

Dès cette période, la particularité de ma recherche a été de *résoudre des problèmes industriels en appliquant des méthodes formelles* et en réalisant des *logiciels sûrs* avec des utilisateurs externes aux laboratoires. J'ai déjà entamé une carrière internationale qui alterne université et industrie, en France, en Corée et en Hongrie.

## 1.1 Recherche

Le fil d'Ariane de ma recherche est le langage formel, au sens général. Si un problème peut s'exprimer formellement, nous avons alors un point d'appui pour parvenir à sa solution, tant théorique que pratique. Au gré des opportunités, j'ai déroulé ce fil conducteur autour de plusieurs domaines : les langages de données, le test des protocoles de télécommunications, la compilation (surtout les frontaux et l'analyse statique) et la didactique des langages formels (programmation et logique).

Dans aucun contexte d'accueil je n'ai soutenu de dichotomie entre théorie et pratique, mathématiques et informatique, académie et industrie, ou enseignement et recherche. Ainsi, dans un laboratoire de télécommunications, j'ai apporté une expertise en compilation et programmation fonctionnelle ; dans un laboratoire d'informatique théorique, j'ai encouragé des préoccupations industrielles, sans transiger avec les normes internationales ; dans un laboratoire des images de synthèse, j'ai suscité un intérêt pour les applications à la didactique de la programmation.

## 1.2 Enseignement

J'ai commencé à enseigner en *1993*, alors que j'étais moniteur à l'université Denis Diderot (Paris 7). Hormi les quelques années passées dans l'industrie, j'ai depuis presque toujours enseigné l'informatique, et même un peu les mathématiques. Les étudiants étaient principalement en licence (selon la nomenclature actuelle) ou en école d'ingénieurs, plus rarement en master, car j'ai un intérêt tout particulier pour les débutants, et j'ai même publié et proposé des articles didactiques, ainsi qu'un livre pour les élèves. Les sujets que j'ai enseigné peuvent être classés ainsi : génie logiciel, langages de programmation, traitement des données semi-structurées, informatique théorique.

## 1.3 Faits divers

Je suis bilingue français-espagnol et je parle couramment l'anglais. J'ai ponctuellement enseigné en espagnol, j'ai également dirigé dans cette langue toute la recherche d'un étudiant colombien en master. De *2005* à *2012*, à l'université Konkuk à Séoul, et de *2013* à *2014* à l'université Eötvös Lórand de Budapest, j'ai enseigné en anglais, ma langue de travail.

Je suis le traducteur du recueil *Vingt poèmes d'amour et une chanson désespérée* de Pablo Neruda, éditions Gallimard, collection Poésie, Paris, seconde édition, *1998*.

J'ai reçu un chèque de Donald Knuth pour avoir trouvé une erreur dans son tome 4 de *The Art of Computer Programming*.

# 2 Curriculum vitæ

**Christian Rinderknecht**
4 Rue de Maguelone
Montpellier, France

http://crinderknecht.free.fr
06.14.99.23.55
rinderknecht@free.fr

## 2.1 Carrière

***2014-*** Cortus SAS (Montpellier, France)
*Ingénieur R&D*

***2013-2014*** Eötvös Lórand University (Budapest, Hongrie)
*Chercheur Invité* (Department of Programming Languages and Compilers)

***2005-2012*** Konkuk University (Séoul, République de Corée)
*Assistant Professor* (Department of Internet and Multimedia Engineering)

***2003-2004*** École Supérieure d'Ingénieurs Léonard de Vinci (Courbevoie, France)
Professeur assistant (Département Génie Logiciel)

***2001-2002*** KAIST (Daejeon, République de Corée)
*Visiting Professor* (Network Architecture Laboratory)

***2000-2001*** MathWorks (Montbonnot, France)
*Ingénieur d'étude*

***1998-2000*** Télécom SudParis (Évry, France)
*Ingénieur de recherche* (Groupe Logiciels et Réseaux)

***1997-1998*** Alcatel-Lucent (Marcoussis, France)
*Ingénieur d'étude* (Object Architecture Unit)

## 2.2 Formation doctorale

***1993-1998*** Études doctorales à l'INRIA (Rocquencourt) sous la direction de Bernard LORHO et Michel MAUNY.

**Sujet** Une formalisation d'ASN.1, Application d'une méthode formelle à un langage de spécification télécom.

**Spécialité** Programmation, sémantique, preuves et langages.

**Soutenance** Université Pierre et Marie Curie (Paris 6), le 2 décembre 1998. Mention très honorable, avec félicitations.

**Jury** Paul GLOESS (rapporteur), Jacques GUYARD (rapporteur), Bernard LORHO (directeur), Michel MAUNY (directeur), Olivier DUBUISSON (examinateur), Thérèse HARDIN (présidente) et Hélène BACHATÈNE (examinateur).

