La forme des mathématiques qui viennent
Alex Kontorovich

Résumé

Nous présentons un aperçu de la manière dont certains outils informatiques interagissent actuellement avec la pratique mathématique, et réfléchissons aux implications pour la recherche mathématique à court et moyen terme, alors que le domaine navigue dans l'ère émergente de l'ia et des systèmes de vérification formelle.

1. Introduction.

Cet article discutera d'une vision de ce à quoi la recherche mathématique pourrait ressembler à l'ère dans laquelle nous semblons maintenant entrer, celle de l'ia et de la formalisation. Compte tenu du rythme des progrès, mes propos seront probablement obsolètes presque au moment où ils apparaissent, mais je prendrai néanmoins le risque de décrire comment je vois le paysage, en septembre 2025.

Mon implication personnelle dans la formalisation a commencé il y a environ cinq ans (voir [9]), motivée par les défis particuliers de mon domaine de recherche. La théorie analytique des nombres se caractérise par des arguments techniques particulièrement longs, s'étendant sur des dizaines de pages. Ces preuves synthétisent typiquement de nombreux lemmes, chacun valable uniquement dans des plages contraintes de leurs paramètres, qui doivent s'aligner parfaitement à la fin pour que le théorème principal soit valide. Cette danse complexe de paramètres et de contraintes est à la fois ce que j'aime dans ce sujet, et ce qui le rend périlleux : une simple erreur de signe, enfouie quelque part au milieu de l'un de ces arguments techniques, peut invalider l'ensemble du résultat principal.

Lorsqu'un argument implique quelque chose comme une manipulation algébrique compliquée, je peux utiliser un système d'algèbre informatique comme Sage ou Mathematica pour le vérifier une fois pour toutes ; alors je ne perds plus de temps à repasser sur cette partie de l'argument encore et encore pour m'assurer qu'aucune erreur n'a été commise. Ne devrait-il pas exister un logiciel similaire, mais pour suivre les dépendances logiques et les contraintes de paramètres à travers l'ensemble de mon article ?

Un tel logiciel existe, et s'appelle un assistant de preuve interactif ; voir § 7. Il permet, en principe, aux utilisateurs de "formaliser" leurs articles, en entrant la preuve ligne par ligne jusqu'à ce que l'ordinateur certifie que l'argument est complet. La réalité, cependant, est édifiante. Dans l'état actuel des choses, je ne peux même pas énoncer formellement la plupart des théorèmes qui m'intéressent, et encore moins les prouver. Les bibliothèques formelles existantes (voir § 7.2) devraient croître de plusieurs ordres de grandeur avant que la recherche sur leur base devienne une réalité.

Cela conduit à une énigme fondamentale : les nouvelles mathématiques semblent croître à un taux exponentiel, disons δ, et les bibliothèques formelles croissent également de manière exponentielle, à un taux, disons ϵ ; mais pour autant que je puisse en juger, nous avons actuellement :

Traduction par ia corrigée de l'article arxiv https://arxiv.org/pdf/2510.15924 : Denise Vella-Chemla, août 2026.

$$\delta > \epsilon.$$

Donc, si cette inégalité persiste, la formalisation ne rattrapera jamais la recherche mathématique en langage naturel, et le rêve d'un logiciel aussi utile au raisonnement, que les systèmes d'algèbre informatique le sont à la manipulation symbolique, restera à jamais un mirage lointain.

À moins, bien sûr, que la formalisation puisse être "super-chargée" (calculée automatiquement ?) par l'automatisation. Et je suis donc passé de "vouloir faire des mathématiques de recherche", à "vouloir une formalisation suffisante pour faire des mathématiques de recherche", à enfin "vouloir une ia suffisante pour faire une formalisation suffisante pour faire des mathématiques de recherche" ! Ce qui suit est ma vision de la façon dont nous pourrions y parvenir, et de ce à quoi les choses pourraient ressembler si/quand nous y parviendrons. Le vieil adage : "Il est difficile de faire des prédictions, surtout sur l'avenir", attribué diversement à Yogi Berra, Niels Bohr et d'autres, est très approprié ici. Alors avant de commencer, prenons du recul.

