Utilisation des assistants de preuves pour l'enseignement en L1

Marie Kerjean, Frédéric Le Roux, Patrick Massot, Micaela Mayero, Zoé Mesnil, Simon Modeste, Julien Narboux, Pierre Rousselin

Utilisation des assistants de preuves pour l'enseignement en L1
Retours d'expériences

Nous rendons compte de cinq expériences récentes de l'enseignement de la démonstration utilisant les assistants de preuve Coq, DƎ∀DUCTION, Edukera et Lean.

- M. Kerjean
- F. Le Roux
- P. Massot
- M. Mayero
- Z. Mesnil
- S. Modeste
- J. Narboux
- P. Rousselin

Table des matières

1. Introduction

Les assistants de preuve, comme Coq, Isabelle, Lean, HOL-Light, sont des logiciels qui permettent de vérifier mécaniquement une démonstration mathématique. Depuis quelques années une communauté croissante de mathématiciens et mathématiciennes utilise ou s'intéresse à cet outil, comme Tom Hales et Vladimir Voïevodski [1] :

> Today we are looking forward to the time when proof assistants will become an everyday tool of a working mathematician.

Pour une introduction à l'utilisation des assistants de preuve en mathématiques voir par exemple les articles de Tom Hales [17], Jeremy Avigad et John Harrison [2] et Assia Mahboubi [21].

Naturellement les personnes qui font de la recherche dans le domaine des assistants de preuve ont utilisé cet outil dans leurs enseignements. Les assistant de preuve sont actuellement utilisés de manière courante par la communauté qui s'intéresse aux fondements des langages de programmation, à la fois pour vérifier des résultats de recherche et pour l'enseignement, notamment la série de livres par B. Pierce et. al. qui repose sur Coq [23]. D'autre part, les assistants de preuve sont aussi utilisés dans le cadre de cours de Logique, nombreux sont les outils permettant de construire des démonstrations en déduction naturelle pour la logique propositionnelle ou du premier ordre [4, 6, 8, 20, 9, 19, 16, 7]. Les assistants de preuve peuvent aussi être utilisés pour formaliser la méta-théorie (la logique comme objet d'étude) [14, 31, 27].

L'utilisation d'assistants de preuve pour enseigner la démonstration et les mathématiques est plus rare. Les premières expérimentations en France datent du début des années 2000 [25, 5]. Donnons ici deux arguments pour leur pertinence, nous y reviendrons plus longuement en conclusion. Tout d'abord, plusieurs outils numériques sont déjà intégrés de façon probante à l'enseignement des mathématiques (logiciels de géométrie dynamique, de calcul formel, exerciseurs…), un de leurs avantages, que possèdent donc également les assistants de preuve, est la possibilité de rétroactions fournies par la machine, permettant plus d'autonomie dans le travail des étudiants et étudiantes (ce qui ne veut pas dire bien sûr qu'il suffit de mettre entre leurs mains un assistant de preuve pour qu'ils et elles apprennent à produire des preuves !). En-

1. https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/2014_11_16_Kuwait.pdf, page 18

Fonctionnement des assistants de preuves

Les deux assistants de preuve utilisés ici, Coq et Lean, fonctionnent de façon très similaire. On avance dans la preuve pas à pas ; à chaque étape, ou *état de preuve*, sont affichés le *contexte*, c'est-à-dire la liste des objets et des propriétés actuellement disponibles, et le *but*, la propriété qu'il reste à prouver. On fait évoluer l'état de preuve au moyen de *tactiques* qui transforment le contexte et/ou le but. Si par exemple le but est une propriété universelle du type $\forall x, P(x)$, la tactique `intro` permet d'ajouter un nouvel objet dans le contexte ; si cet objet est nommé y, le but devient $P(y)$. On voit que cette tactique d'introduction ne fait qu'appliquer une règle syntaxique élémentaire ; mais il existe aussi des tactiques beaucoup plus sophistiquées qui prennent en charge un morceau de preuve plus conséquent. Par exemple, Coq ou Lean fournissent des tactiques qui permettent de prouver automatiquement une égalité qui découle linéairement des égalités présentes dans le contexte. Dans les interfaces graphiques, l'utilisation des tactiques est remplacée par des clics sur des boutons.
On peut ainsi décrire la démonstration en indiquant les règles logiques utilisées, à charge pour l'assistant de preuve de reconstruire les énoncés intermédiaires que l'on déduit : on dit que le langage est impératif. Il est possible aussi de décrire les démonstrations, d'une manière qu'on appelle déclarative, en donnant la liste des énoncés intermédiaires accompagnés de leur justification. Des assistants de preuve comme Mizar ou Isabelle/Isar sont majoritairement utilisés en mode déclaratif. Les preuves déclaratives tendent à refléter plus fidèlement les preuves papier et elles sont généralement plus lisibles isolément que les scripts de preuves impératives. Cette distinction est surtout une question de choix de l'utilisateur, car des modes déclaratifs sont disponibles aussi pour Coq [11] et Lean, et Isabelle permet aussi de décrire des démonstrations à l'aide uniquement de tactiques. La bibliothèque Lean-verbose, proposée par Patrick Massot (voir plus bas), permet entre autres choses d'encourager un style déclaratif pour Lean. Des outils permettent aussi de présenter les énoncés intermédiaires d'une preuve impérative [24].

suite, les difficultés de compréhension et d'usage d'un langage au moins partiellement formel, les difficultés de structuration d'une preuve sont des obstacles bien connus de l'apprentissage des mathématiques à l'entrée dans l'enseignement supérieur notamment, et l'utilisation d'assistants de preuve force à travailler particulièrement ces deux points (là encore, avec des médiations nécessaires pour relier la production de preuve sur l'ordinateur et en papier-crayon).

Récemment les expériences se multiplient un peu partout, par exemple celles menées par Jeremy Avigad [1], Kevin Buzzard [2], ou celles évoquées dans cet article. Celles et ceux qui les mettent en place sont plutôt enthousiastes quant à l'impact de l'utilisation des assistants de preuve sur l'apprentissage de la preuve, mais les recherches sur cette question sont encore balbutiantes. Des éléments d'une analyse a priori de l'impact didactique de différents aspects des assistants de preuve ont été proposés [3], ainsi qu'une étude sur les démonstrations produites par des étudiants et étudiantes ayant utilisé Lean [30].

Un groupe de chercheurs et chercheuses en mathématiques, informatique et didactique se réunit depuis quelques années pour réfléchir à différentes questions que posent ces expérimentations. Une journée d'échanges a été organisée en mars 2022 [3], dont sont issus les témoignages qui suivent. Il s'agit de cinq expériences françaises d'enseignement de la démonstration à l'arrivée à l'université, à l'aide de quatre outils différents : Coq, Lean, DE∀DUCTION, et Edukera. Les deux premiers sont des assistants de preuve en ligne de commande, tels qu'utilisés par les professionnels, quoique Lean soit ici muni d'une surcouche qui rend la syntaxe proche du langage naturel ; les deux derniers sont des interfaces graphiques pour Lean et Coq, du type "preuve par clics".

2. `https://www.ma.imperial.ac.uk/~buzzard/xena/natural_number_game/`
3. En plus des auteur·es de cet article, étaient présent·es à cette journée Evmorfia Bartzia, Faïza Chellougui, Viviane Durand-Guerrier, Antoine Meyer, qui font également partie du groupe de travail mentionné, et Antoine Chambert-Loir et Frédéric Tran Minh.

2. Cinq retours d'expériences

2.1 – Université Paris 13

Marie Kerjean, Micaela Mayero, Pierre Rousselin.

Contexte et discussions préliminaires

La double-licence (DL) mathématiques et informatique de l'institut Galilée (université Paris XIII dénommée Paris Sorbonne-Paris-Nord) est une filière sélective qui recrute des étudiant·e·s dont l'objectif est de décrocher une licence d'informatique et une licence de mathématiques en trois ans. Pour l'année 2021–2022, elle comprenait en première année une cinquantaine d'étudiant·e·s suivant des cours des licences d'informatique et de mathématiques. Ce cours d'*Initiation aux preuves formelles* est nouveau (année 2021–2022). Il est spécifique aux étudiant·e·s de DL et dure 19,5h (6 × 3h de travaux pratiques en salle machine et 1,5h d'évaluation en salle machine également) et a lieu au premier semestre de L1.

Son objectif principal est d'amener les étudiant·e·s à l'écriture de preuves mathématiques rigoureuses en utilisant l'assistant de preuves Coq. Nous voulions qu'ils et elles acquièrent *par la pratique* des réflexes de preuve (introduction de variables et d'hypothèses, application de théorèmes, réécritures, récurrence), tout en évitant certains pièges habituels, comme le fait qu'on ne peut faire commuter un connecteur universel et un connecteur existentiel. Par ailleurs, avant cette année, la double licence n'avait pas de cours spécifique en L1. Il s'agissait donc également de proposer un module « de bienvenue » ayant un contenu qui relève à la fois des mathématiques et de l'informatique, ce qui devait aider à la formation d'un groupe solidaire en double licence. Enfin, ce module devait permettre, à la marge, de renforcer certaines compétences pratiques en informatique : écriture dans un fichier texte, respect de la syntaxe, bonnes pratiques d'indentation, de nommage, ...

Les discussions préliminaires à la mise en place du cours ont révélé certaines réserves, bien compréhensibles, de la part des enseignant·e ·s mathématicien·ne·s. Il y avait d'abord la crainte de confusions entre les notations utilisées en Coq et les notations usuelles des mathématiques. Par exemple, en Coq, la flèche $\to$ est utilisée à la fois entre les ensembles de départ et d'arrivée d'une fonction et pour l'implication (au lieu du symbole $\Longrightarrow$). Ensuite, les fondations « usuelles » des mathématiques (disons la théorie des ensembles au moins à un niveau intuitif) diffèrent de celles de Coq (la théorie des types dépendants), ce qui entraîne des différences importantes qu'on ne peut pas toujours mettre sous le tapis. Ainsi, plutôt que de parler, par exemple, de l'ensemble des réels positifs, qui est un ensemble au même titre que $\mathbb{R}$, on considère le prédicat « être positif » qui n'est donc plus de même nature que le type des réels. Cela peut avoir des conséquences troublantes sur les choix de formalisation des mathématiques. En particulier, en Coq, il est beaucoup plus pratique que les fonctions soient totales (pas de domaine de définition $\mathbb{R}^*$ par exemple, en Coq la fonction inverse est définie sur $\mathbb{R}$ tout entier). Suite à ces discussions nous avons prêté une attention particulière aux explications dédiées aux notations utilisées dans Coq et aux différences, lorsque nous les avons rencontrées, entre la pratique habituelle des mathématiques et celle de ce module.

Pour ce qui est du format (et en partie, du contenu) du cours, nous nous sommes inspirés du programme *software foundations* et plus particulièrement de la partie *logical foundations* [22] en fournissant des fichiers Coq conçus pour être lus sur machine et complétés par les étudiant·e·s. Il fallait leur permettre d'être actifs le plus rapidement et le plus fréquemment possible, il n'y avait aucune heure de cours magistral allouée à ce module.

Contenu et déroulement du cours

Les polycopiés d'algèbre et d'analyse du premier semestre de L1 nous ont servi de guide pour le choix du contenu. Notre ambition (qui s'est avérée déraisonnable) était d'aller vers certaines preuves d'analyse sur les suites numériques.