***1992-1993*** Diplôme d'Études Approfondies d'informatique théorique, calcul et programmation (DEA ITCP), de l'université Pierre et Marie Curie (Paris 6) ; mention bien.

**Sujet du mémoire** Portage du système de métacompilation SAM (*Semantic Abstract Machine*).

**Directeur** Pierre WEIS (INRIA Rocquencourt)

## 2.3 Études prédoctorales

***1991-1992*** Maîtrise d'informatique de l'université Pierre et Marie Curie (Paris 6) ; mention très bien.

***1990-1991*** Licence d'informatique de l'université Pierre et Marie Curie (Paris 6) ; mention bien.

***1989-1990*** Diplôme d'Études Universitaires Générales (DEUG) en physique, mathématiques, option probabilités, de l'université Pierre et Marie Curie (Paris 6) ; mention bien.

# 3 Publications et manuscrits

## 3.1 Articles publiés dans des revues avec comité de lecture

1. Nachum DERSHOWITZ et Christian RINDERKNECHT : *The Average Height of Catalan Trees by Counting Lattice Paths*. Mathematics Magazine, *2015*.
2. Christian RINDERKNECHT : *A Survey on Teaching and Learning Recursive Programming*. Informatics in Education, 13(1):87–119, avril *2014*.
3. Christian RINDERKNECHT: *A Didactic Analysis of Merge Sort*. Teaching Mathematics and Computer Science, 11(2):195–210, octobre *2013*.
4. Christian RINDERKNECHT: *A Didactic Analysis of Functional Queues*. Informatics in Education, 10(1):65–72, avril *2011*.
5. Christian RINDERKNECHT et Nic VOLANSCHI: *Theory and Practice of Unparsed Patterns for Metacompilation*. Science of Computer Programming, 75(3):85–105, *2010*.
6. Christian RINDERKNECHT: *An Algorithm for Validating ASN.1 (X.680) Specifications using Set Constraints*. The Computer Journal, 46(4):401–420, juillet *2003*.

## 3.2 Articles présentés dans des conférences avec comité de lecture

7. Juan Diego TASCÓN VIDARTE, Christian RINDERKNECHT, Jee-In KIM et HyungSeok KIM: *A Tangible Interface for Learning Recursion and Functional Programming*. Proceedings of the International Symposium on Ubiquitous Virtual Reality (ISUVR), Gwangju, République de Corée, juillet *2010*.
8. Nic VOLANSCHI et Christian RINDERKNECHT: *Unparsed Patterns: Easy User-extensibility of Program Manipulation Tools*. Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation (PEPM), pages 111–121, San Francisco, USA, janvier *2008*.
9. Christian RINDERKNECHT: *Proving a Soundness Property of the Joint Design of ASN.1 and the Basic Encoding Rules*. Proceedings of the SDL and MSC Workshop on System Analysis and Modeling (SAM), pages 154–170, Ottawa, Canada, juin *2004*.
10. Patrick DUVAL, Agathe MERCERON, Christian RINDERKNECHT et Michel SCHOLL: *LeVinQam: A Question Answering Mining Platform*. Proceedings of the conference on Information Technology-based Higher Education and Training (ITHET), Istanbul, Turquie, juin *2004*.
11. Ana CAVALLI et al.: *PLATONIS: A Platform for Validation and Experimentation of Multi-protocols and Multi-services*. Proceedings of the conference on Applications and Services in Wireless Networks (ASWN), pages 217–229, Évry, France, juillet *2001*.
12. Ana CAVALLI, Bruno DEFUDE, Christian RINDERKNECHT et Fatiha ZAÏDI: *A Service-component Testing Method and a suitable CORBA Architecture*. Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC), pages 655–666, Hammamet, Tunisie, juillet *2001*.
13. Ana CAVALLI, Bruno DEFUDE, Christian RINDERKNECHT et Fatiha ZAÏDI: *Test de composants de service et exécution de tests sur une plate-forme CORBA*. Actes de la Conférence Francophone en Ingénierie des Protocoles (CFIP), pages 363–378, Toulouse, France, octobre *2000*.
14. Ana CAVALLI, David LEE, Christian RINDERKNECHT et Fatiha ZAÏDI: *Hit-or-Jump: Un algorithme pour le test imbriqué avec des applications aux services R.I.* Actes de la troisième édition des Journées Doctorales Informatique et Réseaux (JDIR). Institut National des Télécommunications, Évry, France, novembre *1999*.

15. Ana CAVALLI, David LEE, Christian RINDERKNECHT et Fatiha ZAÏDI: *Hit-or-Jump: An Algorithm for Embedded Testing with Applications to IN Services*. Proceedings of the conference on Formal Methods for Protocol Engineering and Distributed Systems (FORTE), pages 41–56, Pékin, Chine, octobre *1999*.

