Défi Inria

LiberAbaci : Adaptation des Systèmes de preuve pour l'enseignement des mathématiques universitaires

Version : c7ca5d9 2022-01-14 14:20:25 +0100

- **Coordinateur :** Yves Bertot — **email :** Yves.Bertot@inria.fr

Équipe du coordinateur (EP) : Stamp

Centre de recherche (CRI) : Sophia Antipolis

- **Partenaires (EP/CRI)**

  — Cambium, Paris (contact : Jean-Marie Madiot)
  — Camus, Nancy (Strasbourg) (contact : Arthur Charguéraud)
  — Gallinette, Rennes (Nantes) (contact : Nicolas Tabareau)
  — Pi.r2 (PiCube à partir du 1er juillet 2022), Paris (contact : Hugo Herbelin)
  — Spades, Grenoble (contact : Martin Bodin)
  — Stamp, Sophia Antipolis (contact : Yves Bertot)
  — Toccata, Saclay (contact : Sylvie Boldo)

- **Partenaires extérieurs**
  — Laboratoire d'Informatique de Paris-Nord (contact : Micaela Mayero)

Résumé Il s'agit d'étudier les problèmes scientifiques soulevés par l'utilisation d'outils de preuve sur ordinateur dans l'enseignement des mathématiques durant les premières années universitaires.

La théorie des types, qui sert de fondement théorique à Coq, a montré sa puissance pour des applications informatiques. Si l'on se tourne vers le public mathématique et l'utilisation en enseignement, il faut repenser l'environnement de travail, les outils de documentation ou de recommandation, et les contenus formalisés. Il faut aussi prendre en compte la multiplicité de points de vue que les mathématiciens et les étudiants peuvent avoir sur un même objet et l'évolution de ces points de vue au cours de l'apprentissage.

Motivation

Les systèmes de preuve sur ordinateur sont à l'interface entre l'informatique et les mathématiques. Dès que les premières règles de raisonnement logique sont implémentées dans un système de preuve, il est raisonnable de chercher à comprendre quelles sont les mathématiques que l'on peut représenter de façon naturelle dans ces systèmes. Les premières expériences comme celles d'Automath se sont tournées vers la formalisation de certains domaines des mathématiques (une description formelle d'un livre de Landau sur les fondements de l'analyse).

Le domaine de la preuve sur ordinateur a ensuite obtenu la majeure partie de ses résultats les plus marquants dans des domaines concernant l'informatique et plus particulièrement la théorie des langages de programmation (compilateurs et outils d'analyse), les algorithmes (tri, chiffrement), le calcul numérique (nombres flottants, calcul réel exact), les protocoles, les automates. La formalisation de résultats de mathématiques a continué avec des sujets en théorie des nombres (algorithmes de vérification pour de grands nombres premiers, théorème fondamental de l'arithmétique sur la densité asymptotique des nombres premiers), les graphes (théorème des 4 couleurs), les groupes finis (théorèmes de Sylow et de Feit-Thompson), les polynômes (algorithme de Buchberger), les matrices, etc. Des efforts ont également été faits pour détecter des théorèmes mathématiques particulièrement représentatifs et étudier la capacité des systèmes de preuve à fournir des preuves vérifiées formellement de ces théorèmes, voir par exemple la liste des 100 théorèmes de F. Wiedijk.

En conséquence, les mathématiciens s'intéressent de plus en plus à l'utilisation des systèmes de preuve, que ce soit pour étudier leurs résultats les plus récents ou pour les utiliser en enseignement.

Utiliser un système de preuve sur ordinateur pour enseigner les mathématiques pose des problèmes nouveaux. L'ensemble des travaux cités plus haut représente souvent des « premières », des tours de force effectués par des individus ou des équipes ayant développé un niveau d'expertise exceptionnel dans l'utilisation des systèmes de preuve. Pour enseigner, il faut s'adapter à un public qui n'a pas l'objectif d'atteindre ce niveau d'expertise. Il est nécessaire que le langage des systèmes de preuve reflète le langage utilisé par les mathématiciens pour communiquer, que l'outil soit facile à maitriser, qu'il existe des contenus adaptés, et que la pratique s'intègre dans un processus social attrayant et viable.

Indépendamment du contenu mathématique enseigné, l'utilisation de systèmes de preuve a l'avantage potentiel d'apprendre aux étudiants à reconnaitre les raisonnements valides, à « ne pas dire n'importe quoi », et à développer leur esprit critique, au moins pour ce qui concerne les disciplines scientifiques.

Les systèmes de preuve sont déjà utilisés dans des conditions d'enseignement et il y a un workshop sur le sujet : ThEdu. En étudiant les publications de cette conférence, il apparaît que la majeure partie des expériences est effectuée par des enseignants d'informatique pour un public informaticien. Il est assez naturel que les informaticiens trouvent l'enseignement des méthodes formelles pertinent, car ce sujet est fortement relié à la nécessité de produire du logiciel correct. L'adoption par un public mathématicien pose des question différentes, parce qu'il faut convaincre que le temps investi à apprendre à se servir d'un nouvel outil

sera profitable pour la dissémination des connaissances mathématiques.