La progression, presque imposée par notre envie de faire écrire des preuves « ε, δ », est la suivante :

1. Logique propositionnelle, déduction naturelle
2. Entiers naturels et récurrence
3. Quantificateurs
4. Ensembles
5. Nombres Réels
6. Suites numériques

Il est important de rappeler que les étudiant·e·s de L1 arrivant en septembre ont peu de connaissances structurées sur les notions de logique. Nous avons donc commencé par enseigner, par la pratique sous Coq, les règles usuelles de raisonnement.

Celles-ci pouvaient être écrites au tableau sous forme de règles dans un des formalismes usuels en logique tels que la déduction naturelle ou le calcul des séquents, présentés informellement, en lien avec les tactiques correspondantes de Coq. De plus, pour apprendre la notion de démonstration, ces règles de déduction semblent bien plus utiles que les tables de vérité, finalement très éloignées de la pratique courante des mathématiques.

Chaque fois qu'une tactique doit être utilisée (voir encadré plus haut), un exemple à suivre est donné. Les plus subtils ou importants sont montrés et commentés par l'enseignant·e à l'aide d'un vidéo-projecteur.

La figure 1 est la toute première preuve présentée aux étudiant·e·s, telle qu'ils la voyaient dans CoqIDE. Il s'agit de prouver la formule $\forall P : \mathrm{Prop},\ P \implies P$. La partie en haut à droite de l'écran montre la progression de la preuve sous la forme d'un contexte (type des variables et hypothèses) au-dessus de la ligne et d'un but, en-dessous, qui évoluent tout au long de la preuve. La première tactique, `intros P`, introduit la variable P de type Prop dans le contexte, ce qui supprime le quantificateur $\forall$ du but (introduction du quantificateur $\forall$ en déduction naturelle). La seconde tactique `intros HP` place la prémisse de l'implication dans les hypothèses en nommant cette hypothèse `HP` (introduction de l'implication en déduction naturelle). Enfin, la tactique `assumption` affirme que le but fait partie des hypothèses et demande à Coq de le vérifier, ce qui ici met fin à la preuve.

Le cours aborde ensuite les entiers naturels. Ceux-ci sont définis en Coq par récurrence avec la constante 0 et la fonction successeur. Toutes les opérations usuelles sont définies par récurrence, et les preuves doivent donc suivre le même schéma. Le système Coq est particulièrement adapté à l'enseignement de ces preuves : la tactique `induction` fait clairement apparaître les deux sous-buts (la preuve au rang 0 et l'hérédité) et l'hypothèse de récurrence. Suffisamment bien guidé·e·s, les étudiant·e·s peuvent aller jusqu'à prouver la commutativité de la multiplication en partant de rien.

À ce stade, nous en étions à mi-parcours, et nous avons soumis les étudiant·e·s à un petit test noté d'une demi-heure sur machine. Les résultats de ce test nous ont vraiment encouragé. Les étudiant·e·s savaient, en grande majorité, prouver des théorèmes simples sur le calcul propositionnel ou les entiers naturels.

Les séances suivantes sur les quantificateurs et les ensembles (en fait, en Coq, le calcul des prédicats) se passent moins bien. Les écarts se creusent de façon inquiétante entre les étudiant·e·s et certaines notions mathématiques, nouvelles ou très fraîches, nécessitent un temps d'assimilation plus important.

La partie suivante, sur les réels, s'est un peu mieux passée. Contrairement aux entiers naturels, dont la définition est constructive, les réels sont définis [4] de façon axiomatique, comme corps totalement ordonné, complet et archimédien. Le sujet de travaux pratiques sur les réels introduit petit à petit ces axiomes et demande aux étudiant·e·s de prouver certains théorèmes (par exemple les identités remarquables, ou la stricte décroissante de la fonction $x \mapsto -x$), en ne partant que des axiomes.

Lors de la dernière séance, nous avons commencé l'analyse des suites numériques. Les définitions sont celles d'un cours de mathématiques classique. C'est à ce moment-là que nous nous autorisons à montrer aux étudiant·e·s des tactiques plus puissantes comme `lra` (pour *Linear real and rational arithmetic*) qui permet de prouver automatiquement certaines égalités et inégalités simples. Malheureusement, bien peu d'étudiant·e·s étaient prêts à gravir cette dernière montagne avec nous. Le rythme était certainement trop soutenu et nous avons manqué de temps. La figure 2 présente un extrait du fichier Coq d'analyse réelle. Le théorème énoncé porte sur la somme de deux suites convergentes. À ce niveau, les preuves commencent à être difficiles à écrire. Celle du corrigé, qui reste au niveau du module, utilise une bonne vingtaine de tactiques. Il faudrait donc peut-être réfléchir aussi à des outils techniques et pédagogiques pour rendre ces preuves plus simples à écrire.

Bilan

Nous avons dû faire certains choix lors de l'élaboration des sujets. Les preuves sont longtemps restées très détaillées, sans tactique automatique ou presque. Ce choix était guidé par l'envie d'enseigner en premier lieu la rigueur dans l'écriture des preuves. Pour la partie d'analyse, au contraire, utiliser de l'automatisation permet de se débarrasser de certains (malheureusement pas tous) « détails » qu'on ne songerait jamais à prouver dans un cours d'analyse mathématiques.

Plus généralement, le choix le plus important

4. La formalisation des réels en Coq est due à M. Mayero, l'une des enseignantes du module.

Figure 1 – Extrait du premier fichier Coq sur les bases de la logique.

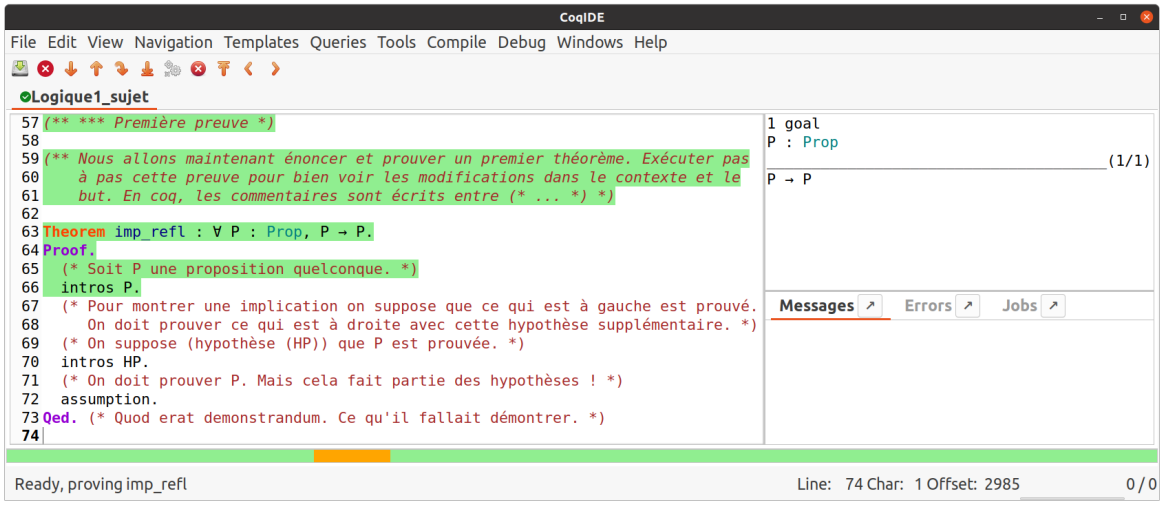
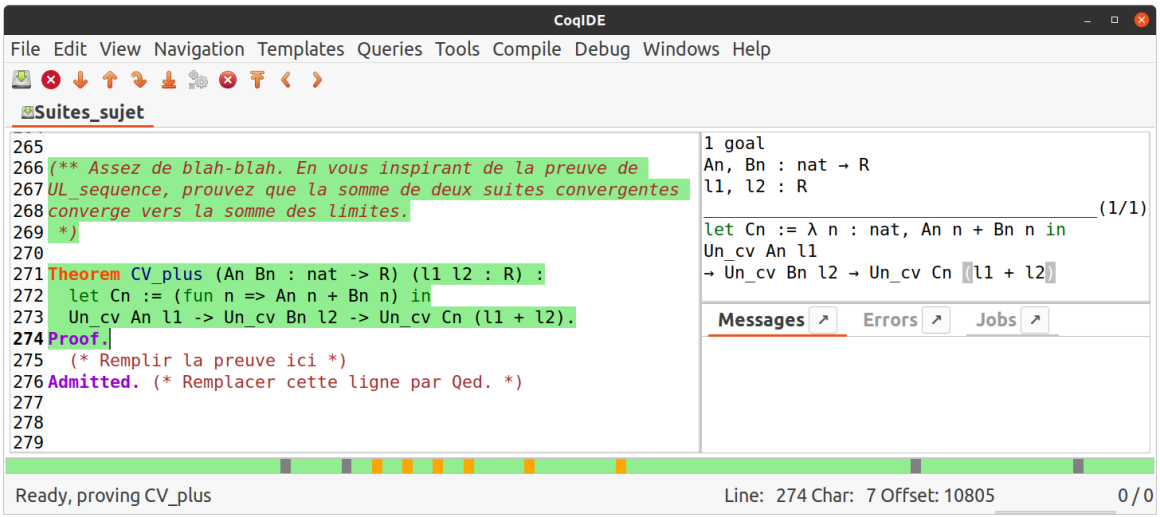


Figure 2 – Extrait du fichier d'analyse réelle.



à faire a été : s'agit-il d'un cours de Coq ou d'un cours de mathématiques avec Coq? Dans le premier cas, il s'agit d'apprendre à utiliser Coq, avec toute sa puissance d'automatisation, pour présenter le problème de la formalisation des mathématiques. Dans le second, qui a été le nôtre, il s'agit d'utiliser Coq pour détailler les preuves mathématiques tout en essayant de rester au plus proche de ce qui peut se faire avec papier-crayon. Et une fois acquis que les propriétés de base d'un corps ou anneau ordonné sont toujours présentes, présenter les tactiques d'automatisation qui nous permettent de nous affranchir de cette lourdeur (comme en mathématiques) et qui rendent Coq plus attirant.

Il était certainement prématuré d'aborder l'analyse réelle avec « ε, δ ». Pour pouvoir le faire dans de bonnes conditions, il aurait fallu que ce cours ait lieu au second semestre avec un volume horaire plus important, ce qui est exclu pour des raisons de moyens et d'organisation.

Malgré ce raté, les retours des étudiant·e·s sont assez positifs, ce qui est encourageant si l'on tient compte du fait qu'il s'agissait d'une première expérience pour nous. Certains étudiant·e·s témoignent

du fait que ce cours les a aidés à comprendre les cours d'algèbre et de logique. Enfin, il apparaît que ce module a bien eu pour effet de souder le groupe, les plaçant d'emblée dans une position inédite et unique qui les poussait à s'entraider.

Tout n'a certes pas été un succès, mais, à ce niveau, les difficultés ne semblent pas tellement dues à des problèmes de notations ou de différences de point de vue entre théorie des ensembles et théorie des types dépendants. Il nous paraît évident que Coq ou en général un assistant de preuve peut être extrêmement utile à l'enseignement des mathématiques et en particulier à l'assimilation de la rigueur. La logique intuitionniste donne à la fois du sens aux connecteurs logiques et une direction naturelle pour l'écriture de la preuve.

2.2 – Faculté des sciences d'Orsay

Patrick Massot

Cours concerné

La licence double diplôme informatique et mathématique de l'université Paris-Saclay à Orsay est une filière sélective accueillant entre cinquante et soixante étudiant·e·s.