## 3.3 Articles publiés dans des revues sans comité de lecture

16. Christian RINDERKNECHT: *Matching Pairwise Divergent Paths in XML Streams*. Journal of Industrial Science and Technology, 32:57–75, décembre *2007*.

## 3.4 Monographies

17. Christian RINDERKNECHT: *Design and Analysis of Purely Functional Programs*, volume 15 de *Texts in Computing*. College Publications, Royaume-Uni, troisième édition, 650 pages, janvier *2012*.

## 3.5 Rapports techniques

18. Christian RINDERKNECHT: *Communication Model in Distributed Cyber-Physical Systems*, chapitre *Model-based design and testing*. Eötvös Lórand University, Budapest, Hongrie, 19 pages, juin *2013*.
19. Michel MAUNY et Christian RINDERKNECHT: *Position paper about the ASN.1 Formal Model*, ISO Working Group, juillet *1997*.
20. Christian RINDERKNECHT: *Une analyse syntaxique d'ASN.1:1990 en Caml Light*. Rapport technique 171, INRIA, 228 pages, avril *1995*.

## 3.6 Traductions personnelles

21. Christian RINDERKNECHT: *Conception et analyse des programmes purement fonctionnels*, volume 12 des *Cahiers de logique et d'épistémologie*. College Publications, Royaume-Uni, seconde édition, 528 pages, mai *2012*.
22. Christian RINDERKNECHT: *Parsing ASN.1 with Caml Light*, 188 pages, août *1995*.

# 4 Activités d'enseignement

## 4.1 Modules

Étant donné la grande variété des institutions où j'ai été amené à enseigner, je regrouperai sous l'appellation *module* la conception originale d'un cours, la rédaction en LaTeX de notes détaillées avec des figures en PostScript, la préparation d'exercices et de solutions pour chaque cours et de devoirs à la maison, la mise à disposition des supports sur un site web, la création de programmes et bases de données pour mettre à jour automatiquement ledit site et vérifier sa correction partielle et complétude, l'interaction avec les élèves hors-cours (permanence au bureau et courriels personnalisés), la conduite des travaux dirigés ou pratiques, l'aide à l'installation des logiciels nécessaires à l'étude, la rédaction et la correction des devoirs et examens, l'évaluation de ceux-ci et la saisie des notes sur un système informatisé.

J'ai eu l'entière responsabilité de modules à l'École Supérieure d'Ingénieurs Léonard de Vinci (ESILV) et à l'université Konkuk (Corée). La charge horaire en Corée variait entre 240 et 288 heures de présentiel par an, pour l'ensemble des modules. En France, la charge était similaire à celle d'un maître de conférence. Au total, ces cours se sont étalés sur une durée d'environ 9 ans.

Les sujets étaient *les langages de programmation* (Erlang, OCaml, Prolog, Java) ; *l'introduction aux réseaux* ; *l'informatique théorique* (algorithmes, spécifications algébriques, automates finis, compilation, interprétation des langages fonctionnels, circuits logiques) et *le traitement des données XML* (XSLT).

Durant mon séjour à l'université Eötvös Lórand, à Budapest, j'ai créé et donné un cours de programmation approfondie en Java pour les élèves du master EIT ICT Labs.

## 4.2 Travaux dirigés et pratiques

Dans cette section se trouvent les travaux dirigés et pratiques que j'ai eu a préparer, appartenant à un module dont je n'étais pas en charge. Je suis ainsi intervenu à l'université René Diderot (Paris 7) pendant mon monitorat (trois ans) sur les sujets suivants : algèbre linéaire, calcul matriciel, calcul intégral et programmation en Pascal. À l'Institut Supérieur d'Électronique de Paris (ISEP), j'ai enseigné pendant deux ans la programmation en C++. À l'ESILV, je suis intervenu dans le cadre du module d'*ingénierie des logiciels* (spécification, test, méthodes formelles, shell Unix, outils de développement GNU), pendant un an. À l'Institut National des Télécommunications, j'ai co-dirigé ponctuellement des travaux pratiques sur la programmation en SDL (*Semantics Definition Language*).

## 4.3 Interrogations orales au collège Stanislas

Lors de l'année scolaire 1996-1997, j'ai préparé 25 élèves du collège Stanislas, à Paris, à l'épreuve de programmation en Caml Light nouvellement validée aux concours d'entrée aux grandes écoles d'ingénieurs. À cette occasion, j'ai créé un contenu en accord avec les programmes et dirigé les travaux et devoirs des élèves. Le volume était de deux heures hebdomadaires pendant toute l'année. D'après Henri Lemberg, le professeur de mathématiques qui était mon responsable, cette promotion d'élèves à qui j'ai enseigné la programmation fonctionnelle à établi la plus haute moyenne à l'épreuve correspondante au concours d'entrée aux grandes écoles.