Nous souhaitons mettre en place un projet permettant de fédérer les efforts des chercheurs Inria autour de l'utilisation et de l'enseignement des mathématiques formalisés dans des assistants de preuve. En effet, la recherche sur les outils de preuve déborde déjà très largement les frontières d'une équipe Inria. Le travail sous-jacent implique de réfléchir aux outils de preuve sur plusieurs niveaux : le langage, l'environnement de travail, et le contenu formalisé. Pour chacun de ces aspects nous prévoyons un travail collaboratif incluant des chercheurs de plusieurs équipes.

Pour le langage, nous voulons profiter de l'expertise présente dans trois équipes : Gallinette, PiCube et Stamp. Ces équipes ont déjà travaillé sur les fondements de la théorie des types et collaboreront donc à la mise au point d'une variante de cette théorie adaptée pour les mathématiques : une variante où l'isomorphisme de Curry-Howard est repensé pour réduire les points d'achoppements pour des mathématiciens et où l'exploitation d'équivalences et isomorphismes est plus naturelle.

Pour l'environnement de travail, plusieurs équipes de l'Inria sont déjà engagées dans le développement d'outils de preuves variés, comme Dedukti, Coq et Abella. Même pour Coq, plusieurs équipes sont impliquées dans son développement. À côté des problèmes de langage, il faut réfléchir aux fonctionnalités qui permettent de supporter les notations en usage dans le public mathématicien, en évitant la verbosité imposée par le niveau théorique, à l'aide d'outils automatiques pour inférer les données utiles : unification, recherche automatique de classes de types, projections canoniques, etc. L'expertise traditionnelle sur ces sujets est également répartie entre les équipes sus-mentionnées, mais nous voulons également explorer des idées originales proposées par Camus et Spades. Cette réflexion sur l'environnement de travail devra également inclure la question de rendre abordable nos travaux pour un public peu à l'aise avec l'informatique.

La tâche de construire des bibliothèques pour des pans entiers des mathématiques a déjà été engagée par différents groupes dans le monde entier, que ce soit avec Isabelle, HOL Light, Lean, ou Mizar. Pour chacun de ces efforts, le nombre de personnes impliquées dépasse la taille usuelle d'une équipe Inria. De plus, les bibliothèques construites sont le reflet de la population impliquée dans leur développement : il est important de faire ce travail de façon collaborative pour obtenir un contenu compréhensible par un public débutant et d'origine diverse. Pour ce travail, nous envisageons des collaborations entre Toccata, Stamp, Cambium et PiCube.

Ce travail devra s'intégrer dans l'environnement scientifique et social de deux façons différentes. D'une part, il faut prendre en compte le fait que de nombreux efforts sont faits dans le monde pour rendre les systèmes de preuves abordables pour la communauté mathématiques, et qu'il existe donc déjà un matériel conséquent qu'il faudrait exploiter si possible. D'autre part, il faut s'assurer que nos travaux s'adaptent à la pratique de l'enseignement des mathématiques, en intégrant le plus possible des enseignants des premières années universitaires dans nos réflexions.

Contexte

La preuve sur ordinateur connait actuellement un essor remarquable, en montrant son

efficacité pour la production de systèmes sûrs, avec des applications en télécommunication, compilation et analyse de programmes, cryptographie, commerce électronique, programmation quantique, etc. Au delà de ces applications, les outils de preuve sur ordinateur ont le potentiel de jouer un rôle clef dans la production scientifique, en permettant la reproductibilité des résultats, en augmentant la confiance dans les publications, et en facilitant l'exploration audacieuse d'idées créatrices, le besoin de rigueur et les tâches fastidieuses étant délégués à l'ordinateur.

Ceci s'applique en particulier aux mathématiques, et un nombre grandissant de mathématiciens s'intéresse aux mathématiques formalisées. Il est important d'organiser nos travaux de recherche pour soutenir cet essor : comprendre comment adapter un langage informatique à la pratique, faciliter l'apprentissage des techniques clefs du domaine, mettre en place de nouvelles habitudes chez les praticiens.

Un tour d'horizon de la formalisation des mathématiques

Les expériences réalisées avec Automath n'ont pas résisté à l'usure du temps. En revanche, Mizar est un outil qui a été développé dès les années 1970 avec l'objectif explicite de formaliser les mathématiques (pratiquement à l'exclusion de sujets issus de l'informatique) et est encore utilisable aujourd'hui.