2. En l'an 2000...

Au tournant du millénaire, certains des mathématiciens les plus éminents du monde se sont réunis à Tel Aviv pour une conférence afin de discuter de ce à quoi les mathématiques pourraient ressembler au 21e siècle. Les actes ont été publiés sous la forme d'un numéro spécial dans GAFA intitulé "*Visions of Mathematics*", et je recommande vivement la lecture de l'intégralité du numéro [1]. Une contribution particulièrement perspicace a été donnée par Tim Gowers, intitulée "*Rough Structure and Classification*" [11]. La section 2 de l'article de Gowers s'intitule de manière provocante : "Les mathématiques existeront-elles en 2099 ?" Dans celle-ci, il imagine un "dialogue entre un mathématicien et un ordinateur dans deux à trois décennies" - et nous y voilà ! Son dialogue imaginaire est étrangement similaire aux expériences que tant d'entre nous vivons désormais régulièrement en interagissant avec des "chatbots" tels que chatgpt d'openai, gemini de google, claude d'anthropic, grok de xai, et d'autres :

Mathématicien. Est-ce que ce qui suit est vrai ? Soit $\delta > 0$. Alors pour N suffisamment grand, tout ensemble $A \subset \{1, 2, \ldots, N\}$ de taille au moins δN contient un sous-ensemble de la forme $\{a, a + d, a + 2d\}$.

Ordinateur. Oui. Si A est non vide, choisissez $a \in A$ et posez $d = 0$.

M. D'accord, d'accord, mais que se passe-t-il si d n'est pas autorisé à être nul ?

O. Avez-vous essayé l'induction sur N, avec un certain $\delta = \delta(N)$ tendant vers zéro ?

M. Cette idée n'est d'aucune aide. Donne-moi quelques exemples, s'il te plaît.

O. L'algorithme glouton évident donne l'ensemble

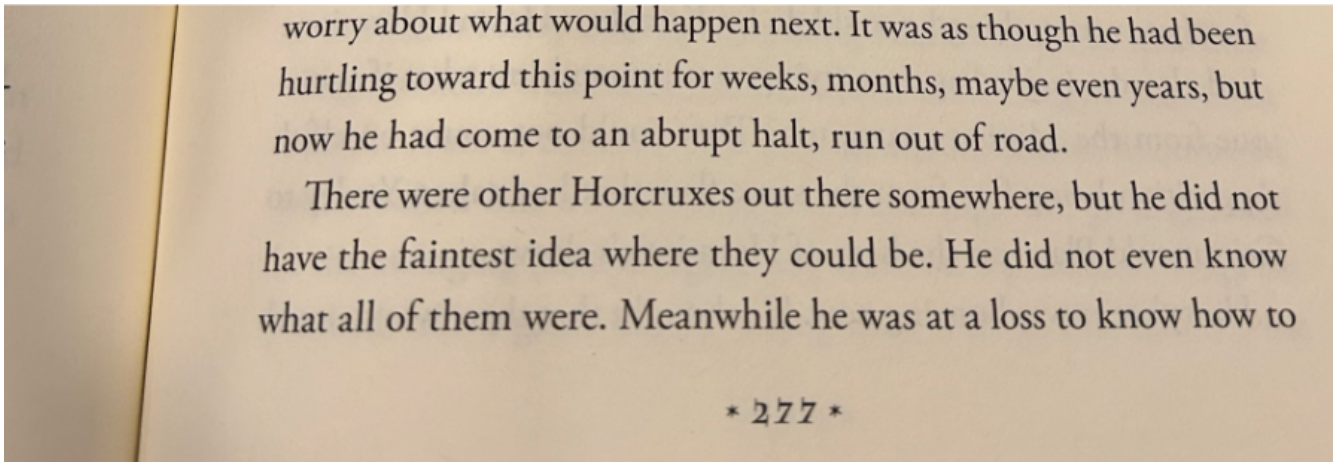
$$\{1, 2, 4, 5, 10, 11, 13, 14, 28, 29, 31, 32, 37, 38, 40, 41, \ldots\}.$$

Je remarque que de grandes parties de l'ensemble sont des translations d'autres parties. En fait, cet ensemble est très semblable à l'ensemble de Cantor, ce qui donne une borne de $\delta \geq N^{(\log 2/\log 3)-1}$.

Gowers note : "L'idée du dialogue est que l'ordinateur est très utile au mathématicien, tout en ne faisant rien de particulièrement intelligent." (souligné par moi). Donc, dans cette veine : établissons un terrain d'entente et soyons précis sur ce que j'entendrai par "grand modèle de langage" (llm).

3. Qu'est-ce qu'un llm ?

Nous avons tous fait l'expérience suivante : en lisant un livre, vous arrivez en bas d'une page au milieu d'une phrase, et alors que vos doigts tâtonnent, luttant pour tourner la page, votre esprit ne peut s'empêcher de courir en avant, imaginant comment la phrase pourrait se poursuivre. Par exemple :


worry about what would happen next. It was as though he had been hurtling toward this point for weeks, months, maybe even years, but now he had come to an abrupt halt, run out of road.

There were other Horcruxes out there somewhere, but he did not have the faintest idea where they could be. He did not even know what all of them were. Meanwhile he was at a loss to know how to

* 277 *

Figure 3.1 : Le bas d'une page de "Harry Potter et les Reliques de la Mort" de J. K. Rowling

Nous lisons que

[Harry] ne savait même pas ce qu'ils étaient tous [les Horcruxes]. Pendant ce temps, il ne savait pas comment...