## 4.4 Formations professionnelles

J'ai conçu et conduit une formation d'une semaine sur C++ auprès de professionnels.

# 5 Responsabilités administratives

## 5.1 Programmes d'échanges internationaux

John CAGNOL et moi avons pris l'initiative d'un accord d'échange académique entre nos employeurs d'alors, l'ESILV et l'université Konkuk, qui fut signé à l'été 2008. Cet accord a permis l'échange d'étudiants des deux institutions. La même année, j'ai mis en contact les universités de Reykjavík (Islande) et Konkuk, qui ont conclu le même type d'accord d'échange.

## 5.2 Encadrement de mémoires de master

J'ai entièrement dirigé la recherche de master de Juan Diego TASCÓN VIDARTE (voir publications), et partiellement (six mois) celle d'une élève coréenne de Konkuk (Corée) sur l'extension d'un compilateur pour Java HP avec des aspects (*Aspect Oriented Programming*).

## 5.3 Encadrement de mémoires d'ingénieur

Lors de mon emploi à l'INT (de nos jours Télécom SudParis), j'ai encadré le mémoire d'ingénieur d'une élève sur la voix sur IP. J'ai aussi aidé (un mois) un élève de DEA à programmer en OCaml les algorithmes de son mémoire.

J'ai dirigé les mémoires de trois élèves du master EIT ICT Labs à l'université Eötvös Lórand (Budapest, Hongrie).

## 5.4 Participation à des comités de lecture

De *1998* à *2002*, j'ai contribué, au sein de comités de lectures, à une quinzaine d'évaluation d'articles pour des conférences et des revues.

# 6 Activités de recherche

## 6.1 Participation à des projets nationaux

| | |
|---|---|
| ***1998-2000*** | Projet RNRT CASTOR : Conception d'un atelier de services pour les plates-formes TINA, CORBA et RI, projet en collaboration avec Alcatel, CNET, INT et Verilog. |
| ***2001*** | Projet RNRT PLATONIS : PLAte-forme de validaTIOn et d'expérimentatioN multI-protocoles et multi-Services. |

## 6.2 Participation à des projets internationaux

En juillet *1997*, à Londres, j'ai représenté la France au groupe de travail sur la norme ASN.1 à l'ISO. J'ai participé très activement aux discussions techniques de *1994* à *1998*.

## 6.3 Exposés et séminaires

J'ai participé régulièrement pendant un an à deux séminaires : Test & Objets et AIDA (technologies et didactique). J'ai donné au moins une dizaine d'exposés sur mes travaux de recherche.

# 7 Résumé des travaux de recherche

## 7.1 Analyse statique de langages de données

L'ambition de ma thèse de doctorat était d'appliquer la théorie des langages de programmation au problème de la télécommunication entre environnements hétérogènes (langages de programmation, systèmes d'exploitation, architectures matérielles), plus particulièrement les techniques de compilation pour réaliser un logiciel satisfaisant les plus hauts critères d'emploi industriel (correction, complétude, ergonomie).

Sur une suggestion de France Télécom, j'ai formalisé le langage ASN.1 (*Abstract Syntax Notation One*), employé pour la *spécification de protocoles de télécommunication*, en particulier la télémaintenance d'équipements de réseaux, mais aussi dans les *bases de données*. Il permet une description des types des données susceptibles d'être échangées au cours d'une transaction entre applications distantes. Ce langage est indépendant des architectures locales, matérielles ou logicielles, par conséquent, le partage d'un module ASN.1 apporte une solution symétrique à la possible hétérogénéité des pairs. De plus, ceux-ci font le choix commun d'un type de codage des valeurs (*encoding rules*) dont les types sont définis dans le module ASN.1. Ces codages sont eux aussi indépendants des architectures, et définissent la sérialisation des valeurs ASN.1, c'est-à-dire la transformation de valeurs abstraites en suites de bits. Dans la pratique, chaque pair compile le module en précisant le codage commun, et obtient une paire codeur-décodeur (*codec*) dans un langage de programmation, et celle-ci est liée ensuite comme une bibliothèque au reste de l'application locale, écrite dans le même langage.

La technique formelle que j'ai choisie est d'ordinaire employée pour définir la *sémantique opérationnelle des langages de programmation*, donc elle conduit plus facilement à la réalisation logicielle d'un analyseur de spécifications, que l'on peut considérer comme un frontal de compilateur, surtout avec un langage de programmation fonctionnelle. Cet analyseur a été utilisé par quelques entreprises étrangères pour valider leurs bases de protocoles, et ma recherche aboutit aussi à des résultats théoriques à propos des analyses elles-mêmes, comme la correction de la composition du codage et du décodage BER (*Basic Encoding Rules*), mais aussi à la production d'un analyseur syntaxique prouvé correct et complet par rapport à la norme internationale (Christian RINDERKNECHT : *Une analyse syntaxique d'ASN.1:1990 en Caml Light*). Là encore, le formalisme logique de la sémantique opérationnelle structurée se prête bien aux *preuves inductives et analyses par cas*, et, dualement, permet d'exprimer tout aussi bien des *programmes fonctionnels*, par exemple à l'aide du langage de programmation OCaml. Ainsi sont rapprochés différents aspects du génie logiciel : spécification, preuves et réalisation.