Les expériences réalisées avec Lean, par exemple visibles sur le blog Xena-project, montrent comment les outils de preuve peuvent être utilisés par des mathématiciens pour résoudre des problèmes d'actualité. Lean est d'ailleurs au centre d'un mouvement remarquable d'adhésion de la communauté mathématique, que nous devrons observer avec attention, pour en reprendre les meilleures idées et éventuellement du matériel, lorsque ce sera possible.

La communauté des utilisateurs du prouveur Isabelle entreprend également un fort investissement pour l'utilisation de ce système par des mathématiciens. Une partie de ce travail passe par la maintenance d'une bibliothèque centrale des développements effectués en Isabelle, archive of formal proofs. L'autre partie bénéficie de la bourse ERC Alexandria attribuée à Larry Paulson.

Le système HOL Light bénéficie aussi d'un soutien visible par la communauté mathématique, d'une part parce que sa bibliothèque d'analyse réelle et complexe a longtemps été en avance sur ses concurrents, et d'autre part parce que ce système est resté essentiellement le projet d'une seule personne, ce qui rend sa taille et sa complexité abordable pour un utilisateur soucieux de convaincre un public réticent. C'est par exemple l'une des raisons du choix de ce système par T. Hales pour la vérification formelle de sa preuve de la conjecture de Kepler, le projet Flyspeck.

L'Inria soutient depuis des décennies les recherches sur la preuve sur ordinateur, avec un succès remarquable sur les questions de fiabilité des langages de programmation, de sécurité informatique et pour des théorèmes mathématiques choisis (Feit-Thompson, Lax-Milgram, etc). L'objectif global est d'améliorer la fiabilité des objets technologiques en profitant de la rigueur apportée par la preuve sur ordinateur. Les questions de fiabilité en robotique, en commerce électronique, en géométrie algorithmique, et en complexité des algorithmes reposent parfois sur des questions d'algèbre et d'analyse. Pour certains algo-

rithmes randomisés et pour la cryptographie, on se repose également sur des questions de probabilités. La recherche sur les outils de preuve et leur application à l'industrie doit donc inclure une réflexion forte sur le développement de contenus de mathématiques formalisées.

Grâce à l'existence du système Coq, développé majoritairement à l'Inria et avec de nombreuses bibliothèques maintenues par la communauté des utilisateurs, nous disposons déjà d'une grande quantité de matériel, à la fois par le langage lui-même et le système interactif qui vient avec et par les différentes bibliothèques de mathématiques qui présentent des connaissances déjà représentées de façon formelle.

Les bibliothèques que nous pouvons citer sont : la bibliothèque standard, les réels « standards » de Coq, Coquelicot, CoqInterval, C-CoRN, Mathematical Components, MathClasses, FOST, MILC, mathcomp-analysis. Plusieurs de ces bibliothèques sont construites les unes sur les autres (MILC au dessus de CoqInterval, Coquelicot, et les réels standards, mathcomp-analysis au dessus de Mathematical Components) et partagent quelques dépendances (Coquelicot dépend d'une partie de Mathematical Components). Des ponts ont également été décrits entre les réels standards et Mathematical Components et C-CoRN, etc. Ces bibliothèques ont été utilisées pour établir certains résultats marquants, tels que le théorème de Feit-Thompson, un théorème de théorie des groupes finis (avec Mathematical Components), la description formelle d'un programme de simulation d'onde (dans FOST, avec Coquelicot), le théorème de Lax-Milgram (dans MILC), la preuve de transcendance de π (avec Mathematical Components et Coquelicot), l'irrationalité de $\zeta(3)$ (avec Mathematical Components et Coquelicot).

Les mathématiques sur ordinateur sont aussi l'objet de la thématique du calcul formel, où l'ordinateur est utilisé pour effectuer des traitements symboliques sur des formules mathématiques, souvent lorsque ces traitements ont une taille hors d'atteinte pour une opération purement humaine. Des expériences récentes ont montré que ces traitements symboliques pouvaient aussi souffrir de problèmes de fiabilité, et Assia Mahboubi a obtenu une bourse ERC (appelée FRESCO) pour attaquer précisément ce problème, avec Guillaume Melquiond comme partenaire. Le sujet de FRESCO est très proche de celui de ce défi, puisqu'il s'agit de preuves formelles pour les mathématiques sur ordinateur, mais le travail de recherche envisagé est la production de calculs symboliques rapides et fiables en se basant sur la théorie des langages de programmation. Le public visé est donc expert à la fois en mathématiques et en informatique. Les points d'interaction possibles entre FRESCO et LiberAbaci concernent le besoin d'utiliser des isomorphismes entre plusieurs présentations du même concept, le besoin d'organiser les hiérarchies de structures, et le besoin de développer une interface homme-machine confortable pour le système de calcul formel.

La cible choisie et l'impact espéré

Le défi final est que l'utilisation de systèmes de preuve dans les cours universitaires devienne une étape naturelle. Il faut pour cela que le langage d'écriture des théorèmes et des preuves soit le plus naturel possible, que les garanties apportées par la preuve formelle soient comprises et emportent l'adhésion des enseignants, et que les contenus disponibles permettent une prise en main rapide avec des bénéfices immédiats.

