Dossier de candidature au poste de Maître de conférences

Martin Bodin

Vous trouverez dans ce dossier les éléments suivants :

De plus, vous trouverez les éléments suivants en pièces jointes à ce dossier de candidature :

- Mon diplôme de doctorat en informatique,
- Le rapport de soutenance,
- Trois articles issues de ma recherche, et leurs résumés en français (voir la partie 2.2 page 5 pour plus de détails sur ces publications),
- Les deux rapports de pré-soutenance du doctorat par les rapporteurs Roberto Giacobazzi et Anders Møllers,
- Les lettres de recommandations d'Alan Schmitt, Éric Tanter, et Philippa Gardner.

1 Parcours

Cette première section présente mon curriculum vitae sous un format classique. Vous y trouverez notamment une description de mon parcours académique et la liste de mes publications.

Mon parcours se distingue par mon expérience internationale, avec notamment deux post-docs, un au Chili et l'autre au Royaume-Uni. Mes post-docs ont été tous les deux liés à une expérience de recherche originale, mais aussi à des activités d'enseignements. Ces post-docs m'ont permis de développer des collaborations scientifiques internationales au travers divers projets de recherche. J'ai aussi eu l'opportunité d'encadrer des étudiants lors de mes post-docs (Tomás Diaz et Benjamin Gunton). Le projet avec Tomás Diaz a d'ailleurs abouti à une publication internationale.

Ma recherche est reconnue internationalement : j'ai été par exemple invité à exposer mes travaux sur la formalisation de R [BDT18] au colloque FMfSS (Formal Methods for Statistical Software), aux États-Unis. J'ai aussi participé au comité d'évaluation de POPL ("Principes of Programming Languages" : le symposium international sur les principes des langages de programmation), une conférence internationale $A^\star$, référence dans ce domaine.

Positions

2018–2019	Postdoc à l'Imperial College, à Londres (Royaume-Uni) sur les fondations théoriques de l'analyse de langages de programmation complexes.
2017–2018	Postdoc au Center of Mathematical Modeling (attaché à l'Universidad de Chile), à Santiago (Chili), sur la formalisation du langage de programmation R dans l'assistant de preuve Coq.

Études

2012–2016	Doctorat en Informatique à l'IRISA, à Rennes, sous la direction d'Alan Schmitt et Thomas Jensen sur les analyses certifiées de JavaScript.
2010–2012	Master d'Informatique, mention bien à l'École Normale Supérieure de Lyon.
2009–2010	Licence d'Informatique, mention très bien à l'École Normale Supérieure de Lyon.
2007–2009	Classes préparatoires aux Grandes Écoles au lycée Henri IV à Paris.
2007	Baccalauréat spécialité Mathématiques, mention très bien.

Expérience personnelle

2013–2016	Membre du projet SecCloud : http://www.seccloud.cominlabs.ueb.eu/
2012–2016	Membre du projet JSCert : http://jscert.org/
2012	Stage à Inria (Rennes) avec Alan Schmitt et Thomas Jensen : *A Certified JavaScript Interpreter*
2011	Stage à Vrije Universiteit (Amsterdam) avec Dimitri Hendriks et Jörg Endrullis : *Proving Stream Equalities in Coq*
2011	Stage à PPS (Paris) avec Stéphane Gimenez et Christine Tasson : *Sequentialization of Proof Nets to Structures*
2010	Stage à Vérimag (Grenoble) avec David Monniaux : *Détection de modes de fonctionnements d'un programme Lustre*

Encadrement

2019	Benjamin Gunton, co-encadré avec Philippa Gardner
2017–2018	Tomás Diaz, co-encadré avec Éric Tanter

Enseignement

2019	Separation Logic. Travaux dirigés de M1 à l'Imperial College.
2018	Models of Computation. Travaux dirigés de L2 à l'Imperial College.
2017	Introducción a Coq : Lógica, Tipos y Verificación. Travaux pratiques de M2 à l'Universidad de Chile.
2014	Conception et vérification formelle de programmes. Travaux dirigés de M1 à l'École Normale Supérieure.
2013–2014	Langages formels. Travaux dirigés de L3 à l'École Normale Supérieure.
2013–2014	Introduction à la programmation fonctionnelle. Travaux pratiques de L1 à l'université Rennes 1.
2012	Introduction à Scheme. Travaux pratiques et dirigés de L1 à l'Insa de Rennes.

Prises de responsabilité

2019	Membre du comité d'évaluation des artéfacts pour POPL.
2012, 2014	Surveillance de l'épreuve pratique d'algorithmique et de programmation du concours d'entrée à l'École Normale Supérieure.

Conférences

2019 Martin Bodin, Philippa Gardner, Thomas Jensen, et Alan Schmitt. *Skeletal Semantics et Their Interpretations*. À POPL.