Au-delà de la diffusion du code source de mon analyseur, la recherche en amont a mis au jour de nombreuses zones obscures dans la norme, car l'analyse récursive par cas des constructions permet d'envisager toutes les combinaisons, ce qui est impossible dans une langue naturelle. (Christian RINDERKNECHT : *Une formalisation d'ASN.1 — Application d'une méthode formelle à un langage de spécification télécom*, Thèse de doctorat, Université Pierre et Marie Curie (Paris 6), décembre 1998.)

Le décodage des valeurs ASN.1 est un sujet qui fait souvent la une car il est régulièrement accusé d'être mal conçu. En m'appuyant sur ma recherche doctorale, j'ai prouvé que les BER sont hors de cause et que les problèmes signalés relèvent tous de mauvais codecs produits par les compilateurs ASN.1. Des codes peuvent être alors volontairement mal formés pour réaliser une attaque par débordement de tableaux et mener à des dénis de service ou des plantages du décodeur. La production de codecs corrects est donc cruciale dans le cadre de la *sécurité de fonctionnement*. (Christian RINDERKNECHT : *Proving a Soundness Property of the Joint Design of ASN.1 and the Basic Encoding Rules*.)

Enfin, j'ai montré comment l'analyse de la validité des spécifications ASN.1 repose exclusivement sur les concepts de finitude des valeurs et de *solvabilité de contraintes de sous-typage sémantique*, où

les types sont interprétés comme des ensembles de valeurs. Cette interprétation ensembliste établit un cadre formel plus intuitif car proche du *modèle sémantique* d'ASN.1 défini en anglais par une norme, en accord avec l'ingénierie des protocoles qui considère les valeurs comme seuls objets concrets. Les difficultés techniques proviennent de la richesse du sous-typage, qui inclut récursivité et complémentation (négation), et des mutuelles dépendances syntaxiques entre types, valeurs et contraintes, d'une part, et typage et sous-typage, d'autre part. L'approche de ma thèse consistait à stratifier les concepts pour éviter la circularité et réduire les constructions peu à peu en vérifiant des propriétés dont l'ensemble à la fin permet d'établir ce qui est attendu globalement de la spécification, en particulier vis-à-vis d'un codage qui tient compte du sous-typage, comme les *Packed Encoding Rules*. Une autre démarche consiste à limiter les étapes intermédiaires et à produire le plus rapidement possible des contraintes ensemblistes que l'on soumettrait à un solveur en boîte noire. Malheureusement, ces derniers restreignent le domaine des contraintes ou leurs opérations pour rendre possible des optimisations pour l'analyse statique à grande échelle, mais superflues pour la compilation. D'un autre côté, le bénéfice de la construction de la plus petite solution (point fixe) aux contraintes rend possible l'optimisation de la taille des codes. Par ailleurs, il faudrait retraduire ces solutions sous forme de types pour pouvoir en dériver la meilleure approximation dans le langage cible du codec, peut-être en mâtinant les isomorphismes de types avec du sous-typage sémantique. (Christian RINDERKNECHT : *An Algorithm for Validating ASN.1 (X.680) Specifications using Set Constraints*.)

Ces dernières années, j'ai étendu mon intérêt pour les langages de données à XML, en particulier le langage de requêtes par sélection XPath et le langage de transformations XSLT, qui est plus général et contient XPath. J'ai proposé une interprétation différente de certaines expressions XPath, qui permettrait d'obtenir moins de résultats ou des résultats plus pertinents quand on veut que des types de nœuds dans une requête donnée ne soient pas dans une relation ancêtre-descendant. En d'autres termes, on pourrait sélectionner plus facilement des sous-arbres disjoints. Cela présente notamment un intérêt avec les requêtes de documents mathématiques spécifiés en MathML, où l'inclusion d'arbres est alors interprétée comme la composition de fonctions mathématiques. Par ailleurs, j'ai supposé que l'accès aux données se faisait uniquement via des *flux d'éléments* (*streaming*), qui n'autorisent que la sélection en aval. Par rapport aux flux au sens strict, où seule une quantité de mémoire constante est disponible, j'ai exploré deux dimensions différentes : le premier algorithme que j'ai conçu pour satisfaire ce nouveau type de requête tâche de réduire l'usage de la mémoire en travaillant sur une pile, le second réduit le temps de traitement en opérant sur des tables d'éléments indexés. (Christian RINDERKNECHT : *Matching Pairwise Divergent Paths in XML Streams*).