L'utilisation de systèmes de preuve en enseignement des mathématiques a le potentiel d'améliorer l'enseignement des mathématiques tout court, en rendant plus palpable la notion de preuve, en permettant aux étudiants de bien comprendre le sens des connecteurs logiques, et en donnant accès aux détails des preuves. Bien sûr, ceci ne peut passer que par une collaboration en bonne intelligence avec les enseignants en mathématiques.

Les mathématiques dont nous visons la formalisation font partie du socle commun à de nombreux ingénieurs, y compris des ingénieurs travaillant sur des questions de physique, de génie civil, d'aéronautique, d'automatique, etc. Il y a également des parties de l'informatique où il est important de disposer de plus d'outils mathématiques pour garantir la correction des algorithmes considérés. Rendre les outils de preuve sur ordinateur plus abordables pour ces ingénieurs permettra d'ajouter de nouveaux outils dans leur environnement de travail. La question de faire des preuves formelles sur les autres outils mathématiques sur ordinateur pour les ingénieurs a du sens et ne sera pas traitée dans ce défi (cette question est reliée au projet ERC FRESCO d'A. Mahboubi).

Un exemple d'impact pour l'informatique est l'étude de la complexité asymptotique des algorithmes. Une erreur de programmation dans un algorithme peut mener à un comportement asymptotique impraticable, ce qui correspond souvent en pratique à un logiciel inutilisable. Pour raisonner sur le comportement asymptotique des algorithmes, il est nécessaire de disposer d'un bagage mathématique proche du programme visé dans ce défi (voir par exemple cet article formalisé avec Isabelle/HOL sur la méthode d'Akra-bazzi).

Un exemple d'impact en milieu industriel est cité par M. Soegtrop (exposé au cours du workshop Coq 2016 à Nancy et communication privée). Il considère que les preuves formelles sur les systèmes temps réels, plus spécifiquement les modems produits dans sa division (au moment de l'exposé chez Intel, mais maintenant chez un autre industriel) aideront à détecter des problèmes qui coûtent des millions à trouver par des méthodes de test traditionnelles. La formalisation de systèmes physiques requiert des mathématiques de l'ingénieur : calcul infinitésimal, analyse, algèbre linéaire, etc. Disposer du matériel nécessaire pour enseigner les mathématiques à l'aide d'outils de preuve aidera également à la dissémination des techniques de preuve formelle en milieu industriel.

Stratégiquement, il y a un risque visible : Lean remporte actuellement de très beaux succès sur le plan de la formalisation des mathématiques et les outils de preuves et travaux développés à l'Inria pourraient être éclipsés par le succès de Lean. C'est pourquoi de nombreux chercheurs dans l'institut estiment qu'il est important de réunir nos forces pour obtenir des résultats dans ce domaine, et de le faire maintenant.

Dans cette compétition, Coq a plusieurs atouts. Un premier atout est la taille de la communauté d'utilisateurs. Cette communauté est bien plus grande pour Coq, travaillant sur des sujets très variés au delà des mathématiques. Pour certains mathématiciens, cela donne l'opportunité d'exploiter les résultats formalisés dans du logiciel implémentant les algorithmes (en se rapprochant de "systèmes de calcul formel certifiés", voir le projet ERC FRESCO) ou avec des simulations numériques garanties (voir les résultats des projets CERPAN, FAUST, et MILC). Un deuxième atout est la stabilité logicielle. Coq a démontré dans le passé sa capacité à permettre le développement de bibliothèques ambitieuses en fournissant des aides pour les questions de maintenance sur le long terme (voir par exemple la progression continue de la bibliothèque Mathematical Components depuis une quinzaine

d'années). Un troisième atout provient de la quantité de mathématiques déjà formalisées : en analyse avec Coquelicot et MathComp Analysis, en algèbre avec Mathematical Components. Ce matériel va permettre de démarrer de nouveaux travaux rapidement de façon compétitive avec la bibliothèque MathLib de Lean.

Approche scientifique & organisation

Le sujet de ce défi est d'adapter un ensemble d'outils à un nouveau public. Ce travail demande une phase de conception préliminaire où de nombreuses solutions devraient être expérimentées et évaluées collectivement avant de lancer des implémentations qui représenteront des investissements lourds. Pour cette raison, nous proposons de constituer plusieurs groupes de travail et de favoriser le travail collectif. Ces groupes de travail s'articuleront autour de sept thèmes.