2018 Martin Bodin, Tomás Diaz, et Éric Tanter. *A Trustworthy Mechanized Formalization of R*. À DLS.

2015 Martin Bodin, Thomas Jensen, et Alan Schmitt. *Certified Abstract Interpretation with Pretty-Big-Step Semantics*. À CPP.

2014 Martin Bodin, Arthur Charguéraud, Daniele Filaretti, Philippa Gardner, Sergio Maffeis, Daiva Naudžiūnienė, Alan Schmitt, et Gareth Smith. *A Trusted Mechanised JavaScript Specification*. À POPL.

2013 Jörg Endrullis, Dimitri Hendriks, et Martin Bodin. *Circular Coinduction in Coq Using Bisimulation-up-to Techniques*. À ITP.

2011 David Monniaux et Martin Bodin. *Modular Abstractions of Reactive Nodes using Disjunctive Invariants*. À APLAS.

Colloques

2019 Martin Bodin, Tomás Diaz, et Éric Tanter. *A Trustworthy Mechanized Formalization of R*. À FMfSS.

2018 Martin Bodin. *A Coq Formalisation of a Core of R*. À CoqPL.

2016 Martin Bodin, Thomas Jensen, et Alan Schmitt. *An Abstract Separation Logic for Interlinked Extensible Records*. À JFLA.

2014 Martin Bodin, Thomas Jensen, et Alan Schmitt. *Certified Abstract Interpretation with Pretty-Big-Step Semantics*. À JFLA.

2013 Martin Bodin, Thomas Jensen, et Alan Schmitt. *Pretty-big-step-semantics-based Certified Abstract Interpretation (Preliminary version)*. À *Semantics, Abstract Interpretation, et Reasoning about Programs : Essays Dedicated to David A.* Schmidt

2013 Martin Bodin et Alan Schmitt. *A Certified JavaScript Interpreter*. À JFLA.

Langues

- Français langue maternelle,
- Bon niveau d'anglais,
- Très bon niveau d'espéranto,
- Niveau moyen de portugais,
- Bas niveau d'espagnol,
- Notions d'allemand.

2 Recherche

2.1 Contexte

Ces dernières années ont vu l'usage de langages de script — tels JavaScript, Python, voire R — grandement augmenter. On recense ainsi plus de deux millions d'utilisateurs de R, notamment dans tous les domaines statistiques (biologie, finance, etc.). De son côté, JavaScript est le premier langage sur GitHub, comptabilisant plus de 320 000 projets actifs [Zap].

Les langages de script se caractérisent par une sémantique complexe composée de très nombreux comportements spéciaux. Ces comportements spéciaux tendent à artificiellement produire un résultat même dans les cas où le programme comporte des erreurs, là où des compilateurs pour d'autres langages refuseraient la compilation. JavaScript et R vont ainsi convertir implicitement des valeurs numériques en des chaînes de caractères dans les cas où une fonction attendrait l'un plutôt que l'autre. Ces comportements spéciaux donnent un faux sentiment de correction, les erreurs de programmations étant moins visibles : les programmes retournent parfois des valeurs fausses, mais qui ont l'air superficiellement correctes. Ces comportements spéciaux complexifient énormément l'analyse de programmes. Ils sont donc souvent ignorés ou simplifiés par la recherche. Malheureusement, ces hypothèses de recherche sur l'utilisation pratique de ces fonctionnalités correspondent rarement aux usages pratiques [Ric+11]. Ces comportements spéciaux sont aussi peu connus des développeurs [Bur11], ce qui donne lieu à de nombreuses erreurs dans leurs programmes.

Ma recherche consiste à résorber ce décalage entre les analyseurs et les pratiques d'industrie en prenant en compte les comportements spéciaux des langages de script. Cela conduit à des modèles des langages de très grande taille : la sémantique de JSCert [Bod+14] (qui formalise tous les comportements spéciaux de JavaScript selon sa spécification [ECM11]) compte plus de 900 règles de dérivation. La formalisation CoqR [BDT18] (formalisant les comportements du langage de programmation R) en compte plus de 2 000. À titre de comparaison, le langage Clight spécifié dans le projet CompCert [Ler+08] contient environ 200 règles.[1] Cette grande quantité de règles supplémentaires est due aux comportements spéciaux.

La taille de ces modèles limite très fortement les analyses possibles. Cependant, les analyses de ces modèles sont applicables telles quelles sur des systèmes réels : ces modèles complexes correspondent précisément à des contextes industriels, et non à une simplification de ces derniers. Les contraintes de ma recherche sont ainsi différentes de beaucoup de systèmes de recherche : la taille de ces modèles de langages impose à ce que les analyses soient définies de manière générale afin de passer à l'échelle vis-à-vis de la complexité des langages.