Le cours intitulé « Logique et démonstrations assistées par ordinateur » a lieu au second semestre depuis quatre ans. Il est obligatoire (sauf en 2021 pour cause de Covid). Il comporte douze séances de cours-TP de deux heures, en salle machine. L'évaluation comporte une part de contrôle continu constituée de cinq ou six devoirs courts à rédiger à la maison et un examen de trois heures comportant une série d'exercices à faire sur ordinateur puis sur papier. Le logiciel utilisé est Lean, avec, depuis deux ans, une surcouche d'interface maison. Le cours a été créé par Patrick Massot, seul la première année, puis aidé par Frédéric Bourgeois à partir de la deuxième année et Christine Paulin à partir de la quatrième année.

Objectifs et contenu mathématique

L'objectif de ce cours est d'améliorer la compréhension et la rédaction de démonstrations mathématiques sur papier. L'ordinateur n'est qu'un outil intermédiaire. L'objectif n'est pas du tout la virtuosité dans l'utilisation du logiciel. Les questions de logique pure et de fondements des mathématiques ne sont pas non plus un objectif et sont cachées le plus possible par la présentation et le choix des exercices. Comme la plupart des assistants de démonstrations, Lean est basée sur la théorie des types plutôt que sur le duo logique du premier ordre et théorie des ensembles de Zermelo-Frankel. Les quelques différences visibles que cela induit sont présentées comme de simples questions de notations et ne posent *aucun* problème. La conclusion au bout de quatre ans est que chaque minute passée à commenter la différence entre théorie des ensembles et théorie des types dans ce contexte est une minute gâchée.

Tous les connecteurs logiques et quantificateurs sont présentés dans un contexte concret, avec des énoncés portant sur des nombres, des suites de réels ou des fonctions d'une variable réelle à valeurs réelles. Au niveau logique, l'accent est mis sur la manipulation des quantificateurs plutôt que sur le calcul des propositions. Les notions mathématiques utilisées sont presque exclusivement des notions abordées au premier semestre. La seule exception est la notion de suite de Cauchy qui n'est pas vue au premier semestre mais est utilisée par deux exercices pour diversifier les combinaisons de quantificateurs.

Toutes les personnes suivant le cours vont jusqu'à aborder des exercices démontrant des résultats de bases sur les limites de suites de réels (unicité de la limite et théorème des gendarmes par exemple). Les plus rapides vont jusqu'à redémontrer la caractérisation séquentielle de la continuité, le fait qu'une fonction continue sur un segment est bornée et atteint ses bornes et le théorème des valeurs intermédiaires.

Logiciel utilisé

Lean est un logiciel libre développé par Leonardo de Moura de Microsoft Research et par la communauté de ses utilisateurs. Il s'agit d'un logiciel récent puisque la première version date de 2013, mais il s'inspire évidemment de nombreux logiciels plus anciens, en particulier Coq. Il n'est pas spécifiquement destiné à l'enseignement, il est utilisé aussi au niveau recherche. C'est par exemple le logiciel qui a été utilisé en 2019 par Buzzard, Commelin et Massot pour expliquer aux ordinateurs la notion d'espace perfectoïde de Scholze. Il est en ce moment utilisé entre autres pour expliquer des résultats de flexibilité en topologie différentielle incluant notamment l'existence des retournements de la sphère.

Les interactions entre le logiciel et la personne qui l'utilise se font via un éditeur de texte qui peut être VSCode (utilisé dans ce cours), vi ou emacs. L'utilisateur ou utilisatrice tape des commandes dans la partie gauche de l'écran et l'ordinateur répond dans la partie droite en mettant à jour la liste des hypothèses et le but courant, ou en affichant des messages d'erreur le cas échéant.

Dans l'utilisation normale de ce logiciel, les instructions ressemblent beaucoup à du code informatique. Ce code a été utilisé lors des deux premières années du cours. Il ne posait pas de problème lors de la résolution des exercices sur ordinateur mais rendait vraiment difficile le passage à la rédaction sur papier. Durant les deux années suivantes, le cours a utilisé des commandes personnalisées qui sont plus proches du langage naturel.

Le code ci-dessous est la démonstration complète du fait que si une fonction f est continue en un point x_0 alors elle y est séquentiellement continue. La première ligne liste les objets intervenant, la deuxième liste les hypothèses, en les nommant `hu` et `hf`, et la troisième ligne annonce l'objectif. Dans tout le cours, tous les énoncés sont fournis. Les étudiant·e·s n'écrivent que les démonstrations (mais bien sûr une démonstration peut nécessiter d'écrire un petit énoncé intermédiaire). Tout ce qui se trouve entre les lignes `begin` et `end` constitue la démonstration.

```
example (f : ℝ → ℝ) (u : ℕ → ℝ) (x₀ : ℝ)
  (hu : limite_suite u x₀) (hf : continue_en f x₀) :
  limite_suite (f ∘ u) (f x₀) :=
begin
  Montrons que ∀ ε > 0, ∃ N, ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε,
  Soit ε > 0,
  Par hf appliqué à [ε, ε_pos] on obtient (δ : ℝ) tel que
    (δ_pos : δ > 0) (hδf : ∀ x, |x - x₀| ≤ δ → |f x - f x₀| ≤ ε),
  Par hu appliqué à [δ, δ_pos] on obtient (N : ℕ) tel que
    (hNu : ∀ n ≥ N, |u n - x₀| ≤ δ),
  Montrons que N convient : ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε,
  Soit n ≥ N,
  Par hδf il suffit de montrer que |u n - x₀| ≤ δ,
  On conclut par hNu appliqué à [n, n_ge]
end
```

À la fin de chaque ligne, l'ordinateur affiche le contexte, c'est-à-dire la liste des objets fixés et des hypothèses courantes. Par exemple voici l'affichage après la ligne « Soit $\varepsilon > 0$ ».

```
1 but
f: ℝ → ℝ
u: ℕ → ℝ
x₀: ℝ
hu: limite_suite u x₀
hf: continue_en f x₀
ε: ℝ
ε_pos: ε > 0
⊢ ∃ (N : ℕ), ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε
```

La première ligne dit qu'il n'y a qu'un seul but en suspens (ce nombre peut augmenter par exemple quand on démontre une équivalence en montrant les deux implications). Le symbole ⊢ qui débute la dernière ligne annonce l'objectif. La patience inlassable dont fait preuve l'ordinateur pour afficher ces réponses est sans doute sa contribution la plus précieuse, plus encore que son implacable et impartiale rigueur. Il n'y a jamais aucune ambiguïté sur ce qui est fixé et ce qui est quantifié. Lors du passage à la rédaction sur papier, il est facile de naviguer dans la démonstration en revoyant le contexte à chaque étape.

Bilan

Globalement ce cours est considéré comme un grand succès. Il est très gratifiant à enseigner et permet à de nombreuses personnes de progresser. Il n'y a pas eu d'étude didactique sérieuse permettant d'étayer cette impression mais on peut donner un exemple concret en manipulation de quantificateurs. Lors de l'examen en 2022, 68% des copies ont su convaincre l'ordinateur que l'image d'une suite de Cauchy de réels par une fonction uniformément continue est une suite de Cauchy. Puis 58% des copies ont obtenu au moins 75% des points attribués à la rédaction sur papier de cette démonstration. La démonstration est structurellement très proche de l'exemple de la section précédente. Dans les deux cas la difficulté est d'appliquer les hypothèses dans le bon ordre et au bon réel strictement positif. La variante de l'examen fait intervenir un peu plus de quantificateurs. La notion de suite de Cauchy était apparue dans deux exercices pendant le semestre mais la notion de fonction uniformément continue n'était introduite que dans l'examen, sous forme de succession de quantificateurs sans commentaire. Ces notions n'avaient pas été vues dans d'autre cours.

L'utilisation de Lean amène à bien distinguer les trois façons dont chaque connecteur logique ou quantificateur peut intervenir : pour former un

énoncé, pour utiliser un énoncé et pour démontrer un énoncé. Un exemple fréquent de confusion entre formation d'un énoncé et utilisation de cet énoncé se présente avec le quantificateur existentiel. Très souvent au tableau mais aussi par écrit, on se laisse aller à écrire « d'après l'hypothèse …, $\exists \delta > 0, \ldots$ » puis à utiliser δ dans la suite de l'argument. Il s'agit en fait d'une contraction de plusieurs étapes, bien distinguées dans la preuve avec Lean. La version logiquement correcte est « d'après l'hypothèse …, $\exists \delta > 0, \ldots$. *Fixons* un tel δ ». Cette version correcte est un peu lourde mais on peut l'abréger en « l'hypothèse …, fournit $\delta > 0$ tel que … ». L'enjeu est de ne laisser aucune ambiguïté sur le fait qu'on a fixé un δ, même s'il n'y avait pas nécessairement unicité. Plus généralement l'enjeu est de bien distinguer les variables libres et les variables liées (voir aussi les exemples donnés dans la conclusion de ce dossier).

L'exemple de la section précédente peut être relu en se concentrant sur la précision entourant ces questions de variables libres et liées, à la fois dans la syntaxe de la démonstration et dans la réponse de l'ordinateur. On notera que l'affichage permanent du but et du contexte (la liste des objets fixés et des hypothèses) donne un sens très concret aux notions de variables libres et liées. Les variables libres apparaissant dans le but ou dans une ligne du contexte sont libres d'aller se référer à une autre ligne tandis que les variables liées ne le sont pas. Ainsi dans l'exemple le but fait apparaître un ε libre de se référer à l'avant dernière ligne du contexte, à savoir $\varepsilon : \mathbb{R}$, tandis que N est lié par le $\exists$. Ici la technologie aide bien mais il est tout à fait possible sans ordinateur d'être vigilant et vigilante, particulièrement quand nous sommes au tableau, au moins dans les petites classes (niveau L).

Du côté des difficultés rencontrées, on peut noter une légère dissonance cognitive induite par l'utilisation d'un langage naturel contrôlé. Les commandes tapées ressemblent à du langage naturel mais la syntaxe est aussi rigide que dans un langage de programmation. Toujours du côté des commandes, il y a une tension entre la volonté d'offrir un langage expressif et celle de ne pas créer une montagne de commandes à connaître. Bien sûr le cours met à disposition un manuel des commandes et un aide-mémoire qui sont disponibles y compris durant l'examen. Un autre équilibre délicat à trouver est celui de l'automatisation partielle. Idéalement on voudrait que l'ordinateur fasse tout seul les détails qu'on ne demanderait pas sur papier. Cet idéal lui même est très ambigu et l'implémentation pose aussi des difficultés. Il est probable qu'il y aura toujours de la marge de progression dans ce domaine.

2.3 – Sorbonne Université

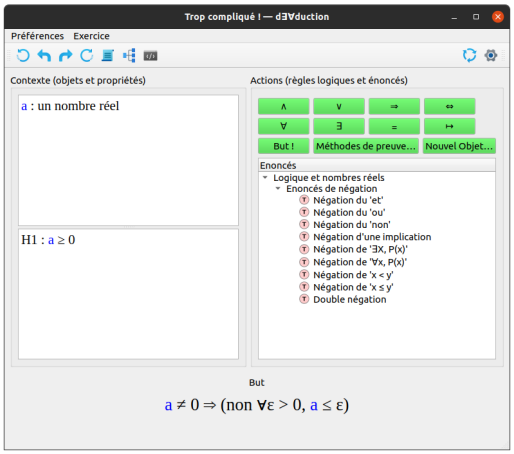
Frédéric Le Roux

Cours concerné

DƎ∀DUCTION est un logiciel d'aide à l'apprentissage de la démonstration, essentiellement une interface graphique pour l'assistant de preuve Lean. Il peut être téléchargé sur la page web de l'auteur (lien dans les signatures en fin d'article). Il a été expérimenté une première fois en juin 2021, à Jussieu, sur 3 séances de 3h hors cursus, avec un groupe d'une dizaine d'étudiant·e·s volontaires de L1 [5]. Une deuxième expérience a eu lieu en 2022, toujours en L1, dans un « atelier de recherche encadrée » concernant 2 groupes de 16 étudiant·e·s, l'un en maths-info et l'autre en maths-physique, représentant au total environ 5% des effectifs du portail "Sciences formelles". L'atelier était organisé en 10 séances de 2h. Les étudiant·e·s suivaient en parallèle un module de mathématiques classique.