1. Collaborations avec des enseignants en mathématiques
2. Fondements de la théorie des types
3. Primitives de structuration et d'héritage
4. Notations extensibles et langage de surface
5. Automatisation et calcul mathématique
6. Environnements de travail interactifs
7. Contenus éducatifs sur des domaines précis

Il est également important que ces différents groupes de travail communiquent entre eux pour garantir que les choix faits sur chacune des tâches soient exploités au mieux pour le résultat final. Puisqu'il s'agit de s'adresser à un public nouveau, la diversité de points de vue est importante pour prévoir les difficultés de ce public. La question de développer des contenus éducatifs devra de toutes façons être évaluée par une confrontation au public visé et il est indispensable que toutes les décisions de conception soient étudiées sous cet angle. Nous organiserons donc également une tâche de coordination générale, reposant sur des rencontres régulières de tous les membres du défi et la préparation de démonstrations vivantes.

Détail sur la recherche pour une sélection de sujets

Tâche 1 : Collaborations avec des enseignants

Nous chercherons à établir des contacts avec des enseignants en mathématique dans les différentes universités où nos chercheurs se trouvent. Nous prendrons également contact avec des chercheurs en didactique, surtout pour les mathématiques. Il y a déjà eu quelques expériences dans le passé (e.g., utilisation de Coq à l'université de Nice, sous l'égide de A. Hirschowitz et L. Pottier), et une expérience récente de M. Mayero avec des enseignants du laboratoire LAGA de Paris-Nord (à la rentrée à l'automne 2021).

Les chercheurs *a priori* intéressés par ce sujet sont Hugo Herbelin, Marie Kerjean, Micaela Mayero, et Pierre Rousselin (professeur agrégé à l'Université de Paris Nord), et Yves Bertot. Yves Bertot mènera des actions pour impliquer d'autres chercheurs dans cette tâche.

Tâche 2 : Fondements de la théorie des types

Nous voulons rester dans le cadre de la théorie des types, pour conserver la proximité avec Coq, mais cette théorie repose sur l'amalgame entre plusieurs phénomènes habituellement distincts pour les mathématiciens (e.g., preuves et programmes, implication et types de fonctions), ce qui peut entraîner des confusions chez les débutants. Il s'agira donc de comprendre comment modifier le langage pour rétablir les distinctions nécessaires à l'enseignement.

Une caractéristique importante et manquante dans Coq par rapport à Lean, du point de vue d'un mathématicien professionnel est l'absence de types quotients [1]. En effet, l'utilisation de types quotients est omniprésente en mathématiques et il est donc nécessaire de disposer d'une notion primitive pour les gérer.

La définition des types quotients dans Lean est ad-hoc et brise des propriétés fondamentales importantes (comme *subject reduction*) mais les utilisateurs semblent la préférer à l'absence de types quotients natifs, comme c'est le cas dans Coq.

Nous proposons de définir de manière native dans Coq une notion plus raisonnée de l'égalité satisfaisant la non-pertinence de la preuve, admettant les types quotients et satisfaisant l'extensionnalité fonctionnelle. Plusieurs idées émergent pour cette étude, nous pouvons citer la théorie des types observationnelle et le modèle setoïde de la théorie des types. Les progrès récents dans la compréhension des sortes en Coq permettent une réévaluation de ces approches.

Cependant, l'ajout des principes de base n'est pas toujours suffisant pour rendre un système de preuve confortable : tous les outils automatiques doivent également tirer parti de l'expressivité ajoutée. Par exemple, si l'extensionnalité est fournie, la réécriture sous les lieurs doit être naturellement fournie aux utilisateurs.

Les chercheurs intéressés par ce sujet sont Nicolas Tabareau, Matthieu Sozeau et Hugo Herbelin.

Moyens demandés : Ce travail de haut niveau pourra donner lieu à un travail pris en charge par un post-doctorant sur 2 ans.

Tâche 3 : Structures, inférence et hiérarchies

Les enseignants et chercheurs qui travailleront à la création de contenu abordable pour les étudiants seront très vite confrontés à l'existence de nombreuses définitions et de nombreux théorèmes dupliqués entre différents contextes d'utilisation. Ce problème est connu à la fois dans la tradition mathématique sur papier et dans la tradition des language de programmation purement fonctionnels. Dans le premier cas, la solution apportée est celle

1. Stricto sensu, les quotients sont possibles dans Coq, tout simplement en supposant leur existence, comme on utiliserait un nouvel axiome. Mais cette approche axiomatique est trop faible pour l'usage.

de structures algebriques (monoides, groupes, anneaux, modules, corps, espaces topologiques, vectoriels, etc, mais aussi catégories, foncteurs, etc), et dans le second cas la solution est basée sur des méthodes d'inférences telles que les *classe de type* ou des variations telles que les *structure canonique* ou les *unification hints*.

Les approches dans la tradition des languages de programmation ne permettent pas de reproduire directement et fidèlement la pratique mathématique informelle. En effet cette dernière propose souvent plusieurs caractérisations equivalentes pour la même structure et passe d'une structure à une plus faible sans parfois vérifier que les différentes façons possibles de le faire sont équivalentes, ce qui pose des problèmes notables, tels que ceux décrits dans Hierarchy Builder: algebraic hierarchies made easy in Coq with Elpi.