Devant de tels modèles, une question naturelle est de savoir la complexité des modèles peut entrainer des erreurs humaines dans les analyses formelles sous-

1. Le langage Clight tel qu'il est spécifié ne contient en fait que 60 règles de dérivation, mais il est défini dans un style différent de JSCert. Les estimations données ici sont des approximations du nombre de règles obtenues si on spécifiait tous ces langages de façon similaire.

jacentes. Une grande partie de ma recherche consiste donc à formaliser tous ses modèles et analyses dans l'assistant de preuve Coq [CHP+84]. Ce dernier fourni un cadre idéal pour spécifier formellement des systèmes complexes, et pour prouver mathématiquement des propriétés de ces modèles — en particulier la correction des analyses spécifiées. Coq garantit un très haut niveau de confiance sur ces preuves, malgré la complexité des modèles. Cette confiance en Coq est de plus en plus reconnue par l'industrie [Ler14].

2.2 Publications

En pièces jointes, trois publications (ainsi que leurs résumés en français) sont présentées :

- Skeletal Semantics and Their Interpretations [Bod+19],
- A Trustworthy Mechanized Formalization of R [BDT18],
- A Trusted Mechanised JavaScript Specification [Bod+14].

Les articles présentés en annexe ont été choisis car ils représentent bien les axes de recherche principaux que j'ai explorés jusqu'à maintenant : la formalisation de langages de programmation effectivement utilisés, sans simplifications qui pourraient nuire à leur exactitude ; et la conception de techniques de preuves pour ces formalisations souvent complexes.

Ainsi les deux articles [Bod+12 ; BDT18] formalisent respectivement JavaScript et R, deux langages de programmation réputés pour leur complexité. Ces deux articles formalisent ces langages de programmation tels qu'ils sont, sans simplification qui pourrait nuire à la correction des analyses basés sur ces derniers. Ces deux formalisations se distinguent par leur taille : plus de 900 règles de dérivation pour JavaScript et plus de 2 000 pour R. De telles tailles de formalisations n'avaient jamais été étudiés sous cette forme jusqu'à maintenant.

Quant à lui, l'article [Bod+19] présente un langage de description de sémantiques (dans lequel les deux travaux précédant peuvent être convertis) qui permet de construire certains types d'analyses correctes par construction. Cette correction par construction permet de passer à l'échelle vis-à-vis de la complexité des formalisations de langage.

Une grande partie de ma recherche actuelle consiste à appliquer les méthodes proposés par ce dernier article [Bod+19] à des langages de programmation effectivement utilisés, à commencer par ceux formalisés par les deux premiers travaux [Bod+12 ; BDT18]. Ceci permettrait d'aboutir à des analyseurs utilisables en industrie, et donc permettre de détecter des problèmes (de correction ou de sécurité) en amont.

Ces trois articles ont été publiés dans des conférences internationales reconnues dans le domaine. En particulier [Bod+12] et [Bod+19] ont été publiés à POPL, une conférence internationale $A^\star$ référence dans ce domaine. Ces trois articles sont tous accompagnés d'artefacts :

- Pour [Bod+14], la formalisation Coq de JavaScript, ainsi que ne nombreuses informations supplémentaires sur le développement avant et après JSCert : http://jscert.org.
- Pour [BDT18], la formalisation en Coq de R : https://github.com/Mbodin/CoqR/releases/tag/DLS2018.
- Pour [Bod+14], la formalisation en Coq du langage de description décrit dans l'article, ainsi qu'une preuve des principaux résultats de l'article : https://gitlab.inria.fr/skeletons/Coq/tree/POPL2019.

Ce dernier artéfact a été évalué par un comité spécifique comme étant réutilisable, ce qui est la meilleure note possible pour un artefact dans cette conférence.

Ci-dessous se trouve une liste de toutes mes publications, regroupées selon leurs catégories. Tous ces articles ont été acceptés et présentés dans des conférences internationales.

Liste d'articles dans des conférences

[MB11] David Monniaux et Martin Bodin. «Modular Abstractions of Reactive Nodes using Disjunctive Invariants». In : *APLAS*. 2011. DOI : https://doi.org/10.1007/978-3-642-25318-8_5. URL : 10.1007/978-3-642-25318-8_5.

[EHB13] Jörg Endrullis, Dimitri Hendriks et Martin Bodin. «Circular Coinduction in Coq Using Bisimulation-Up-To Techniques». In : *ITP*. 2013. DOI : 10.1007/978-3-642-39634-2_26. URL : https://doi.org/10.1007/978-3-642-39634-2_26.

[Bod+14] Martin Bodin, Arthur Charguéraud, Daniele Filaretti, Philippa Gardner, Sergio Maffeis, Daiva Naudžiūnienė, Alan Schmitt et Gareth Smith. «A Trusted Mechanised JavaScript Specification». In : *POPL*. 2014. DOI : 10.1145/2535838.2535876. URL : https://hal.inria.fr/hal-00910135.