5. Cette expérience, menée avec l'aide de Zoé Mesnil et Camille Lichère, est décrite dans le mémoire de M1 de cette dernière, disponible sur https ://perso.imj-prg.fr/frederic-leroux/.

Figure 3 – D∃∀DUCTION : un début de démonstration



Preuve de la propriété $\forall a \geqslant 0, ((\forall \varepsilon > 0, a < \varepsilon) \Rightarrow a = 0)$ après introduction de la variable a par un clic sur le bouton ∀, puis utilisation de la preuve par contraposée.

Le logiciel

La fenêtre principale, représentée sur la figure 3, donne à tout moment un instantané de la preuve en cours. Elle se découpe essentiellement en trois zones :

- à gauche, la zone de *contexte*, divisée en deux : en haut les objets présents à ce stade du raisonnement, et en bas leurs propriétés ;
- en bas, la propriété à démontrer, appelée *but* ;
- à droite, la zone des actions, qui permettent de progresser dans le raisonnement en faisant évoluer le contexte et le but : en haut des boutons avec différents symboles logiques qui permettent d'appliquer les règles syntaxiques correspondantes, en dessous une liste de définitions et d'énoncés qui peuvent également être utilisés.

La figure 4 montre l'utilisation d'une propriété du type ∀…∃…. Précisons que les preuves ne sont pas "pré-enregistrées", l'utilisateur·rice dispose d'une certaine liberté (dans la limite des possibilités de l'interface), et peut procéder par essais et erreurs en revenant sur ses pas chaque fois que nécessaire à l'aide des boutons de navigation de la barre d'outils.

Contenu de la formation

Le but est d'apprendre à concevoir et rédiger des démonstrations mathématiques, et le logiciel n'est qu'un outil. Le choix qui a été fait vis à vis de la logique est d'avoir une attitude pragmatique, on n'introduit aucune connaissance théorique, pas de déduction naturelle ni même de calcul propositionnel.

La première séance est une séance de prise en main, on démontre essentiellement des propriétés « évidentes ». Dans la deuxième séance, on introduit les notions d'image directe et réciproque, et on cherche à comprendre si les formules

$$A = f^{-1}(f(A)), \quad B = f(f^{-1}(B))$$

sont vraies. Les étudiant·e·s peuvent chercher à montrer chaque inclusion (vraie ou fausse) avec le logiciel, puis à dessiner des contre-exemples. La présentation en questions ouvertes et l'utilisation des "patates" pour les contre-exemples permettent d'éviter de s'enfermer dans une démarche purement syntaxique. On explore ensuite les notions d'injectivité et de surjectivité. Une séance est consacrée à la négation. Les deux dernières séances portent sur les limites des suites, sans l'aide du logiciel puisque les exercices correspondant n'y ont pas encore été intégrés [6].

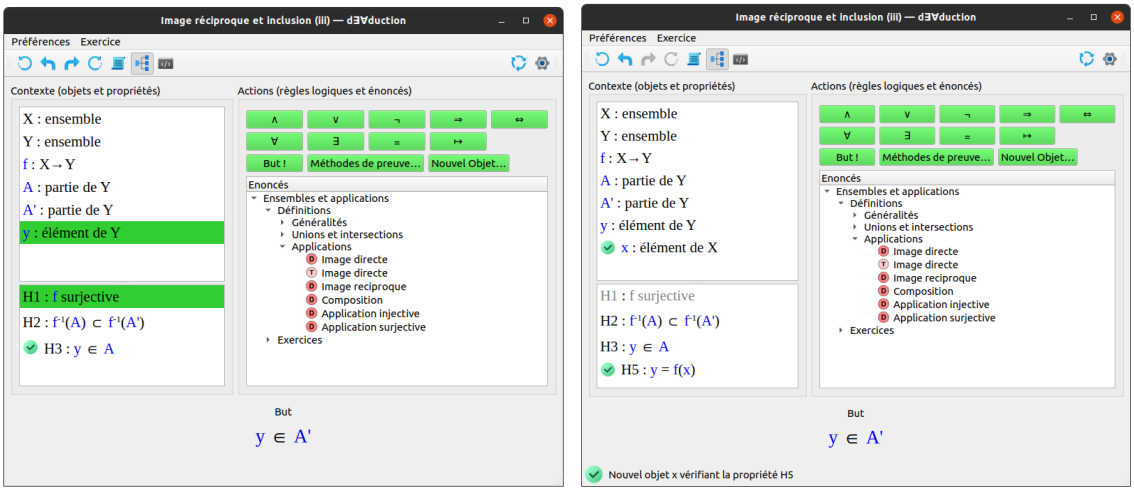
À partir de la troisième séance on commence à convertir les preuves sur ordinateur en preuves rédigées classiques. En particulier, on fait remplir aux étudiant·e·s un tableau indiquant les formulations possibles pour les deux types d'emploi de chacun des boutons logiques (pour démontrer une propriété, ou bien pour utiliser une propriété du contexte). Les étudiant·e·s rendent chaque semaine au moins une démonstration rédigée avec (ou sans) l'aide du logiciel, qui leur est rendue avec des commentaires détaillés.

Bilan provisoire

Les étudiant·e·s s'approprient l'interface dès la première séance, les difficultés qui peuvent apparaitre sont déjà d'ordre logique (notamment dans l'utilisation et la démonstration des disjonctions du type "$x \in A \cup B$"). L'interface peut dérouter un ·e étudiant·e avancé ·e (ou même un ·e professionnel ·le) qui se demande comment faire comprendre à l'ordinateur la preuve qu'il ou elle a en tête ; mais elle

6. Travail en cours…

Figure 4 – DƎ∀DUCTION : utilisation d'un énoncé universel



Une étape de la preuve de la propriété suivante : f surjective $\Rightarrow (\forall A, A' \subset Y, (f^{-1}(A) \subset f^{-1}(A') \Rightarrow A \subset A'))$. Après avoir déroulé les implications et les quantificateurs universels, l'utilisatrice sélectionne l'objet y et la propriété *f surjective* et clique sur le bouton ∀ ; le logiciel introduit le nouvel objet x et la propriété $y = f(x)$. Le paramétrage du logiciel permet de régler le niveau de détail, en obligeant par exemple l'utilisatrice à déplier la définition de surjectivité avant de pouvoir l'appliquer, contrairement à ce qui se passe ici.

rassure les débutant·e·s, qui n'ont pas d'idée préalable pour résoudre l'exercice, et qui acceptent plus facilement de se laisser guider par le formalisme. Après quelques séances, la plupart sont capables de résoudre en quelques minutes, sur logiciel, un exercice comme celui de la figure 4.

Le point clé est le passage de la preuve sur logiciel à une preuve rédigée. Contrairement à ce qu'on aurait pu penser, ce passage n'est pas du tout automatique ! Voici trois types de difficultés qui apparaissent :

– La machine est très pointilleuse, et retranscrire toutes les étapes conduirait à une preuve totalement indigeste. L'étudiant·e ne peut donc pas coller à la machine, et le choix du niveau de détail pose beaucoup de problèmes. Par exemple, à quel moment doit-on faire un rappel du but ?

– Les étudiant·e·s ont du mal à différencier clairement le but des hypothèses du contexte. Ceci est particulièrement problématique lorsqu'on doit réécrire le but (par exemple pour transformer « $x \in f^{-1}(B)$ » en « $f(x) \in B$ »). Comment exprimer clairement que la propriété manipulée est le but et non pas une hypothèse du contexte ?

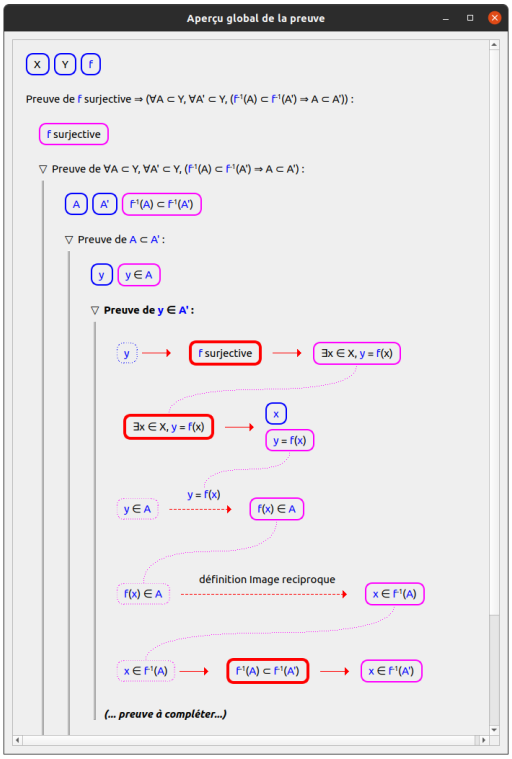
– L'introduction des variables, et notamment le fait de différencier clairement les variables libres du contexte des variables liées pose également problème, surtout lors de l'introduction d'une variable obtenue en appliquant un énoncé existentiel. Sur ce point, on a fait le choix [7] de suggérer fortement aux étudiant·e·s de réserver la formulation "il existe x" pour une variable muette, et d'utiliser "on obtient x" ou "la propriété fournit un x" lorsque la propriété est appliquée et donne naissance à un nouvel objet du contexte.

Malgré ces difficultés, les retours des étudiant·e·s sont très majoritairement positifs. Ils confirment que l'étape de rédaction est essentielle pour les empêcher de se contenter de cliquer sans comprendre, de se laisser guider aveuglément par la machine. Beaucoup notent un effet positif dans les TDs classiques suivis en parallèle ; plusieurs mentionnent un déblocage ("ça m'a rassurée sur mes capacités à comprendre le cours et les démonstrations"), une influence sur leur choix de continuer en licence de maths. Leurs regrets concernent le fait que le contenu soit trop axé sur la théorie des ensembles et pas assez sur l'analyse, contrairement au module principal de maths. Par ailleurs, le logiciel semble moins utile pour les plus avancé·e·s,

7. Choix un peu tradif malheureusement, influencé par l'exposé de Patrick Massot présenté dans ce dossier.

qui préfèrent apprendre à utiliser Lean, et pour celles et ceux qui ont trop de lacunes pour que le logiciel soit bénéfique (environ 2 étudiant·e·s sur 16 pour chacune des deux situations). Nous n'avons pas perçu de différence notable entre le groupe maths-info et le groupe maths-physique.

Figure 5 – DƎ∀DUCTION : aperçu global de la preuve



La fenêtre montre les objets et hypothèses introduites, les déduction et ré-écritures successives tout au long de la démonstration. On y est presque !