## 7.2 Test à partir de modèles

Lors de mon doctorat, seule la transmission des données était prise en compte. À l'Institut National des Télécommunications (aujourd'hui Télécom SudParis), j'ai étendu ma démarche au comportement des protocoles spécifiés à l'aide du langage SDL (*Semantics Definition Language*), qui peut être abstraitement compris comme un langage d'automates finis, étendus avec des effets sur des variables, et qui communiquent par échanges asynchrones de messages (un paradigme qu'on trouve aussi avec Erlang, sauf que ce dernier est un langage de programmation fonctionnelle).

Le contexte de ma recherche était les *heuristiques de production automatique de tests de conformité à partir de spécifications formelles.* Le test de conformité consiste à vérifier qu'une réalisation est conforme à un protocole donné, seulement en observant les interactions externes (configuration dite de boîte noire). En d'autres termes, cette section ne décrit pas un axe de recherche totalement différent de la précédente, mais une extension de la composante purement télécommunication de celle-ci, à ceci près que l'accent porte ici plus sur la vérification de modèles que sur la preuve. De plus, ASN.1 est employé pour décrire les données échangées entre automates SDL.

*Hit or jump* est une heuristique pour le test de couverture de sous-composants de processus com-

municants, ce qui est particulièrement difficile à cause de l'explosion combinatoire du nombre d'états lors des simulations. Elle combine et généralise les recherches exhaustives et aléatoires, et produit des séquences de test avec un fort taux de couverture de fautes. J'ai programmé en OCaml un outil qui a détourné le simulateur ObjectGeode, piloté en boîte noire, pour réaliser cette heuristique. Cela a supposé une forte dose de rétro-ingénierie et de techniques élaborées d'analyse syntaxique des traces (au moins sept, non documentées). *Hit or jump* a été appliqué avec succès au test de certains modules de réseaux (téléphoniques) intelligents (Ana CAVALLI, David LEE, Christian RINDERKNECHT et Fatiha ZAÏDI : *Hit-or-Jump: An Algorithm for Embedded Testing with Applications to IN Services*) ainsi que dans le contexte de CORBA (Ana CAVALLI, Bruno DEFUDE, Christian RINDERKNECHT et Fatiha ZAÏDI : *A Service-component Testing Method and a suitable CORBA Architecture*) et du projet national de téléphonie mobile PLATONIS (Ana CAVALLI et al. : *PLATONIS: A Platform for Validation and Experimentation of Multi-protocols and Multi-services*).

Alors que j'étais invité en tant que chercheur à l'université Eötvös Lórand, à Budapest, j'ai contribué un chapitre dans un rapport technique, sur le sujet de la conception et le test à partir de modèles pour les systèmes cyber-physiques distribués, plus connus de nos jours sous l'appellation anglaise *Internet of Things* (Christian RINDERKNECHT : *Communication Model in Distributed Cyber-Physical Systems*, chapitre *Model-based design and testing*).

## 7.3 Compilation

J'entends ici la *métacompilation* comme la production automatique de compilateurs à partir de spécifications abstraites, ou bien comme l'ajout de fonctionnalités à des compilateurs existants. La première acception a constitué le cadre de mon mémoire de DEA, la seconde a fait l'objet d'une étude conjointe avec Nic Volanschi. Cette section ne décrit pas non plus un axe de recherche différent de la première, mais une extension de la composante compilation de celle-ci, à ceci près que l'accent porte ici sur la paramétrisation.

Lorsqu'un compilateur doit être étendu par un utilisateur pour détecter certaines constructions erronées, inefficaces ou d'intérêt particulier, il faut en général modifier l'analyseur syntaxique pour qu'il lise les métavariables des motifs (*patterns*). Ces derniers sont des fragments de programmes avec des trous qui sont nommés par les métavariables. Les motifs sont transformés en arbres de syntaxe abstraite pour filtrer l'arbre de syntaxe abstraite du programme, ce qui conduit les métavariables à être liées à des sous-arbres en cas de succès. Mais toute modification d'un analyseur syntaxique est une tâche très ardue, aussi l'idée de *motifs plats* (*unparsed patterns*), c'est-à-dire qui ne sont pas transformés en arbres, présente un attrait certain, en particulier pour l'intégration dans des compilateurs existants. En contrepartie, la complétude du filtrage est perdue en général et il faut parfois, pour la rétablir, dénoter des sous-structures à l'aide de métaparenthèses, ou typer les métavariables. J'ai simplifié et parachevé l'algorithme de filtrage initialement conçu par Nic Volanschi, j'ai prouvé son équivalence avec une version abstraite non-déterministe, ainsi que sa correction par rapport au filtrage classique à base d'arbres (Christian RINDERKNECHT et Nic VOLANSCHI : *Theory and Practice of Unparsed Patterns for Metacompilation* ; Nic VOLANSCHI et Christian RINDERKNECHT : *Unparsed Patterns: Easy User-extensibility of Program Manipulation Tools*).