[BJS15] Martin Bodin, Thomas Jensen et Alan Schmitt. «Certified Abstract Interpretation with Pretty-Big-Step Semantics». In : *CPP*. 2015. DOI : 10.1145/2676724.2693174. URL : https://hal.inria.fr/hal-01111588.

[BDT18] Martin Bodin, Tomás Diaz et Éric Tanter. «A Trustworthy Mechanized Formalization of R». In : *DLS*. 2018. DOI : 10.1145/3276945.3276946. URL : https://hal.archives-ouvertes.fr/hal-02409674.

[Bod+19] Martin Bodin, Philippa Gardner, Thomas Jensen et Alan Schmitt. «Skeletal Semantics and their Interpretations». In : *POPL* (2019). DOI : 10.1145/3290357. URL : https://doi.org/10.1145/3290357.

Liste d'articles dans des colloques

[BJS13] Martin Bodin, Thomas Jensen et Alan Schmitt. «Pretty-big-step-semantics-based Certified Abstract Interpretation (Preliminary version)». In : *Festschrift for David Schmidt*. 2013. DOI : 10.4204/eptcs.129.23. URL : https://arxiv.org/abs/1309.5149.

[BS13] Martin Bodin et Alan Schmitt. «A Certified JavaScript Interpreter». In : *JFLA*. 2013. URL : https://hal.inria.fr/hal-00779459.

[BJS14] Martin Bodin, Thomas Jensen et Alan Schmitt. «Pretty-big-step-semantics-based Certified Abstract Interpretation». In : *JFLA*. 2014. DOI : 10.1145/2676724.2693174. URL : https://hal.inria.fr/hal-01111588.

[BJS16] Martin Bodin, Thomas Jensen et Alan Schmitt. «An Abstract Separation Logic for Interlinked Extensible Records». In : *JFLA*. 2016. URL : https://hal.archives-ouvertes.fr/hal-01333600.

[Bod18] Martin Bodin. «A Coq Formalisation of a Core of R». In : *CoqPL*. 2018. URL : https://popl18.sigplan.org/details/CoqPL-2018/3/A-Coq-Formalisation-of-a-Core-of-R.

[Bod19a] Martin Bodin. *A Trustworthy Mechanized Formalization of R*. Formal Methods for Statistical Software (FMfSS). 2019. URL : https://samate.nist.gov/FMSwVRodeo/FMfSS2019.html.

Autres publications

[Bod16] Martin Bodin. «Sémantique et analyse certifiées de JavaScript». Thèse de doct. 2016. URL : https://tel.archives-ouvertes.fr/tel-01478722.

[Bod19b] Martin Bodin. *Skeletal Semantics*. Poster during the Verified Software Workshop. 2019. URL : https://www.doc.ic.ac.uk/~mbodin/esplorado/Verified-Software-Workshop-2019.pdf.

2.3 Projet de recherche

Ma recherche vise à concevoir des outils d'analyses certifiés pour des langages de programmation très utilisés. Ces langages ont beaucoup de comportements spéciaux souvent peu connus et donc susceptibles de mener à des erreurs. Ce projet s'attaque aux problèmes de sécurité associés en proposant des analyses spécifiques pour ces langages. Il s'oriente autour de trois axes :

- La formalisation de langages de programmations utilisés par l'industrie, notamment WebAssembly, JavaScript, et R. Le but est de formaliser complètement ces langages, sans simplifications qui pourraient masquer des comportements spéciaux parfois importants.

- La conception d'analyses passant à l'échelle vis-à-vis de la taille de ces formalisations de langages.
- Relier ces travaux à d'autres systèmes d'analyses déjà existant.

Formalisation de langages de programmation. J'ai déjà participé à la formalisation en Coq de JavaScript [Bod+14] et R [BDT18]. Ces formalisations étaient très novatrices lors de leur publication : c'était les premières formalisations en Coq de ces langages de programmation, mais c'était aussi les premières formalisations de ces langages de programmation dont la correction était assurée par deux méthodes complémentaires (correspondance ligne-à-ligne et test). Cependant, ces formalisations peuvent être améliorées. D'une part parce que JavaScript et R évoluent rapidement et leurs spécifications ont donc changé entre-temps. Il est donc important de concevoir une sémantique formelle qui puisse évoluer au même rythme que ces spécifications.

D'autre part, il s'est avéré que ces formalisations demandent un grand travail pour pouvoir être utilisés dans une analyse certifiée. Ces formalisations sont en effet très grandes et poussent les capacités des assistants de preuve à leurs limites. Les squelettes [Bod+19] sont une représentation formelle de sémantiques conçue pour pallier à ce problème : un squelette permet la définition et la preuve de divers analyses dans un cadre général. Ce formalisme est ainsi adapté à la formalisation de langages de programmation complexes. Il est cependant très récent, et les travaux de ce projet pourront amener à des évolutions de ce dernier.