Du point de vue de l'enseignant, plusieurs points sont particulièrement satisfaisants. D'une part, on peut laisser les étudiant·e·s travailler principalement en autonomie pendant 2h, et les voir avancer à leurs rythmes, sans se décourager, y compris sur des exercices qui leur sembleraient inaccessibles dans le contexte d'un TD classique. D'autre part, les corrections de preuves au tableau peuvent s'appuyer sur la pratique commune du logiciel, ce qui permet de construire collectivement les preuves avec une assez grande assurance que le discours autour de la démonstration soit à la portée de tous et toutes. L'expérience sera renouvelée l'an prochain, avec une interface améliorée permettant par exemple d'afficher un aperçu global de la preuve, comme sur la figure 5.

Le développement du logiciel DƎ∀DUCTION a démarré en juillet 2020, avec un groupe constitué de l'auteur et de Marguerite Bin, Florian Dupeyron, Antoine Leudière, tous trois étudiants en master, en se basant sur un client Python pour Lean écrit par Patrick Massot. La première séquence d'enseignement a ensuite été mise au point avec l'aide de Camille Lichère, elle aussi étudiante en master, et Zoé Mesnil. La suite du développement sera collective ou ne sera pas... Je cherche à constituer une équipe de collègues intéressé·e·s par le futur du logiciel (développement et utilisation). Si c'est votre cas, contactez-moi !

2.4 – Université de Strasbourg

Pierre Guillot, Julien Narboux

Cours concerné : *Fondements du calcul et du raisonnement*

Le cours *Fondements du calcul et du raisonnement* est un cours de première année/deuxième semestre qui compte pour 3 ECTS. Ce cours est commun à la licence de Mathématiques et la licence d'Informatique. Le public est donc constitué d'étudiant·e·s ayant pour la plupart suivi la spécialité Mathématiques au lycée. Le cours est répété à chaque semestre de printemps et d'automne. Les effectifs sont d'environ 200 étudiant·e·s au printemps et une centaine à l'automne. Parmi eux, une quarantaine suivent le cursus renforcé et sélectif *Cursus Master en Ingénierie*. Le cours existe depuis 2018 et comporte un volume horaire de 24h de cours intégré associé à 4h de TP en salle machines pour la prise en main de l'assistant de preuve.

L'objectif du cours est d'apprendre la structure et le vocabulaire utilisé dans un document mathématique et d'apprendre les règles de déductions et à rédiger une démonstration mathématique. Le polycopié prend le parti de lister explicitement toutes les règles de raisonnements nécessaires pour une démonstration en s'inspirant de la déduction naturelle, mais en langue naturelle sans utiliser le formalisme de Gentzen.

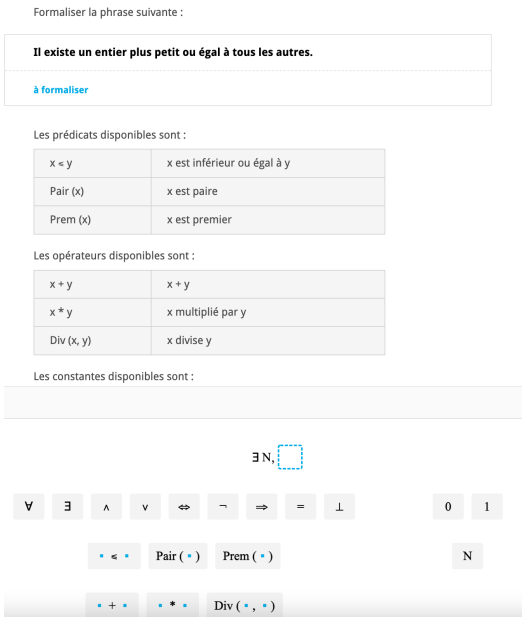
Le logiciel

Le logiciel que nous avons choisi d'utiliser est le logiciel Edukera développé par Benoit Rognier et Guillaume Duhamel [26]. Edukera est un assistant de preuve sous la forme d'une application Web, l'interaction avec l'utilisateur·rice repose essentiellement sur une interface graphique à base de clics, sans avoir à apprendre une syntaxe particulière. Edukera utilise l'assistant de preuve Coq [29], mais celui-ci est invisible pour l'utilisateur·rice. Edukera est un peu à l'apprentissage de la démonstration ce que Scratch est à l'apprentissage de la programmation. C'est à dire que c'est un système conçu spécifiquement pour l'enseignement, ce n'est pas un logiciel utilisable professionnellement. Le logiciel est facile d'accès, il est disponible en français et anglais, et les notations utilisées sont celles des mathématiques usuelles [8], des didacticiels permettent de découvrir conjointement les règles de la logique (déduction naturelle) et le logiciel avec très peu d'aide nécessaire de la part de l'enseignant·e. Cela est un atout dans notre contexte avec une équipe pédagogique fluctuante et des moyens limités en terme d'heures de face à face pédagogique en salle machine. L'intégration de la plateforme dans Moodle permet de proposer des exercices d'entraînement à la maison ainsi que des exercices notés à des cohortes d'étudiant·e·s conséquentes avec récupération automatique des notes, sans problèmes d'installation. Pour les devoirs maison, le mode d'interaction avec le logiciel via une interface graphique a aussi l'intérêt de rendre plus compliqué le plagiat pour les étudiant·e·s qu'avec un logiciel se basant sur une interface textuelle, où les fichiers peuvent être copiés facilement. Edukera possède néanmoins des limitations majeures : le logiciel n'est pas *open-source* ni libre, les démonstrations réalisées par les utilisateur·rices ne sont pas vérifiées par Coq, ce qui rend le logiciel plus vulnérable à des bugs mettant en jeu la validité des démonstrations produites. De plus Edukera n'est plus maintenu depuis 2020, il comporte quelques bugs et lenteurs et l'enseignant·e ne peut pas ajouter ses propres exercices.

Edukera propose deux types d'exercices : des exercices de formalisation et de démonstration. Les exercices de formalisation consistent en une phrase en langue naturelle qu'il s'agit de traduire sous la forme d'une proposition en logique de premier ordre dans un langage donné. La réponse proposée est alors comparée à la réponse attendue à l'aide d'un démonstrateur automatique (Fig.6).

8. En comparaison des assistants de preuve comme Coq, pour lesquels il est usuel, par exemple, de présenter l'application des fonctions à leurs arguments sous forme curryfiée (`f x y` à la place de `f(x,y)`).

FIGURE 6 – Edukera en mode exercices de formalisation



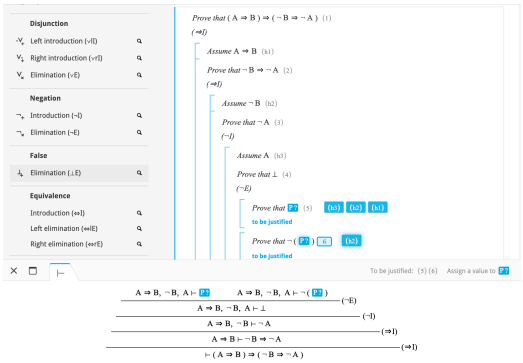
L'interface pour saisir la réponse n'est pas textuelle, mais consiste à saisir l'arbre de syntaxe abstraite de la formule en notation polonaise préfixée. Cela impose à l'utilisateur·rice d'avoir en tête la structure de la proposition ainsi que la notion de connecteur principal.

Edukera propose principalement des exercices de démonstration : un énoncé est proposé et l'utilisateur·rice doit produire de manière interactive une démonstration.

Deux modes de démonstration sont proposés : le mode logique et le mode maths.

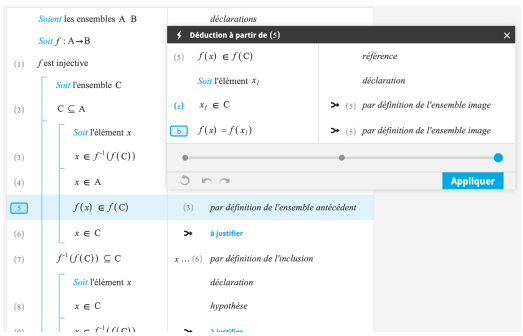
Dans le mode logique, les exercices sont constitués de tautologies en logique propositionnelle ou du premier ordre. Les règles de déduction proposées sont celles de la déduction naturelle de Gentzen, sans automatisation. La démonstration construite est visualisée au fur et à mesure à la fois sous une forme textuelle et une forme arborescente. L'enseignant·e peut choisir entre plusieurs types de visualisation des preuves à la Fitch, Prawitz ou Gentzen (Fig. 7).

FIGURE 7 – Edukera : visualisation d'une démonstration sous forme d'arbre à la "Gentzen"



Dans le mode maths, les exercices proposés sont des propriétés élémentaires des ensembles, des fonctions (image directe et réciproque, injectivité, surjectivité), des relations, ainsi que des propriétés à démontrer par récurrence ou en manipulant les sommes et produits. Ici le niveau de granularité dans l'élaboration des preuves se rapproche plus de la pratique usuelle des mathématiques. Par exemple, si on sait que $f(x) \in f(C)$, en un clic le logiciel permet d'obtenir x_1 tel que $x_1 \in C$ et $f(x) = f(x_1)$ (Fig.8), alors qu'en déduction naturelle il faudrait utiliser la règle d'élimination de l'existentiel puis la règle d'élimination de la conjonction [9].

FIGURE 8 – Edukera : exemple d'une fenêtre modale permettant d'applique la définition d'image directe



Contenu

Les deux premiers cours consistent en une introduction au vocabulaire des mathématiques

9. Une vidéo d'un exemple de démonstration en Edukera est disponible ici :`https://hal.archives-ouvertes.fr/hal-03648357/file/Coq-reverse-inclusion.mov`

(Axiome, Théorème, Lemme, Définition, Corollaire, condition nécessaire, contraposée, réciproque, ...), à la structure d'un document mathématique, et aux règles de raisonnement. Ce cours est suivi de deux séances de TP, les étudiant·e·s étant censés avoir réalisé en autonomie le didacticiel, l'objectif est de répondre aux questions sur les exercices plus difficiles et de créer du lien entre l'interaction avec le logiciel et la manière de rédiger des preuves sur papier. Nous insistons sur la distinction entre $\Rightarrow$ et "donc", la nécessité d'introduire les objets avant de les utiliser et la notion de variable libre/liée mise en rapport avec l'utilisation de ce même concept en informatique. La suite du cours est classique, vocabulaire ensembliste, ensemble des parties, produit cartésien, fonction injective, surjective, bijective, image directe, image réciproque, relation d'équivalence et relation d'ordre, quotient, entiers et démonstration par récurrence.

Organisation de la formation

Nous avons expérimenté plusieurs formules concernant l'organisation et les modalité d'évaluation, la première année nous avions opté pour un planning très dense en début de semestre, ce cours étant supposé être un pré-requis pour les cours d'analyse et de géométrie. Les exercices Edukera étaient évalués via un TP noté en temps limité. Les années suivantes nous avons opté pour un planning plus étalé dans le temps. Nous avons constaté que certains étudiant·e·s avaient pu réaliser tous les exercices en Edukera, mais qu'ils n'étaient pas toujours capables de reproduire ces mêmes exercices sur papier lors de l'évaluation. Après analyse de copies, nous avons constaté que les étudiant·e·s avaient pu acquérir des connaissances relative à la structure des démonstrations, mais parfois sans mémoriser les définitions. Pour pallier ce problème, nous avons d'abord tenté d'introduire dans le contrôle continu des QCMs constitués de questions de cours sur les définitions. Cela n'a pas été très concluant, car beaucoup d'étudiant·e·s avaient l'impression d'avoir atteint l'objectif en ayant appris les définitions. En 2022, nous leur avons proposé un auto-test (hors évaluation) afin qu'ils et elles puissent vérifier par eux-même leur connaissance du cours, en étant capable de produire à la fois les définitions formelles, une phrase en français et des exemples et contre-exemples variés.