Alors que j'étais invité en tant que chercheur à l'université Eötvös Lórand, à Budapest, j'ai participé à la conception d'un langage de programmation dédié pour les systèmes cyber-physiques distribués, fondé sur Erlang, et à la réalisation d'un compilateur en OCaml. En particulier, j'ai cherché des solutions au problème de la mobilité du code entre nœuds du réseau.

## 7.4 Didactique des langages formels

Mon intérêt pour l'enseignement de la programmation, en particulier à destination des débutants, a débuté dès mon emploi à l'ESILV avec un travail de groupe autour d'une plate-forme web d'assistance à l'apprentissage de la logique formelle (Patrick DUVAL, Agathe MERCERON, Christian RINDERKNECHT et Michel SCHOLL : *LeVinQam: A Question Answering Mining Platform*).

J'ai ensuite développé une réflexion personnelle sur l'usage de la technologie au-delà du web et proposé une méthode d'apprentissage de la récursivité sur les structures de données linéaires à l'aide de la réalité augmentée (Juan Diego TASCÓN VIDARTE, Christian RINDERKNECHT, Jee-In KIM et HyungSeok KIM : *A Tangible Interface for Learning Recursion and Functional Programming*). Plus récemment, j'ai établi une histoire de la didactique de la récursivité dans le domaine de la programmation (Christian RINDERKNECHT : *Teaching and Learning Recursive Programming*).

J'ai mis en évidence de nouvelles façons de présenter certains sujets classiques rencontrés au gré de mes enseignements, pour les rendre plus accessibles (Christian RINDERKNECHT : *A Didactic Analysis of Functional Queues* ; Christian RINDERKNECHT : *A Didactic Analysis of Merge Sort* ; Nachum DERSHOWITZ et Christian RINDERKNECHT : *The Average Height of Catalan Trees by Counting Lattice Paths*). J'ai couché par écrit l'ensemble de ma méthode didactique dans une épaisse monographie concernant la programmation fonctionnelle et l'analyse mathématique des programmes (Christian RINDERKNECHT : *Design and Analysis of Purely Functional Programs*).

# 8 Expérience industrielle

Mon expérience industrielle relève essentiellement du *génie logiciel*.

Lors de mon emploi à Alcatel-Lucent, j'ai mené une étude de la *qualité du logiciel* pour un filiale opérationnelle dont la principale activité était la programmation des réseaux.

J'ai plus tard travaillé à PolySpace Technologies (aujourd'hui MathWorks), où j'ai développé un *analyseur statique* pour JavaCard. J'ai aussi participé à l'automatisation des tests, la rétro-ingénierie et la maintenance, mais j'ai aussi effectué ponctuellement des études clients et du support technique aux commerciaux en avant-vente.

Ces deux expériences, l'une dans une grande entreprise multinationale, l'autre dans une jeune pousse, m'a apporté une compréhension pratique et personnelle de la réalité des jeunes diplômés en informatique qui débutent leur carrière.

Récemment, en *2014*, j'ai quitté l'université pour venir travailler dans une petite entreprise d'électronique, nommée Cortus. J'y développe un compilateur pour un langage à objets, dont la cible finale est un générateur de code LLVM dédié aux processeurs conçus par Cortus. Cette expérience m'apporte une vue complémentaire des phases d'un compilateur : jusqu'à présent, ma spécialité était les parties initiales (*front end*), et j'ai enfin la possibilité de concevoir aussi les optimisations (*middle end*) et la génération de code (*back end*).

# 9 Projets de recherche

## 9.1 Compilation de TTCN-3

TTCN-3 (*Testing and Test Control Notation version 3*) est un langage de programmation fortement et statiquement typé dédié au test de conformité des systèmes de télécommunication, mais constitue aussi un langage de spécification d'interfaces pour les infrastructures de test. Il est utilisé dans les gros projets de télécommunication, par exemple la téléphonie mobile, plutôt que dans le génie logiciel en général. (Il serait d'ailleurs intéressant de déterminer si l'origine de cette différence est culturelle ou technique.)

Du point de vue de la théorie de la programmation, ce langage présente de nombreux aspects intéressants qui méritent une étude académique approfondie en relation avec des entreprises qui l'utilisent. Par exemple, TTCN-3 possède

— une sémantique opérationnelle normalisée,
— un système de type expressif,
— un mécanisme de filtrage d'expressions,
— un paradigme de communication par messages (asynchronisme) ou par activation de procédures distantes (synchronisme) etc.