Ce premier axe consiste en la formalisation de nouveaux langages de programmation, en particulier WebAssembly [Wat18], dont la formalisation Coq est en cours, mais aussi en la mise à jour des formalisations de JavaScript et de R. Ces formalisations seront conçues pour être réutilisables et permettre la conception et la certification d'analyseurs. De plus, ces formalisations seront construites pour être exhaustives vis-à-vis des comportements spéciaux de ces sémantiques : comme beaucoup d'erreurs viennent de ces comportements spéciaux, ils ne peuvent être simplifiés dans les formalisations. L'approche de JSCert et de CoqR — consistant à pouvoir contrôler ces formalisations, tant par leur ressemblance à la spécification de référence, tant par le test — sera conservée.

Conception d'analyses passant à l'échelle. Les squelettes n'ont pour l'instant été associé qu'à un unique type d'analyses, (une sous-classe de l'interprétation abstraite), et ce de manière assez théorique [Bod+19]. Cet axe vise à développer l'ensemble des analyses possibles, mais aussi à les mettre en pratique sur les langages précédemment formalisés. En particulier, l'exploration de domaines relationnels [CH78 ; Min06] et d'autres domaines déjà utilisés et formalisés dans des analyseurs tels que Verasco [Jou16]. Ces domaines rentrent dans le cadre plus général de l'interprétation abstraite et représentent un premier objectif pour le développement de nouvelles analyses sur les squelettes.

De manière orthogonale, la logique de séparation [IO01 ; BCO05] a montré pouvoir raisonner efficacement sur des structures complexes et représente ainsi un objectif important pour la formalisation d'analyses. En particulier,

Iris [Jun+17b] a récemment proposé une généralisation importante de cette logique. Iris est formalisée en Coq, mais sa partie sémantique n'a pas été conçue pour des langages complexes : le langage le plus complexe formalisé dans le contexte d'Iris est à ma connaissance RustBelt [Jun+17a] et ne contient que 30 règles de dérivations (à comparer aux 900 règles de JSCert). Les squelettes pourraient fournir un cadre formel intéressant pour la partie sémantique d'Iris, permettant de passer à l'échelle sur des langages de programmation complexes.

D'autres types d'analyses sont aussi possibles. Des travaux préliminaires ont par exemple été faits sur le déterminisme des squelettes, et plus généralement des hyperpropriétés de programme [CS10]. Cet axe de recherche vise à formaliser ces différents types d'analyses dans le cadre des squelettes, mais aussi à les implémenter en pratique : d'une part dans des cas d'applications simples de langages jouets, mais surtout d'autre part sur les langages formalisés dans le premier axe de ce projet de recherche. De tels analyseurs se concentreraient sur tous les problèmes liés aux comportements spéciaux des sémantiques et offriront des applications concrètes immédiates : dans le cas de R, il n'existe pas encore d'analyseurs du langage [Bur11] et ces analyseurs seront donc les premiers.

Symbiose avec d'autres travaux. De nombreux projets d'analyses [Bes+10 ; Fra+17] se concentrent sur des langages intermédiaires. De tels langages sont généralement associés à une sémantique beaucoup plus simple que le langage d'origine, ce qui simplifie les analyses. En particulier, il existe des langages intermédiaires génériques [Gar+19 ; RŞ10], déjà associés à divers analyses. Ceci est une opportunité pour les squelettes : il est possible de concevoir des interprétations squelettiques produisant des programmes dans ces différents langages intermédiaires, conduisant ainsi à des compilateurs génériques. Ces compilateurs peuvent être vérifiés de manière générique grâce aux outils fournis par les squelettes.

Cette approche de compilation des squelettes ouvre une voie intéressante. D'une part, elle permet de lier toute la recherche sur les squelettes avec ces différents analyseurs. D'autre part, il est possible de formellement spécifier l'analyse de ces analyseurs dans le cadre des squelettes, et de montrer que la compilation conserve ces propriétés. Ceci permet ainsi de reformuler l'action de ces analyseurs dans un contexte formel et général, conduisant à une formalisation de ces derniers (ils ne sont que très rarement formalisés dans un assistant de preuve).

Cet axe pose enfin la question de la compilation certifiée. Il existe déjà un compilateur de squelette vers OCaml [CCS19], mais ce dernier n'est pas formalisé en Coq. Cet axe de recherche permettra de donner des bases solides pour une telle formalisation.

3 Enseignements

Mes activités d'enseignement ont été jusqu'à maintenant assez proches de mon domaine de recherche : programmation fonctionnelle, analyse de programe, etc. J'ai cependant essayé de diversifier les approches : en plus de mon domaine principal, l'interprétation abstraite, j'ai aussi enseigné des domaines proches mais différents comme la logique de séparation ou le model checking.