Comment quoi ? Votre esprit commence instinctivement à générer des possibilités. Comment "détruire" les Horcruxes ? Ou peut-être comment "trouver" ? Ou, bien moins plausiblement, comment "transformer en sandwich" ?

J'imagine que je participe à un jeu de Family Feud, et que j'entends Steve Harvey annoncer : "D'après notre sondage !...". Mais ici, le "sondage" n'interroge pas 100 personnes au hasard - il échantillonne tout le texte humain disponible ! Les résultats pourraient ressembler à ceci : étant donné le contexte, environ 20% des gens, disons, continueraient la phrase par "détruire", 7% supplémentaires utiliseraient "trouver", et peut-être un infime $0,003\%$ suggéreraient "transformer en

sandwich".

Ceci, à la base, c'est un grand modèle de langage : c'est une fonction dont l'entrée est le "contexte" - une séquence potentiellement très longue de "tokens" (mots, signes de ponctuation, etc.) - et dont la sortie n'est pas le mot suivant (comme on le comprend souvent à tort), mais plutôt une distribution de probabilité sur tous les tokens suivants possibles.

Figure 3.2 : Un llm est une fonction allant du contexte (une séquence de tokens) à une distribution de vraisemblance des tokens suivants potentiels.

Pour nos besoins, nous pouvons mettre de côté l'ingénierie insondable qui rend une telle fonction possible : l'architecture (les transformeurs), la représentation des données (les plongements dans des espaces vectoriels de grande dimension), le processus d'entraînement (la descente de gradient, le pré-entraînement, l'ajustement fin), et ainsi de suite. Nous tenons simplement pour acquis qu'une telle fonction existe.

C'est sur la base d'un tel modèle que l'on construit le chatbot mentionné ci-dessus ; le bot doit décider comment choisir le token suivant réel à partir de la distribution de probabilité que le modèle produit. Ce choix dépend de paramètres comme la "température" - en sélectionnant peut-être toujours le token le plus probable (ce qui tend à produire une prose un peu guindée), ou en échantillonnant aléatoirement à partir de la distribution, ou diverses options intermédiaires. Une fois qu'un token est choisi, le bot l'ajoute au contexte, et renvoie la séquence entière à travers le modèle ; il choisit ensuite le token suivant, puis le suivant, et ainsi de suite, jusqu'à ce qu'il sélectionne un token spécial qui signale "arrêter de parler". (Nous reviendrons plus tard sur les comportements agentiques plus complexes, comme l'utilisation d'outils et les flux de travail auto-dirigés ; pour l'instant, je me concentre sur le mécanisme d'un chatbot "pur".) C'est peut-être ce comportement de chatbot pur que Gowers envisageait comme un processus informatique "ne faisant rien de particulièrement intelligent" dans son article auquel j'ai fait référence.

Une observation cruciale ici est que ce processus est profondément stochastique. À chaque étape, le bot fait un choix aléatoire, et chaque choix affecte la distribution de probabilité pour tous les choix ultérieurs. L'aléa se cumule à chaque token - aléa sur aléa sur aléa. Cette stochasticité est à la fois la grande force du système - lui permettant d'aborder un vaste éventail de problèmes avec une remarquable polyvalence - mais, comme nous le verrons, elle crée des défis fondamentaux lorsqu'elle est appliquée à des problèmes déterministes comme la preuve mathématique.

4. Problèmes stochastiques vs. déterministes.

Considérons la différence entre une exactitude approximative et une exactitude parfaite. Un long poème avec un mot mal choisi au milieu peut encore être magnifiquement émouvant ; de même pour un solo de jazz improvisé avec une note accidentellement dissonante. Ce sont des domaines où un succès de 99% représente une réussite authentique. Mais un théorème prouvé à 99% ? Une seule erreur peut rendre tout ce qui suit dénué de sens ! Les mathématiques exigent une exactitude déterministe, pas une approximation probabiliste.

Pour comprendre pourquoi cela importe quantitativement, supposons qu'un llm atteigne une précision de 99% à chaque étape individuelle - un taux de réussite remarquablement élevé. Si un argument mathématique nécessite d'enchaîner 1000 de ces étapes en séquence, avec le succès de chaque étape indépendant des autres, quelle est la probabilité de succès global ? La réponse : $0,99^{1000} \approx 0,004\%$. Un processus qui semble hautement fiable à chaque étape devient presque certainement erroné lorsque de nombreuses étapes se combinent.