Ces caractéristiques se retrouvent partiellement dans le langage de programmation fonctionnelle distribuée Erlang, qui est employé dans la programmation des réseaux. D'autres aspects, non cités ici (comme *snapshot semantics*), sont très spécifiques au domaine d'application de TTCN-3. Par ailleurs, il n'existe pas de compilateur TTCN-3 dont le code source soit diffusé à des fins de recherche.

Il me paraît donc prometteur de mettre à profit mon expérience de l'application des méthodes formelles et de la programmation fonctionnelle à la compilation de TTCN-3, et de créer des liens avec l'industrie, comme ce fut le cas avec mon travail sur ASN.1 pendant ma thèse de doctorat — d'ailleurs, ASN.1 est un des langages de données possibles pour TTCN-3. Une piste serait d'écrire une sémantique opérationnelle structurée de TTCN-3 à partir de la norme en langue anglaise, mais la question d'une sémantique par traduction vers Erlang serait aussi à envisager, étant donné que beaucoup de projets sont réalisés directement en Erlang, en particulier le routage, et que cela permettrait donc de distribuer les composants de tests sur un réseau sous forme de superviseurs, ce qui est une architecture classique en Erlang.

## 9.2 Compilation de XSLT

Étant donné un ou plusieurs documents XML, il peut être utile

— de fouiller les documents et afficher ce qui a été trouvé dans un format acceptable pour une autre application, en particulier, XML (filtrage inclusif) ;
— de recopier l'entrée, peut-être sans certaines parties (filtrage exclusif), et/ou ajouter des données (mise à jour).

Quand de telles exigences se présentent, c'est une bonne idée que d'utiliser les langages de programmation fonctionnelle XQuery ou XSLT (anglais, *eXtensible Stylesheet Language Transformations*). Même si les deux langages conviennent à un grand nombre d'usages communs (au point d'avoir en commun un sous-langage, XPath), la première application, qui est plus orientée vers la gestion de bases de données, est plus communément entreprise avec XQuery, alors que le second usage est souvent développé avec XSLT.

Un processeur XSLT lit un document XML et un programme XSLT, ensuite applique au document

des *transformations* définies en XSLT, et le résultat est imprimé, souvent sous forme de texte libre, XML ou HTML. La tournure inattendue est qu'un fichier XSLT est en réalité un document XML, ce qui permet d'utiliser XSLT pour transformer des programmes XSLT.

La future norme XSLT 3.0 verra l'ajout de fonctionnalités puissantes comme les fonctions d'ordre supérieur et les transformations de flux (*streaming*). Contrairement à XQuery, ce langage n'a été que très peu étudié par les théoriciens (une exception est Philip Wadler, au début) et couvre un usage plus large, aussi bien côté serveur (traitement à grande échelle de données XML) que côté client (exécution dans un navigateur web, déchargeant ainsi le serveur). D'après les experts, la clef de l'adoption de toute recherche académique sur XSLT est le traitement exhaustif de la norme internationale qui le défini. C'est ce que je me propose de faire, dans l'esprit de ma thèse de doctorat.

Pour l'instant, la sémantique de XSLT n'est définie qu'en anglais et laisse le choix entre évaluation stricte et paresseuse. Le meilleur compilateur actuel est commercial et réalise cette dernière, car elle ouvre la porte à beaucoup plus d'optimisations automatiques, dans la philosophie des langages de requêtes de base de données relationnelles, ou Haskell. Étant influencé par la tradition des langages de la famille ML, dont OCaml, je prône une évaluation stricte pour que le comportement des programmes soit plus prévisible, en particulier en ce qui concerne l'allocation de la mémoire, même si cela demande au programmeur plus d'habileté pour optimiser ses transformations.

Sur le sujet de la transformation de flux, j'ai en tête d'explorer plus avant les idées du langage jouet XStream (Alain Frisch & Keisuke Nakano), fondé sur la réécriture de termes et qui diffère des requêtes paresseuses et ne force aucune restriction, comme le fragment dit *forward XPath*. En particulier, il faut une réalisation suffisamment générale pour pouvoir servir de cible à un traducteur de XSLT, et il faudrait profiter de l'orthogonalité de cette approche avec les autres travaux de recherche pour la composer avec des optimisations plus classiques (comme la réécriture de sélections XPath). Une autre piste, dans une direction complémentaire, est la distribution de transformations XSLT sur les nœuds d'un réseau et la synchronisation des résultats. Dans cette optique, une compilation vers Erlang mérite d'être explorée.

Ce dernier point nous amène à examiner la relation de XSLT avec d'autres langages de traitement de données XML. Par exemple, il serait intéressant de donner une sémantique de XSLT en le traduisant vers CDuce et dégager au passage un sous-ensemble de XSLT qui bénéficie des mêmes garanties associées au typage statique fort (XSLT n'est que faiblement typé et, même si ses données peuvent être soumises à des schémas, il ne s'agit pas là vraiment d'un système de types).