J'ai enseigné principalement en français et en anglais et je suis bien entendu prêt à continuer à enseigner dans ces deux langues. Je suis aussi prêt à m'adapter aux circonstances : par exemple les TP de Coq que j'ai encadré au Chili étaient majoritairement en espagnol, le cours associé étant dans cette langue et les élèves étant naturellement plus à l'aise dans cette dernière.

Le tableau ci-dessous récapitule les informations principales des cours auxquels j'ai participé, en particulier les heures équivalent TD. Une description plus précise de chacun de mes enseignements est décrite après ce tableau. Dans tous ces cours, j'ai bien entendu participé à la correction des examens avec les autres responsables du cours.

Nom du cours	année	université	niveau	séances	h eq. TD
Introduction à Scheme	2012	INSA Rennes	L1	28h TD, 13h TP	36,7
Introduction à la programmation fonctionnelle	2013 2014	Rennes 1	L1	26h TP 26h TP	17,3 17,3
Langages formels	2013 2014	ENS Rennes	L3	30h TD 30h TD	30 30
Conception et vérification formelle de programmes	2014	ENS Rennes	M1	26h TD	26
Introducción a Coq	2017	Universidad de Chile	M2	22h TD, 3h CM	26,5
Models of Computation	2018	Imperial College London	L2	27h CTD	27
Separation Logic	2019	Imperial College London	M1	28h CTD	28

Introduction à la programmation fonctionnelle Ce cours consistait en une introduction à la programmation fonctionnelle en Lisp. J'étais chargé des TP.

Introduction à Scheme Ce cours à l'INSA consistait en des cours/TD et des TP de programmation fonctionnelle, en Scheme. J'étais chargé d'à la fois les cours/TD et des TP d'un groupe.

Il y avait des projets, et j'ai naturellement participé à leur notation. J'ai pris grand soin de donner des commentaires les plus personnalisés possibles, de façon à ce que chaque élève sache comment il peut progresser. Mes élèves avaient beaucoup appréciés ces retours.

Langages formels Ce cours présente des bases d'informatique théorique (langages réguliers, automates, grammaires, etc.) à l'ENS. Je me suis chargé des TD. Afin de former les élèves à plus de rigueur mathématique dans leurs preuves, il était demandé chaque semaine à un groupe d'élève de rédiger une correction des

TD au format TeX. Arnaud Jobin et moi étions responsables de la correction de ces corrigés. Une attention tout particulière a été portée à donner des retours constructifs aux élèves, qui a été appréciée.

Conception et vérification formelle de programmes Ce cours conçu avec François Schwarzentruber propose d'aborder divers formalismes utilisés pour décrire formellement un langage, des diagrammes UML utilisés en industrie à la logique modale utilisée en analyse de programme concurrents. Je me suis chargé des TD. J'ai de plus conçu deux TP pour ce cours. L'un deux consistait en une introduction à Spin, un model checker très utilisé. Les élèves devaient formaliser le problème du dîner des philosophes dans ce contexte. L'autre consistait en une introduction à l'assistant de preuve Coq. J'ai aussi conçu du matériel pour TD dont le but consistait à approcher de manière ludique les différents formalismes vus en cours durant le semestre. Ces différents matériels de TP et de TD sont accessibles depuis ma page web à l'adresse `http://www.doc.ic.ac.uk/~mbodin/instruado/2013/CVFP/`. Le cours se terminait pour les étudiants sur un travail de documentation, où ils devaient étudier et présenter un article du domaine de leur choix (soit parmi une liste de publication que nous leur avions proposé, soit un article externe de leur choix). J'ai naturellement activement participé à la notation de leurs présentations.

Introducción a Coq Éric Tanter a profité de mon expertise Coq pour ouvrir ce cours basé sur le matériel de cours de Software Foundations [Pie+10], que nous avons orienté vers la vérification de programme. Ce cours introduit Coq sur des exercices simples, puis définit une sémantique d'un langage While pour finir sur de la preuve de propriétés sur un tel programme. Je me suis chargé des TP. Le cours s'est terminé sur des présentations techniques de Coq dans des contextes de recherche. Nous avons ainsi invité Federico Olmedo à présenter ses recherches. Mon travail sur JSCert [Bod+14] s'inscrivant exactement dans la continuité de ce cours, j'ai ainsi effectué trois heures de cours et un TP présentant JSCert et proposant aux élèves des exercices sur ce projet.

Models of Computation Ce cours donne des bases en informatique théorique (sémantiques grands et petits pas, triplets de Hoare, λ-calcul, etc.) à l'Imperial College London. Le cours était sous la forme de cours/TD pour un grand nombre d'étudiants : il y avait 166 étudiants et nous étions responsables pour les aider.