Des travaux récents ont proposé un nouveau langage spécialisé pour décrire les hiérarchies à un meilleur niveau d'abstraction, masquant les détails d'implémentation. Cette approche est décrite dans l'article Competing inheritance paths in dependent type theory: a case study in functional analysis. Nous voulons améliorer ce type de langage spécialisé pour le mettre entre les mains des concepteurs de contenu (professeurs de mathématiques ou développeurs Coq).

En effet, il reste des problèmes de conception pour un tel langage spécialisé : en simplifier encore l'usage en proposant des syntaxes encore plus minimalistes mais qui génèrent encore plus de théorèmes et d'aides à l'inférence, intégrer ce language à Coq plutôt que de le compiler vers du code Coq afin d'améliorer la robustesse et pouvoir ajouter des fonctionalités, et aussi améliorer certains problèmes de performances.

Nous pourrons évaluer les progrès effectués grâce à cette tâche en lien avec la tâche 1, par exemple en se concentrant sur un cours ou intervient naturellement un partage de théorie entre des objets de types différents (par exemple un cours d'algèbre linéaire, où l'addition entre nombres pris dans un corps cohabite avec l'addition entre vecteurs).

Les chercheurs intéressés pour travailler sur ce sujet sont Cyril Cohen, Enrico Tassi, et Matthieu Sozeau.

Moyens demandés : Ce travail sera effectué par un post-doctorant sur 2 ans, sous la responsabilité des trois chercheurs mentionnés ci-dessus.

Tâche 4 : Notations extensibles et langage de surface

Pour faire des mathématiques efficacement sur ordinateur, il faut s'adapter aux notations en usage. Les systèmes de preuves héritent du domaine de l'informatique une structuration du traitement des entrées en plusieurs phases : analyse lexicale, analyse syntaxique, calcul des types, etc. À cause de cette structuration, l'influence des informations de typage sur l'analyse des formules n'apparaît que tardivement dans le processus. Cette structuration s'adapte mal au mode de fonctionnement de l'esprit humain, très enclin à faire intervenir des informations sémantiques (comme les informations de type) dès les premières phases d'analyse d'un texte.

Nous voulons expérimenter avec de nouveaux mécanismes de notations facilitant l'utilisation d'informations de typage plus tôt dans le processus. Plusieurs questions doivent être traitées : comment les nouvelles notations doivent-elles être spécifiées, quels sont les risques d'ambiguïté, quels algorithmes utiliser pour limiter l'impact sur le temps d'analyse

des formules ?

En complément de ce travail, des éléments d'environnement de travail devront probablement aussi être envisagés : documenter les notations utilisées, activer ou désactiver les notations de façon locale, permettre un affichage bi-dimensionnel.

Les notations jouent un rôle à l'échelle du symbole et de la formule mathématique. Un cours ou un exposé mathématique comporte d'autres éléments : des introductions, des commentaires, des exercices et des expérimentations. Pour l'instant, la structuration des fichiers Coq suit de façon naïve l'influence imposée par le mode d'interaction sous forme de boucle « read-eval-print », où chaque commande de quelques lignes est saisie et lue, puis analysée, puis un résultat est produit (et ce résultat est destiné à être affiché en dehors de l'espace de travail). Il y a de nombreuses opportunités pour revisiter ce mode de fonctionnement trop séquentiel. Par exemple, chaque partie du document peut contenir des éléments vérifiables incrémentalement, certaines parties du document peuvent être complétées par le système de preuve, et les résultats d'interactions peuvent servir pour obtenir un document plus complet. Mais ceci pose de nouveaux défis en milieu éducatif, car un document final vérifié incrémentalement indique moins clairement dans quel ordre l'utilisateur a fourni chaque élément.

Enfin, le fonctionnement actuel des outils de preuve repose souvent sur deux modes : un mode définition et un mode preuve. Pendant le mode preuve, l'approche la plus fréquente est d'utiliser des *tactiques* pour modifier l'état courant de la formule à prouver. Le langage des tactiques proposé à l'utilisateur méritera une attention particulière, de façon à proposer aux débutants une combinaison équilibrée entre procédures élémentaires, procédures avancées, facilité d'apprentissage, et quantité d'informations à maîtriser.

Au delà des modes définition et preuve, il est important de permettre des modes d'interaction plus riches, où l'énoncé prouvé est le résultat d'un calcul ou d'une recherche. Par exemple, on doit pouvoir demander « calcule une approximation à 2^{-128} près de π^2 ». Le résultat devrait être un théorème qui fournit le bon intervalle, sans que l'utilisateur ait eu besoin de fournir les bornes rationnelles de cet intervalle à l'avance. Il devrait également être envisageable d'interagir avec des représentations graphiques des objets mathématiques.