Bilan

Nous avons un retour globalement positif des étudiant·e·s, beaucoup indiquent que l'utilisation de l'assistant de preuve les a aidés à comprendre les mécanismes de démonstration (voir Fig.9). Néanmoins, certains rapportent parvenir à réaliser des exercices sans vraiment comprendre ce qu'ils ou elles ont fait grâce à l'automatisation relative fournie par le logiciel ainsi que la boucle d'interaction permanente mise en jeu. D'autres indiquent que pour les exercices les plus difficiles, il est nécessaire de réfléchir et d'être familier avec les concepts mis en jeu.

Notre impression globale (qu'il serait intéressant de valider scientifiquement) est que étant donné le mode d'interaction d'Edukera, en particulier le fait que le logiciel ne nécessite pas de connaître les définitions des objets mis en jeux, il est crucial que les enseignant·e ·s soient attentifs à l'acquisition des définitions des concepts mathématiques de façon complémentaire à la pratique des exercices sur papier et sur machine.

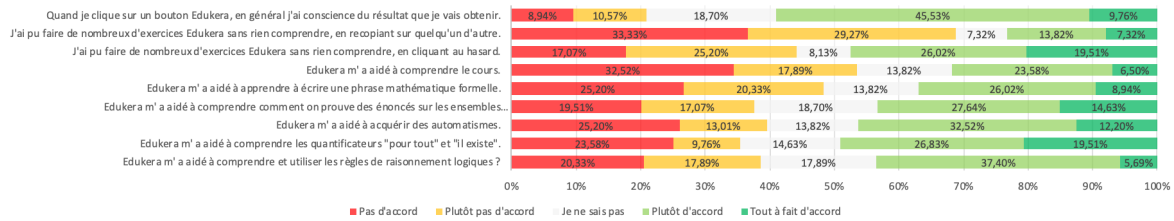
2.5 – Université de Montpellier

Simon Modeste

Cours concerné : *Raisonnement et théorie des ensembles - L1 mathématiques*

Le cours Raisonnement et théorie des ensemble est un petit cours de début d'année au premier semestre de L1 de mathématiques. Ce cours est composé de 9 h de CM (6 × 1h30) et de 10,5h de TD (7 × 1h30). Il concerne environ 160 étudiant·e·s (L1 mathématiques, double Licence mathématiques et informatique, CUPGE), qui ont donc suivi des mathématiques au lycée et se destinent à des formations à forte composante mathématique. Ce cours a pour objectif d'introduire en tout début de semestre, avant le démarrage des UE d'Analyse I (fonctions d'une variable et suites) et d'Algèbre I (systèmes linéaires) les bases du discours mathématique, des éléments de logique (des propositions, puis des prédicats), les raisonnements et méthodes de démonstrations classiques, le vocabulaire et les opérations ensemblistes ainsi que les méthodes de démonstration associées, et finalement les notions de base sur les applications mathématiques.
Le cours essaie de rendre explicite les règles de fonctionnement des démonstrations en mathéma-

Figure 9 – Edukera : résultat d'une enquête auprès des étudiant·e·s de Strasbourg



tiques, et vise à ce que les étudiant·e·s sachent rédiger des démonstrations sur des objets élémentaires, connus depuis le lycée (fonctions, nombres, suites…). Ceci doit leur permettre de mieux comprendre les démonstrations présentées dans les autres UE de mathématiques, et de produire des démonstrations dans ces UE. Bien que concentrée en tout début d'année (CM sur 2 semaines, TD sur 3 semaines), l'UE est évaluée avec des CC durant le semestre (partant de l'hypothèse qu'ils et elles continuent de progresser au fil du semestre).
C'est dans le contexte de cette évaluation en CC intégral qu'un assistant de preuve a été introduit dans l'UE.

Le logiciel

Le logiciel utilisé est le logiciel Edukera, présenté en début de partie 2.4. Dans notre situation, il présente l'avantage d'une prise en main possible en quasi autonomie, d'une compatibilité avec Moodle, et d'une base d'exercices (dont certains avec tutoriel) à partir de laquelle on peut construire un ensemble d'exercices proposés aux étudiant·e·s. Le langage utilisé par le logiciel et la forme des démonstrations sont proches des pratiques en mathématiques.
Un exercice peut être recommencé autant de fois qu'on le souhaitent pour le valider. La visualisation des preuves utilisée s'est limitée au mode appelé "Fitch" (qui est le plus proche des preuves en mathématiques, voir figure 8).

Contenu

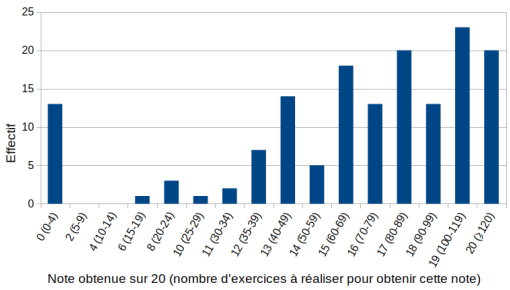
En CM sont présentées les bases du discours mathématique (Que trouve-t'on dans un texte mathématiques ? Qu'est-ce qu'un énoncé mathématique ?) et les éléments de logique utiles en mathématiques (logique des propositions, variables, quantification). La notion de démonstration et les démonstrations spécifiques aux différentes formes d'énoncés sont présentées (énoncés existentiels et universels, conjonction et disjonctions, négations, etc.), ainsi que les raisonnements spécifiques (absurde, contraposée, récurrence…). La suite du cours concerne les ensembles, leur utilisation, et les démonstrations impliquant les ensembles. Enfin, la dernière partie du cours aborde les applications, leur définition et leur utilisation, les notions d'image, image réciproque, la composition, l'injectivité, la surjectivité et la bijectivité ainsi que la notion d'application réciproque. Les exercices de TD commencent par travailler la formalisation des énoncés, et leur valeur de vérité, puis les démonstrations d'énoncés mathématiques, et enfin le travail ensembliste et avec les applications. Le nombre de CM et de TD est relativement faible par rapport au contenu, mais nous comptons sur le fait que les notions introduites et travaillées sont réutilisées et approfondies dans les autres UE de mathématiques du semestre (Algèbre, Analyse, Géométrie).

Organisation de la formation

L'UE est évaluée avec 3 CC. Les CC 1 et CC 3 sont des devoirs sur table d'une durée d'une heure, qui ont lieu en début et fin de semestre (le premier une ou deux semaines après la fin de l'UE). C'est pour l'évaluation de CC 2 que l'assistant de preuve a été utilisé. Cette évaluation, pour une UE de seulement 2 ECTS et pour la première année d'utilisation de l'assistant de preuve, a été relativement bienveillante et valorisant le travail régulier. Les étudiant·e·s ont eu à leur disposition une liste d'exercices Edukera sur laquelle ils et elles sont supposées travailler tout au long du semestre, en autonomie à la maison. À la fin du semestre, c'est le nombre d'exercices réalisés qui permet de produire la note de CC 2 (on peut lire sur la figure 10 le barème de correspondance nombre d'exercices réalisés - note, donné aux étudiant·e·s en début

d'année).
Environ 170 exercices ont été sélectionnés parmi ceux proposés par la plateforme Edukera. Cela commence par 42 exercices de formalisation logique (énoncés hors-mathématiques puis mathématiques), puis d'énoncés à prouver avec l'assistant de preuve, commençant par de la logique booléenne, puis du premier ordre, puis utilisant les ensembles, et enfin des énoncés portant sur les applications et leurs propriétés. Ces exercices étaient assez proches (parfois un peu plus techniques) de ce que l'on pouvait demander en TD, et la progression d'Edukera assez proche de celle du cours (on peut sélectionner un sous-ensemble d'exercices, mais pas choisir l'ordre de présentation, déterminé par le logiciel – l'étudiant·e peut néanmoins traiter les exercice dans l'ordre souhaité et sauter des exercices).
On retrouve en figure 10 la répartition des notes de CC 2 des 169 inscrits. Plus de la moitié ont traité 70 exercices ou plus.

Figure 10 – Répartition des notes



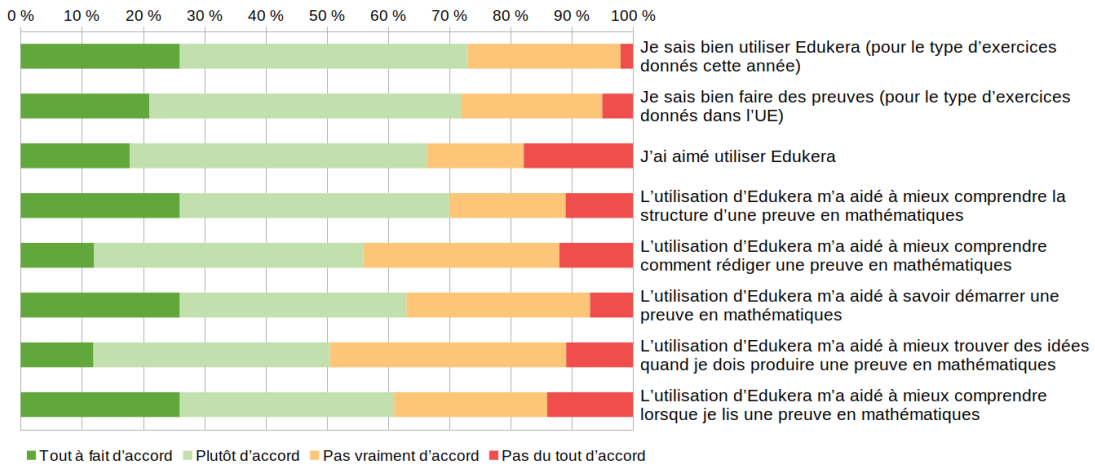
Aucun moyen supplémentaire n'était disponible pour accompagner la prise en main et l'avancée dans les exercices. Edukera comporte des tutoriels, qui permettent de prendre en main chacune des nouvelles fonctionnalités, introduites au fil des exercices, et un document de présentation avait été réalisé pour la toute première connexion et prise en main. La situation nous a aussi permis de mettre en place un créneau hebdomadaire d'aide sur l'assistant de preuve, en salle informatique, ouvert à tous et toutes. Ceci a été permis par l'existence d'un mentorat par des étudiant·e·s de M2, déjà financé, et le fait qu'un des mentors avait déjà utilisé un assistant de preuve, dans un cours de L3.

Bilan

Le bilan de cette expérience, du point de vue de l'enseignant, est plutôt positif.