Separation Logic Ce cours de niveau M1 donne les bases de la logique de séparation, majoritairement vue depuis l'approche de Peter O'Hearn [Rey08 ; IO01]. Ce cours se conclut par la participation de chercheurs présentant leurs travaux, notamment l'outil Infer [CD11] de Facebook et Gillian [Fra+19] de l'Imperial College London. Comme Models of Computation, ce cours était sous la forme de cours/TD.

L'avantage de la logique de séparation est qu'elle permet de formaliser relativement simplement de nombreuses structures de données. J'ai donc pu participer à l'élaboration du sujet d'examen en proposant des structures de données que les élèves n'avaient pas vu en cours (listes XOR, union-find, etc.). J'ai ainsi rédigé une des parties de l'examen. J'ai aussi participé à la production de matériel pédagogique pour le cours, notamment tous les polycopiés liés à la preuve de correction de la logique de séparation présentée pour la première année dans ce cours.

Références

[CH78] Patrick Cousot et Nicolas Halbwachs. «Automatic discovery of linear restraints among variables of a program». In : *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM. 1978, p. 84-96.

[CHP+84] Thierry Coquand, Gérard Huet, Christine Paulin et al. *the Coq Proof Assistant*. 1984. url : https://coq.inria.fr/.

[IO01] Samin Ishtiaq et Peter O'Hearn. «BI as an Assertion Language for Mutable Data Structures». In : *POPL* (2001).

[BCO05] Josh Berdine, Cristiano Calcagno et Peter W. O'Hearn. «Smallfoot : Modular Automatic Assertion Checking with Separation Logic». In : *International Symposium on Formal Methods for Components and Objects*. 2005.

[Min06] Antoine Miné. «The Octagon Abstract Domain». In : *Higher-Order and symbolic computation* (2006).

[Ler+08] Xavier Leroy, Sandrine Blazy, Zaynah Dargaye et Jean-Baptiste Tristan. *CompCert: Compilers you can formally trust*. 2008. url : http://compcert.inria.fr/.

[Rey08] John C. Reynolds. «An Introduction to Separation Logic». In : *Engineering Methods and Tools for Software Safety and Security* (2008).

[Bes+10] Frédéric Besson, Thomas Jensen, David Pichardie et Tiphaine Turpin. «Certified result checking for polyhedral analysis of bytecode programs». In : *International Symposium on Trustworthy Global Computing*. Springer. 2010, p. 253-267.

[CS10] Michael R Clarkson et Fred B Schneider. «Hyperproperties». In : *Journal of Computer Security* 18.6 (2010), p. 1157-1210.

[Pie+10] Benjamin C Pierce, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg et Brent Yorgey. *Software foundations*. 2010. url : http://www.cis.upenn.edu/bcpierce/sf/current/index.html.

[RŞ10] Grigore Roşu et Traian Florin Şerbănuţă. «An Overview of the $\mathbb{K}$ Semantic Framework». In : *Journal of Logic and Algebraic Programming* (2010).

[Bur11] Patrick Burns. *The R Inferno*. 2011.

[CD11] Cristiano Calcagno et Dino Distefano. «Infer : An automatic program verifier for memory safety of C programs». In : *NASA Formal Methods Symposium*. 2011.

[ECM11] ECMA International, éd. *ECMAScript Language Specification. Standard ECMA-262, Edition 5.1*. 2011. URL : http://www.ecma-international.org/publications/standards/Ecma-262-arch.htm.

[MB11] David Monniaux et Martin Bodin. «Modular Abstractions of Reactive Nodes using Disjunctive Invariants». In : *APLAS*. 2011. DOI : https://doi.org/10.1007/978-3-642-25318-8_5. URL : 10.1007/978-3-642-25318-8_5.

[Ric+11] Gregor Richards, Christian Hammer, Brian Burg et Jan Vitek. «The `eval` that Men Do. A Large-Scale Study of the Use of `eval` in JavaScript Applications». In : *ECOOP*. 2011.

[Bod+12] Martin Bodin, Arthur Charguéraud, Daniele Filaretti, Philippa Gardner, Sergio Maffeis, Daiva Naudžiūnienė, Alan Schmitt et Gareth Smith. *JSCert : Certified JavaScript*. 2012. URL : http://jscert.org/.

[BJS13] Martin Bodin, Thomas Jensen et Alan Schmitt. «Pretty-big-step-semantics-based Certified Abstract Interpretation (Preliminary version)». In : *Festschrift for David Schmidt*. 2013. DOI : 10.4204/eptcs.129.23. URL : https://arxiv.org/abs/1309.5149.

[BS13] Martin Bodin et Alan Schmitt. «A Certified JavaScript Interpreter». In : *JFLA*. 2013. URL : https://hal.inria.fr/hal-00779459.