Les recherches décrites dans cette section sont très reliées à l'*expérience utilisateur*. Nous évaluerons cette tâche dans le cadre des collaborations mises en place dans la tâche 1.

Les chercheurs intéressés par ce sujet sont Arthur Charguéraud, Martin Bodin, Emilio Gallego, et Hugo Herbelin.

Moyens demandés : Après les expériences préliminaires menées par les chercheurs, ce travail pourra nécessiter une implémentation prise en charge par un ingénieur sur 2 ans. La mise en place des collaborations et des moyens d'évaluation pour les modifications du langage de surface sera le travail d'un post-doctorant sur 2 deux ans.

Tâche 5 : Traitements automatiques

L'automatisation des raisonnements interfère avec l'apprentissage des mathématiques de diverses manières. Au stade initial, l'apprentissage doit passer par l'acquisition d'automatismes par l'étudiant. Dans cette phase, il est nécessaire que l'ordinateur fournisse très peu d'automatisation pour ne pas interférer avec cette acquisition.

Au fur et à mesure que les techniques de base sont acquises, le système de preuve doit fournir de plus en plus d'automatisation pour que les étapes élémentaires pour l'étudiant de plus en plus confirmé soient également traitées aisément dans le dialogue avec l'ordinateur.

Dans l'état actuel, les systèmes de preuve fournissent des procédures de preuve automatique pour certains fragments des mathématiques bien délimités : les égalités entre formules polynomiales (tactique `ring`), les comparaisons entre formules linéaires et affines (tactique `lia`), ou le calcul d'une fonction dérivée (tactique `autoderive`, dans la bibliothèque Coquelicot). Il existe également des passerelles de collaboration entre Coq et des outils automatiques, comme SMTCoq ou sniper. Même lorsque ces domaines sont compris par l'étudiant, l'intégration dans la pratique et le processus éducatif peut être entravée par la qualité des messages d'erreurs, parce que l'échec d'une procédure automatique peut avoir plusieurs causes à traiter séparément : la formule espérée peut être grossièrement fausse, cette formule peut être en dehors du champ d'application de l'outil automatique invoqué, ou encore la formule peut être sous-spécifiée (juste seulement dans certains cas il importe de préciser par l'ajout d'hypothèses).

L'utilisation d'outils automatiques extérieurs à Coq demande un travail d'aptation pour faciliter l'alignement entre les symboles des bibliothèques mathématiques en Coq et les concepts utilisés dans les systèmes de preuves. Il faut également prévoir la mémorisation dans des caches du context commun à de nombreuses preuves, pour éviter des temps de recalcul. La question d'alignement peut amener des questions nouvelles, parce que les concepts utilisés dans certains énoncés structures canoniques ou des classes de types, alors que les outils de preuve automatique manipulent essentiellement des concepts au premier ordre. Pour ces aspects techniques d'adaptation et de mémorisation nous prévoyons le besoin d'un ingénieur.

Pour cette tâche dont nous prévoyons l'importance, nous avons contacté les chercheurs spécialistes de l'automatisation autour de Coq, par exemple Frédéric Besson ou Chantal Keller, mais ils ont d'autres objectifs à court terme. Arthur Chraguéraud a déclaré son intérêt et propose de nouveaux contacts. Yves Bertot continuera à prospecter les chercheurs potentiellement intéressés.

Moyens demandés : Le travail d'exploration à faire sur ce sujet pourra reposer sur un doctorant. La tâche d'adaptation d'outils automatiques nécessitera le travail d'un ingénieur sur 2 ans.

Tâche 6 : Environnements interactifs

L'une des barrières à l'adoption de nouveaux outils informatiques est la difficulté pour le public visé de disposer d'un environnement de travail fonctionnel et relativement uniforme entre les différents membres du public. Pour résoudre cette difficulté nous comptons concentrer nos efforts sur deux approches. La première consiste à proposer une configuration pour un outil déjà largement adopté (l'outil Visual Studio Code, proposé en open source par Microsoft) et la seconde consiste à proposer une solution entièrement hébergée dans les navigateurs web, en profitant de l'utilisation quasi générale de JavaScript [2].

2. À ce sujet, une étude de l'utilisation de Jupyter est prévue dans le projet ERC FRESCO, porté par A. Mahboubi.

Une partie importante du travail à faire pour cette tâche relève de l'ingénierie, mais les réflexions sur la forme des documents et les différents modes d'interaction mentionnés au point précédent amènent une composante de recherche à ce niveau également.

Pour organiser le travail sur cette tâche, nous monterons des groupes de réflexions sur différents aspects : l'aide à la rédaction (récupérer les informations d'usage pour les fonctions existantes, proposer des patrons de phrases grâce aux informations de typage, recommandations de fonctions en fonction du problème abordé), l'aide à l'exploration (fournir des moyens rapides pour connaître les fonctions existantes, filtrer des recherches dans la base de données), et la mise en valeur des données fournies par le système (affichage graphique des concepts géométriques, customisation de l'affichage, intégration d'hyperliens, *literate programming and proving*, etc). En lien avec la tâche 1, nous prévoyons également dans cette tâche une évaluation empirique des améliorations apportées à l'environnement.