Un questionnaire d'évaluation de l'outil et de sa perception a été proposé aux étudiant·e·s à l'issu du semestre, avec des questions fermées (à échelle de Lickert) et des questions ouvertes.
Parmi les questions fermées avec échelles de Lickert, les 3 premières concernaient le niveau de maîtrise du logiciel, le niveau de maîtrise de la preuve, et l'appréciation du logiciel, les 5 suivantes concernaient l'apport perçu du logiciel pour l'apprentissage de la preuve. Nous avons reçu 57 réponses. La figure 11 montre ces questions, et les proportions des réponses obtenues.
On constate qu'une majorité semble considérer que l'utilisation d'Edukera a contribué à des apprentissages autour de la compréhension, rédaction, conception de preuves en mathématiques. Les proportions de chaque opinion pour chaque item semblent assez stables (à une dizaine de pourcents près), y compris entre les 3 premières questions (entre elles) et avec les 5 autres. Concernant les apprentissages perçus, on note que l'item le plus approuvé concerne la compréhension de la structure des preuves (70%), suivi par le démarrage d'une preuve (63%), et la lecture de preuves (61%). L'item le moins approuvé est celui concernant l'aide à trouver les idées pour les preuves (51%). Ces réponses semblent assez cohérentes avec ce que nous identifions comme les intérêts d'un assistant de preuve pour l'enseignement des mathématiques.
Dans les commentaires, un point revient souvent : les étudiant·e·s disent avoir eu des difficultés à prendre en main le logiciel sans aide, et regrettent qu'une séance de prise en main ou une présentation en amphi n'ait pas eu lieu. Nous allons essayer de mieux intégrer la présentation de l'outil, l'année prochaine, lors des CM.
Un autre possibilité serait de profiter des appels à soutien aux projets pédagogiques qui existent dans beaucoup d'universités pour obtenir des heures ou des moyens supplémentaires pour cette expérimentation.

Figure 11 – Edukera : résultat d'une enquête auprès des étudiant·e·s à Montpellier



3. Conclusion

Bien que les assistants de preuve aient été initialement conçus pour la recherche en informatique et en mathématiques, il nous semble effectivement pertinent, comme le suggèrent Hanna et Yan [18], de revisiter l'enseignement de la preuve en s'appuyant sur cet outil (relativement) nouveau, et de produire des analyses fines d'expérimentations, intégrant l'analyse de l'outil utilisé, mais aussi du déroulement des séquences d'enseignement, des productions des étudiant·e·s. De telles analyses pourront s'appuyer sur une littérature didactique riche sur l'enseignement et l'apprentissage de la preuve dans le supérieur (voir par exemple une synthèse dans[28]). Nous pensons d'une part que derrière un même intérêt quel que soit le logiciel utilisé, il y a des différences entre les assistants de preuve qui peuvent avoir des conséquences sur les apprentissages, d'autre part que ces logiciels, comme tout outil d'enseignement, doivent être pensés avec leur « mode d'emploi » pour ce qui est de l'enseignement. Concernant le premier point, nous avons produit une analyse a priori des impacts didactiques de différents assistant de preuve [3]. Concernant le deuxième point, des analyses d'expérimentations avec Edukera ou DƎVDUCTION ont été faites ou sont en cours.

Dans notre conclusion de cet article, nous reprenons quelques observations saillantes issues des expérimentations relatées ci-dessous ou ailleurs, puis nous soulignerons quelques points qui font débat et qui amènent à prendre certaines précautions d'usage.

3.1 – Observations et constats

Les diverses expérimentations d'utilisation d'assistants de preuve montrent que les étudiant·e·s s'engagent dans la recherche de preuves avec ces outils, là où en papier-crayon nous les voyons trop souvent renoncer, ne sachant même pas par où commencer. L'utilisation d'un ordinateur permet que chacun et chacune avance à son rythme, tout en échangeant éventuellement avec les autres, tâtonne en bénéficiant des rétroactions de l'assistant de preuve. L'interprétabilité de ces rétroactions est un élément important dans le développement des assistants de preuve pour l'enseignement. D'une part, les rétroactions fournies par l'assistant de preuve peuvent être plus ou moins détaillées pour aider la compréhension des erreurs. Mais quels que soient ces retours, un accompagnement par un·e enseignant·e est nécessaire pour aider les étudiant·e·s à comprendre le niveau de granularité attendu par la machine dans les preuves.

L'utilisation d'un assistant de preuve conduit à enseigner plus explicitement certains principes de la logique, ce que recommandent Durand-Guerrier, Boero, Douek, Epp et Tanguay [13]. Dans les expérimentations relatées ici, comme dans d'autres déjà citées, le travail avec un assistant de preuve permet une prise de conscience des mécanismes logiques sous-jacents à la construction et à la manipulation

des énoncés mathématiques, notamment dans la production des preuves. Tout d'abord, et ce quel que soit le degré de formalisme avec lequel cela est fait (et l'on voit dans les expérimentations relatées ici qu'il y a différentes positions sur le sujet), il est nécessaire de nommer et de donner quelques propriétés des connecteurs et quantificateurs, et, comme cela a déjà été mentionné, de bien distinguer les trois façons dont chaque connecteur ou quantificateur peut intervenir : pour former un énoncé, pour utiliser un énoncé et pour démontrer un énoncé.

Donnons deux exemples d'une telle distinction. Un assistant de preuve distingue clairement les deux étapes que sont affirmer une implication (et alors le symbole $\Rightarrow$ est utilisé dans la formation de cet énoncé qui se trouve dans le contexte), puis utiliser cette implication, ce qui consiste à affirmer également la prémisse, et à en déduire la conclusion, ce que l'on rédige fréquemment en utilisant le mot « donc ». Malheureusement, il est très courant de voir écrit $P \Rightarrow Q$ comme abréviation de « on sait que P et que $P \Rightarrow Q$ donc on obtient Q ». On notera que la version correcte fait intervenir le connecteur d'implication dans son rôle de formation d'énoncé et une utilisation de cet énoncé. Ce type de raccourci est sans conséquence pour un expert, mais conduit parfois à une confusion tragique, par exemple si on applique cette interprétation à la définition de limite d'une suite qui devient « pour tout ε strictement positif, il existe N tel que, pour tout n, $n \geqslant N$ donc $|u_n - l| < \varepsilon$ ».

Un autre exemple type, cette fois-ci de confusion entre formation d'un énoncé et démonstration de cet énoncé provient du quantificateur universel et fait aussi intervenir une confusion entre variables libres et variables liées. La suite de symboles $\forall x$ est le début d'un énoncé, pas d'une démonstration. La démonstration sur papier d'un tel énoncé commencerait par « Soit x » ou « Fixons x » et avec un assistant de preuve, par un terme précis bien distinct du symbole $\forall$. Mais on voit parfois au tableau des démonstration commençant par $\forall x$ ou par les mots « Pour tout x ». Là encore il y a risque de confusion car la variable x est alors liée par le quantificateur alors que la version correcte introduit une variable libre et donc un objet mathématique fixé jusqu'à la fin de la démonstration.

À l'issu de leur analyse de preuves produites par des étudiant·e·s ayant ou non travaillé avec Lean, Thoma et Iannone [30] concluent que l'usage de Lean favorise l'utilisation correcte du langage mathématique et du symbolisme, et conduit à préciser plus clairement à quel ensemble appartiennent les objets manipulés. Pour utiliser un assistant de preuve, on est confronté à la nécessité d'un langage formalisé, qu'il faut apprendre car il y en a besoin pour parler avec la machine. Ce langage est utilisé comme un médiateur entre l'étudiant·e et l'ordinateur, mais aussi entre le langage mathématique académique et le langage utilisé par les étudiant·e·s qui n'ont pas encore complètement acquis la familiarité avec les pratiques langagières de la communauté mathématique. Ce n'est pas seulement l'usage en soi d'un assistant de preuve qui permet de lever les implicites de ces pratiques, et de montrer l'importance d'être précis quant aux mots utilisés dans la rédaction de preuves, mais plutôt des discussions avec les étudiant·e·s suscitées par des allers-retours entre preuve avec l'assistant de preuve et preuve en papier-crayon.

Le travail avec un assistant de preuve montre aussi qu'il y a dans la production de preuve une part qui peut-être déroulée de façon automatique, en se basant sur quelques règles de déduction, et une part qui demande de s'arrêter un instant pour réfléchir. Les logiciels comme DƎ∀DUCTION ou Edukera prennent en charge une partie de l'automatisation : un clic sur le bon bouton (et cela vient assez rapidement) permet de transformer l'état de preuve, et par exemple de passer d'un but sous forme d'implication à l'introduction d'une hypothèse qui en est la prémisse, et d'un nouveau but qui en est la conclusion. Ainsi, quand les étudiant·e·s travaillent sur le logiciel, ils et elles sont soulagées de cette partie, et peuvent se concentrer sur la recherche des idées. Mais cela leur montre aussi que ces automatismes, basés sur les liens entre structure de la preuve et structure de l'énoncé à prouver, sont reproductibles dans la rédaction sur papier, et qu'il est même possible dans ce cas de se permettre plus de souplesse sur les mots utilisés ou sur le niveau de détail. Là encore, les analyses de Thoma et Iannone [30] montrent une meilleure structuration des preuves des étudiant·e·s ayant travaillé avec Lean.

3.2 – Débats et précautions d'usage

Malgré ces observations positives et apports, l'utilisation d'assistants de preuve pour l'enseignement semble aussi produire certaines difficultés, qui pourraient être liées à la nouveauté de ces enseignements, à des éléments de contextes (horaire trop faible) ou aux assistants de preuve intrinsèquement.

Un premier point que nous pouvons noter est le risque de glissement vers la preuve formelle [15]. L'assistant de preuve demande parfois d'entrer dans des détails de preuve qui ne sont pas habituellement exigés ou présentés dans l'enseignement. Il rend visible les règles formelles liées à la production/écriture d'une démonstration (et prend en charge le contrôle de l'application conforme de ces règles) mais peut produire un effet de réduction de l'activité de preuve à celles-ci. Ainsi, on observe que des étudiant·e·s semblent parvenir à réaliser des preuves sans vraiment comprendre ce qu'ils ou elles ont fait ou ce que fait la preuve, ce qui peut favoriser une approche syntaxique dans la production de preuve [32]. Or, différentes recherches montrent l'importance d'articuler les aspects syntaxiques et sémantiques [12]. Par ailleurs, Hanna et Yan [18] s'interrogent sur la possibilité, avec un assistant de preuve, de travailler sur la preuve dans sa fonction d'explication et pas seulement dans sa fonction de validation. Les assistants de preuves ne prennent pas en charge l'aspect sémantique, ni la dimension explicative, il est donc nécessaire d'accompagner leur utilisation par des activités plus traditionnelles (recherche d'exemples, de contre-exemples, utilisation de dessins, rédaction papier-crayon).

Cet aspect peut être renforcé par l'automatisation plus ou moins forte de l'application des règles de déduction, et le contrôle permanent qu'exerce le logiciel sur les énoncés produits et la validité des règles appliquées. Cela peut engendrer une boucle essai-erreur assez courte, qui peut maintenir les étudiant·e·s dans une approche superficielle (c'est particulièrement le cas avec les logiciels disposant d'une interface cliquable, qui favorise une prise en main rapide, mais peuvent produire un effet pilote-automatique). Cet aspect serait à comparer avec des phénomènes similaire dans l'apprentissage de la programmation.