[EHB13] Jörg Endrullis, Dimitri Hendriks et Martin Bodin. «Circular Coinduction in Coq Using Bisimulation-Up-To Techniques». In : *ITP*. 2013. DOI : 10.1007/978-3-642-39634-2_26. URL : https://doi.org/10.1007/978-3-642-39634-2_26.

[Bod+14] Martin Bodin, Arthur Charguéraud, Daniele Filaretti, Philippa Gardner, Sergio Maffeis, Daiva Naudžiūnienė, Alan Schmitt et Gareth Smith. «A Trusted Mechanised JavaScript Specification». In : *POPL*. 2014. DOI : 10.1145/2535838.2535876. URL : https://hal.inria.fr/hal-00910135.

[BJS14] Martin Bodin, Thomas Jensen et Alan Schmitt. «Pretty-big-step-semantics-based Certified Abstract Interpretation». In : *JFLA*. 2014. DOI : 10.1145/2676724.2693174. URL : https://hal.inria.fr/hal-01111588.

[Ler14] Xavier Leroy. «How much is a mechanized proof worth, certification-wise». In : *Principles in Practice* (2014).

[BJS15] Martin Bodin, Thomas Jensen et Alan Schmitt. «Certified Abstract Interpretation with Pretty-Big-Step Semantics». In : *CPP*. 2015. DOI : 10.1145/2676724.2693174. URL : https://hal.inria.fr/hal-01111588.

[Bod16] Martin Bodin. «Sémantique et analyse certifiées de JavaScript». Thèse de doct. 2016. URL : https://tel.archives-ouvertes.fr/tel-01478722.

[BJS16] Martin Bodin, Thomas Jensen et Alan Schmitt. «An Abstract Separation Logic for Interlinked Extensible Records». In : *JFLA*. 2016. URL : https://hal.archives-ouvertes.fr/hal-01333600.

[Jou16] Jacques-Henri Jourdan. «Verasco: a Formally Verified C Static Analyzer». Thèse de doct. Université Paris Diderot-Paris VII, 2016.

[Fra+17] José Fragoso Santos, Petar Maksimović, Daiva Naudžiūnienė, Thomas Wood et Philippa Gardner. «JaVerT : JavaScript verification toolchain». In : *Proceedings of the ACM on Programming Languages* 2.POPL (2017), p. 50.

[Jun+17a] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers et Derek Dreyer. «RustBelt : Securing the foundations of the Rust programming language». In : *Proceedings of the ACM on Programming Languages* 2.POPL (2017), p. 66.

[Jun+17b] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Aleš Bizjak, Lars Birkedal et Derek Dreyer. «Iris from the Ground Up». In : *Submitted to JFP* (2017).

[Bod18] Martin Bodin. «A Coq Formalisation of a Core of R». In : *CoqPL*. 2018. URL : https://popl18.sigplan.org/details/CoqPL-2018/3/A-Coq-Formalisation-of-a-Core-of-R.

[BDT18] Martin Bodin, Tomás Diaz et Éric Tanter. «A Trustworthy Mechanized Formalization of R». In : *DLS*. 2018. DOI : 10.1145/3276945.3276946. URL : https://hal.archives-ouvertes.fr/hal-02409674.

[Wat18] Conrad Watt. «Mechanising and verifying the WebAssembly specification». In : *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM. 2018, p. 53-65.

[Bod19a] Martin Bodin. *A Trustworthy Mechanized Formalization of R*. Formal Methods for Statistical Software (FMfSS). 2019. URL : https://samate.nist.gov/FMSwVRodeo/FMfSS2019.html.

[Bod19b] Martin Bodin. *Skeletal Semantics*. Poster during the Verified Software Workshop. 2019. URL : https://www.doc.ic.ac.uk/~mbodin/esplorado/Verified-Software-Workshop-2019.pdf.

[Bod+19] Martin Bodin, Philippa Gardner, Thomas Jensen et Alan Schmitt. «Skeletal Semantics and their Interpretations». In : *POPL* (2019). DOI : 10.1145/3290357. URL : https://doi.org/10.1145/3290357.

[CCS19] Nathanaël Courant, Enzo Crance et Alan Schmitt. «Necro : Animating Skeletons». In : *ML 2019*. Berlin, Germany, août 2019.

[Fra+19] José Fragoso Santos, Petar Maksimović, Gabriela Sampaio et Philippa Gardner. «JaVerT 2.0 : Compositional Symbolic Execution for JavaScript». In : 2019. doi : 10.1145/3290379.

[Gar+19] Philippa Gardner, José Fragoso Santos, Maksimović Petar et Sacha-Élie Ayoun. *Gillian : A General Static Analysis Framework based on Separation Logic*. ECOOP Summer School. 2019.

[Zap] Carlo Zapponi. *GitHut, A small place to discover languages in GitHub*. url : https://github.info/ (visité le 2019).