Les chercheurs et ingénieurs intéressés par ce sujet sont Maxime Dénès, Emilio Gallego, Hugo Herbelin, et Laurent Théry.

Moyens demandés : Ce travail nécessitera une implémentation prise en charge par un ingénieur sur 2 ans dans VsCoq. L'étude de l'approche *literate programming and proving*, sera menée par un doctorant.

Tâche 7 : Création de bibliothèque sur des domaines précis

Un moyen de faciliter l'entrée des débutants dans un système de preuve sur ordinateur est de fournir un ensemble de contenus pensés pour une prise en main facile, et en évitant les choix multiples. Il faudrait ainsi disposer d'une collection de bibliothèques thématiques motivée par le besoin d'enseignement. Pour chaque bibliothèque, nous proposons de réunir des spécialistes de formalisation, un ou des enseignants de mathématiques, et des jeunes chercheurs qui seront ainsi formés à l'utilisation de Coq et au travail à l'interface avec les enseignants.

Le travail effectué jusqu'à maintenant a permis l'exploration de chemins divers, mais il manque un parcours balisé pour l'enseignement, où les différents aspects d'une bibliothèque (progression des concepts, uniformité des nommages, complétude des théories pour chaque niveau d'abstraction) sont organisés. Le développement de ces bibliothèques thématiques est un travail d'envergure et représente un risque assez fort si la complétude et l'uniformité ne sont pas obtenues de manière satisfaisante.

Un domaine important des mathématiques concerne l'analyse réelle, et c'est d'ailleurs pour cette raison que Coq dispose déjà de bibliothèques anciennes sur ce sujet. Toutefois, ces bibliothèques souffrent de défauts hérités de choix initiaux dont les conséquences sont apparues avec le temps. Nous proposons de reprendre ce travail et de l'utiliser comme étude de cas.

Sur la base des expériences précédentes, certains choix pour la prochaine bibliothèque sont clairs, alors que d'autres aspects font encore l'objet de réflexions. Ainsi, il est certain que cette bibliothèque d'analyse pour l'enseignement doit reposer sur des principes de logique classique : il faut partir des axiomes connus, axiome du tiers-exclu, axiome du choix, principe d'extensionnalité pour les fonctions. Il faut également exploiter ces axiomes au maximum dans les outils de preuve automatique.

Il serait tentant de décider maintenant le contenu des bibliothèques à formaliser, mais il est important de synchroniser cet partie du travail avec les enseignants.

Un point qui pourra demander plus d'expérimentation concerne le niveau d'abstraction vis-à-vis des espaces à une ou plusieurs dimensions. Considérer les théorèmes sur $\mathbb{R}$ comme des cas particuliers de théorèmes plus généraux sur $\mathbb{R}^n$ permet de faire certaines preuves une seule fois. En revanche, une telle économie présente une difficulté supplémentaire pour le public étudiant, qui doit être capable d'absorber simultanément les concepts spécifiques au théorème et le raisonnement en plusieurs dimensions.

Il faudrait aussi que cette nouvelle bibliothèque puisse être reliée aux autres développements existants, pour permettre à la fois une transition confortable vers des développements plus évolués (tels que Coquelicot, MILC, ou mathcomp-analysis) et une réutilisation efficace de matériel déjà formalisé.

Il faut noter que des expériences d'utilisation en milieu universitaire ont déjà commencé.

Les chercheurs intéressés par ce sujet sont Sylvie Boldo, François Clément, Micaela Mayero, Marie Kerjean, Pierre Rousselin, Jean-Marie Madiot, Yves Bertot, Cyril Cohen.
Moyens demandés : Une partie du travail de recherche sur les bonnes formes de langage à employer pourra être menée dans le cadre d'une thèse par un doctorant. Une partie du travail de recherche sur la façon d'établir des correspondances entre des résultats déjà formalisés dans des cadres différents pourra être menée dans le cadre d'une deuxième thèse par un doctorant. Une fois que les choix de langage seront faits, la majeure partie du travail d'implémentation de mathématiques formalisées pourra faire l'objet du travail d'un doctorant ou d'un ingénieur pour 2 ans.

Récapitulatif pour la demande de moyens

ingénieurs Tâche 4 (2 ans), Tâche 5 (2 ans), Tâche 6 (2 ans), Tâche 7 (2 ans),

doctorants Tâche 5 (1 doctorant), Tâche 6 (1), Tâche 7 (3),

post-doctorants Tâche 2 (1 post-doctorant), Tâche 3 (1), Tâche 4 (1).