L'usage d'assistants de preuve en cours de mathématiques pose aussi des questions sur les objectifs de l'enseignement dispensé. Ceci met parfois à jour des différences de points de vue sur la démonstration mathématique et son apprentissage au sein des équipes pédagogiques. Ces débats transparaissent dans les témoignages de cet article. Un point concerne l'enjeu d'enseignement de « la » preuve versus « des » preuves. Le but de l'usage d'un assistant de preuve dans l'enseignement mathématique est-il de comprendre ce qu'est la démonstration en mathématiques ou de réaliser des démonstrations mathématiques ? Le but est-il de s'assurer de produire des preuves valides et de certifier les résultats mathématiques, ou de mieux comprendre les étapes-clés ou la structure globale de leurs preuves ? L'objectif est-il de savoir construire (ou de savoir qu'on peut construire) des preuves formelles des résultats mathématiques, ou d'outiller l'étudiant·e pour lire, produire et contrôler des preuves « papier-crayon » ? Plus profondément, ceci questionne quelle est la référence de ce qu'est une preuve ou une démonstration dans un cours de mathématique de début d'université, et sur quels éléments de logique et quel formalisme on s'appuie. Les assistants de preuve, en rendant visibles cette logique et ce formalisme, contribuent nécessairement à cette référence pour les étudiant·e·s, mais la façon dont on les présente et dont on rend compte de leur fonctionnement peut varier. Par exemple, les assistants présentés dans cet article reposent tous sur des systèmes logiques qui ne sont pas la théorie des ensemble (utilisée classiquement dans l'enseignement), notamment la théorie des types. Cette différence entre théories peut-elle (et doit-elle) rester transparente ? Nécessite-t-elle d'introduire certains éléments théoriques (au delà des règles de déduction, qui nous semblent unanimement importantes à expliciter) ? Peut-elle poser des problèmes de compréhension de l'assistant de preuve si on laisse certaines choses implicites ? Plus prosaïquement, la question de la distance entre la preuve des mathématiciens et la preuve dans l'enseignement de début d'université se pose aussi. Par exemple, comme mentionné plus haut, l'usage du quantificateur existentiel, et de l'application implicite de son élimination, dont on sait qu'il pose problème aux étudiant·e·s demande une réflexion sur la nécessité de lever cet implicite dans l'enseignement sans trop s'éloigner de la pratique mathématique [10].

Ces débats existent déjà dans l'enseignement supérieur des mathématiques, mais il nous semble que l'usage d'assistants de preuve peut être un révélateur de ces questions importantes.

L'assistant de preuve peut contribuer à l'enseignement et l'apprentissage de la preuve et il nous semble nécessaire de développer des expérimentations et des recherches sur le sujet. L'usage d'assistants de preuve ne peut se substituer à un enseignement plus classique de la démonstration, et c'est dans l'articulation entre preuves formelles et preuves papier-crayon, accompagnée par l'enseignant·e, que nous voyons un potentiel important.

Références

[1] J. Avigad. "Learning Logic and Proof with an Interactive Theorem Prover". In: *Proof Technology in Mathematics Research and Teaching*. Ed. by G. Hanna, D. A. Reid, and M. de Villiers. Mathematics Education in the Digital Era. Cham: Springer International Publishing, 2019, pp. 277–290. ISBN: 978-3-030-28483-1. DOI: 10.1007/978-3-030-28483-1_13. URL: https://doi.org/10.1007/978-3-030-28483-1_13 (visited on 12/06/2020).

[2] J. Avigad and J. Harrison. "Formally verified mathematics". *Commun. ACM* **57**, no.4 (Apr. 2014), pp. 66–75. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2591012. URL: https://dl.acm.org/doi/10.1145/2591012 (visited on 06/22/2022).

[3] E. Bartzia, A. Meyer et J. Narboux. « Proof assistants for undergraduate mathematics and computer science education: elements of a priori analysis ». In : *INDRUM 2022: Fourth conference of the International Network for Didactic Research in University Mathematics*. Sous la dir. de M. Trigueros. Hanovre, Germany : Reinhard Hochmuth, oct. 2022. URL : https://hal.archives-ouvertes.fr/hal-03648357.

[4] J. Barwise et J. Etchemendy. « Hyperproof: for Macintosh » (1994).

[5] J. Blanc et al. « Proofs for freshmen with Coqweb ». *PATE'07* (2007). bibtex: blanc2007proofs, p. 93.

[6] R. Bornat. *Natural Deduction Proof and Disproof in Jape (March 2004)*.

[7] J. Breitner. « Visual theorem proving with the Incredible Proof Machine ». In : *International Conference on Interactive Theorem Proving*. Springer, 2016, p. 123-139.

[8] K. Broda et al. « Pandora: A reasoning toolbox using natural deduction style ». *Logic Journal of the IGPL* **15**, n° 4 (2007). Publisher: OUP, p. 293-304.

[9] D. M. Cerna. « AXolotl: A Self-study Tool for First-order Logic » ().

[10] F. Chellougui. « La déduction naturelle de Copi comme outil didactique pour l'analyse de preuves mathématiques - Revue RDM ». *Recherches en didactique des mathématiques* **38**, n° 1 (2022), p. 1-15. URL : https://revue-rdm.com/2020/la-deduction-naturelle-de-copi-comme-outil-didactique-pour-lanalyse-de-preuves-mathematiques/ (visité le 08/07/2022).

[11] P. Corbineau. « A declarative language for the Coq proof assistant ». In : *International Workshop on Types for Proofs and Programs*. Springer, 2007, p. 69-84.

[12] V. Durand-Guerrier. "Truth versus validity in mathematical proof". *ZDM Mathematics Education* **40**, no.3 (June 21, 2008), pp. 373–384. ISSN: 1863-9690, 1863-9704. DOI: 10.1007/s11858-008-0098-8. URL: http://link.springer.com/article/10.1007/s11858-008-0098-8 (visited on 10/26/2016).

[13] V. Durand-Guerrier et al. "Examining the Role of Logic in Teaching Proof". In: *Proof and Proving in Mathematics Education*. Ed. by G. Hanna and M. d. Villiers. New ICMI Study Series 15. Springer Netherlands, 2012, pp. 369–389. ISBN: 978-94-007-2128-9 978-94-007-2129-6. DOI: 10.1007/978-94-007-2129-6_16. URL: http://link.springer.com/chapter/10.1007/978-94-007-2129-6_16 (visited on 10/26/2016).

[14] A. H. From, J. Villadsen et P. Blackburn. « Isabelle/HOL as a Meta-Language for Teaching Logic ». *arXiv preprint arXiv:2010.16014* (2020).

[15] M. Gandit. « Etude épistémologique et didactique de la preuve en mathématiques et de son enseignement. Une ingénierie de formation. » In : *Actes du séminaire national de didactique des mathématiques 2010*. Paris : IREM de Paris, Association pour la Recherche en Didactique des Mathématiques (ARDM), 2011, p. 175-197. ISBN : 2-86612-330-1.

[16] O. Gasquet, F. Schwarzentruber et M. Strecker. « PANDA: a proof assistant in natural deduction for all. a Gentzen style proof assistant for undergraduate students ». In : *International Congress on Tools for Teaching Logic*. Springer, 2011, p. 85-92.

[17] T. Hales. « Formal Proof ». *Notices of the American Mathematical Society* **55** (1er jan. 2008).

[18] G. Hanna et X. Yan. *COMPUTER-ASSISTED PROVING IN THE CLASSROOM*. 23 mars 2021.

[19] G. Leach-Krouse. « Carnap: An open framework for formal reasoning in the browser ». *arXiv preprint arXiv:1803.03092* (2018).

[20] J. S. Lodder. « The Design and Use of Tools for Teaching Logic » (2020).

[21] A. Mahboubi. *Un ordinateur pour vérifier les preuves mathématiques*. 2014. URL : https://images.math.cnrs.fr/Un-ordinateur-pour-verifier-les-preuves-mathematiques.

[22] B. C. Pierce et al. *Logical Foundations*. Sous la dir. de B. C. Pierce. **1**. Software Foundations. Electronic textbook, 2021.

[23] B. C. Pierce et al. *Programming Language Foundations*. Software Foundations series, volume 2. Electronic textbook, mai 2018.

[24] C. Pit-Claudel. "Untangling mechanized proofs". In: *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering*. SPLASH '20: Conference on Systems, Programming, Languages, and Applications, Software for Humanity. Virtual USA: ACM, Nov. 16, 2020, pp. 155–174. ISBN: 978-1-4503-8176-5. DOI: 10.1145/3426425.3426940. URL: https://dl.acm.org/doi/10.1145/3426425.3426940 (visited on 06/22/2022).

[25] C. Raffalli et R. David. « Computer Assisted Teaching in Mathematics ». In : *Workshop on 35 years of Automath*. Edinburgh, United Kingdom, 2002.

[26] B. Rognier et G. Duhamel. « Présentation de la plateforme edukera ». In : *Vingt-septièmes Journées Francophones des Langages Applicatifs (JFLA 2016)*. Sous la dir. de J. Signoles. bibtex: rognier:hal-01333606 bibtex[hal_id=hal-01333606;hal_version=v1]. Saint-Malo, France, jan. 2016. URL : https://hal.archives-ouvertes.fr/hal-01333606.

[27] A. Schlichtkrull, J. Villadsen et A. H. From. « Students' Proof Assistant (SPA) ». *arXiv preprint arXiv:1904.00617* (2019).

[28] A. Selden. "Transitions and Proof and Proving at Tertiary Level". In: *Proof and Proving in Mathematics Education: The 19th ICMI Study*. Ed. by G. Hanna and M. de Villiers. Dordrecht: Springer Netherlands, 2012, pp. 391–420. ISBN: 978-94-007-2129-6. DOI: 10.1007/978-94-007-2129-6_17. URL: https://doi.org/10.1007/978-94-007-2129-6_17 (visited on 03/22/2022).

[29] The Coq development team. « The Coq proof assistant » (2020). DOI : 10.5281/zenodo.4021912. URL : http://doi.org/10.5281/zenodo.4021912.

[30] A. Thoma and P. Iannone. "Learning about Proof with the Theorem Prover LEAN: the Abundant Numbers Task". *Int. J. Res. Undergrad. Math. Ed.* (July 7, 2021). ISSN: 2198-9745, 2198-9753. DOI: 10.1007/s40753-021-00140-1. URL: https://link.springer.com/10.1007/s40753-021-00140-1 (visited on 03/26/2022).

[31] J. Villadsen, A. H. From et A. Schlichtkrull. « Natural deduction and the Isabelle proof assistant ». *arXiv preprint arXiv:1803.01473* (2018).

[32] K. Weber and L. Alcock. "Semantic and Syntactic Proof Productions". *Educational Studies in Mathematics* **56**, no.3 (2004), pp. 209–234. ISSN: 0013-1954. DOI: 10.1023/B:EDUC.0000040410.57253.a1. URL: http://link.springer.com/10.1023/B:EDUC.0000040410.57253.a1 (visited on 11/29/2020).

Marie Kerjean
CNRS ,LIPN - Université Sorbonne Paris Nord
kerjean@lipn.fr
https://lipn.univ-paris13.fr/~kerjean/

Frédéric Le Roux
UMR 7586 CNRS, Sorbonne Université
frederic.le-roux@imj-prg.fr
https://perso.imj-prg.fr/frederic-leroux/

Patrick Massot
Faculté des Sciences d'Orsay
patrick.massot@math.cnrs.fr
https://www.imo.universite-paris-saclay.fr/~pmassot/

Micaela Mayero
UMR 7030, Université Sorbonne Paris Nord
mayero@lipn.univ-paris13.fr
https://lipn.univ-paris13.fr/~mayero/

Zoé Mesnil
LDAR, Université Paris Cité, Univ. Paris Est Créteil, CY Cergy Paris Université, Univ. Lille, UNIROUEN
zoe.mesnil@u-paris.fr

Simon Modeste
IMAG, Université de Montpellier, CNRS
simon.modeste@umontpellier.fr
https://imag.umontpellier.fr/~modeste/

Julien Narboux
UMR 7357 CNRS, Université de Strasbourg, France
narboux@unistra.fr
https://dpt-info.u-strasbg.fr/~narboux/

Pierre Rousselin
Université Paris 13
rousselin@math.univ-paris13.fr
https://www.math.univ-paris13.fr/~rousselin/