Le cerveau humain semble également fonctionner selon deux modes assez distincts, popularisés dans *Système 1 / Système 2* de Daniel Kahneman. Le premier est rapide, automatique et sans effort ; le second est lent, délibéré et demande un effort. Lorsqu'on demande "Combien font $8 + 3$?", la plupart des adultes répondent instantanément - c'est le traitement du Système 1. Vous avez rencontré ce calcul tellement de fois que votre cerveau a essentiellement mémorisé la réponse. (C'est pourquoi l'apprentissage des tables de multiplication par les enfants est si important : il convertit ce qui était autrefois un calcul du Système 2 demandant un effort en un rappel automatique du Système 1, libérant des ressources cognitives pour des tâches plus complexes.)

Mais lorsque je calcule 437×82 sur papier, j'ai besoin du traitement du Système 2. Cependant, parce que j'ai appris l'algorithme standard, je peux décomposer ce problème en une longue chaîne de simples calculs du Système 1 : d'abord calculer $2 \times 7 = 14$, écrire le 4, retenir le 1, et ainsi de suite. L'idée-clé est que le raisonnement mathématique du Système 2 consiste souvent à enchaîner soigneusement de nombreuses étapes du Système 1, où le succès exige que chaque maillon de la chaîne soit correct.

Cela nous ramène à notre problème stochastique. Si votre "machine du Système 1" est précise à 99% à chaque étape, mais que vous devez enchaîner 1000 étapes pour résoudre un problème du Système 2, vous êtes confronté au taux de succès de $0,004\%$ mentionné ci-dessus. C'est pourquoi l'utilisation d'outils stochastiques pour les mathématiques déterministes est fondamentalement difficile.

Pourtant, cet argument suppose que les capacités de l'ia resteront statiques. Compte tenu du rythme extraordinaire des progrès de ces dernières années, ces limitations ne sont-elles peut-être que temporaires ?

5. Les capacités de l'ia progressent rapidement...

Les preuves d'une poursuite des progrès rapides sont convaincantes. Considérons les performances à l'Olympiade Internationale de Mathématiques (IMO), largement considérée comme l'une des compétitions mathématiques les plus difficiles pour les lycéens : en 2023, le nombre de systèmes d'ia capables de marquer un seul point à l'Olympiade était... zéro. En 2024, alphaproof $^+$ alphageometry de deepmind a réalisé une performance de médaille d'argent, à un point de l'or [10]. En 2025, de multiples systèmes d'ia ont revendiqué des scores de médaille d'or, environ la moitié utilisant des méthodes de vérification formelle et l'autre moitié travaillant uniquement en langage naturel.[1]

Cette trajectoire est remarquable. En deux ans, nous sommes passés d'une incapacité totale à une performance de médaille d'or. De plus, le succès en 2025 des approches purement en langage naturel démontre que la vérification formelle n'est pas strictement nécessaire pour résoudre des problèmes mathématiques difficiles - du moins au niveau de l'Olympiade.

D'ici la prochaine Olympiade à l'été 2026, il peut être raisonnable de prédire que le nombre de systèmes d'ia de pointe participant à l'Olympiade reviendra à zéro - non pas parce qu'ils ont régressé, mais parce que la compétition sera devenue trivialement facile pour eux. (Les modèles open source pourraient continuer à participer, mais les systèmes commerciaux considéreront probablement cela en dessous de leurs capacités.)

Ce modèle d'accélération des capacités n'est pas unique aux mathématiques. Il reflète ce que Richard Sutton a appelé la "Leçon amère" de soixante-dix ans de recherche en ia : les solutions spécifiques à un domaine conçues par des experts humains sont systématiquement dépassées par des méthodes générales qui exploitent des ressources informatiques toujours croissantes [23]. Encore et encore, nous avons vu que le passage à l'échelle du calcul et des données l'emporte sur les astuces d'ingénierie intelligentes.

Si la loi de Moore se poursuit - ou même si elle ralentit plutôt que de s'arrêter - nous pouvons nous attendre à ce que les llm soutiennent des chaînes de raisonnement de plus en plus longues. Les systèmes qui peinent avec des arguments de 100 étapes aujourd'hui pourraient gérer des arguments de 1000 étapes demain, et des arguments de 10 000 étapes après-demain. La transition des mathématiques de lycée au niveau licence, puis au niveau doctorat, puis au niveau de la recherche, pourrait simplement nécessiter une mise à l'échelle de la capacité de raisonnement soutenu par un multiple important.

En effet, on peut proposer une analogie approximative avec les classements aux échecs : un grand maître d'échecs avec un classement Elo de 2500 pourrait correspondre à un docteur en mathématiques fraîchement diplômé ; 2600 à un chercheur établi ; 2700 à un expert de premier plan ; et 2800+ aux géants des mathématiques. Si les systèmes d'ia peuvent atteindre le niveau équivalent de 2300 pour les mathématiques de lycée d'élite, pourquoi ne pas atteindre éventuellement 2800 et au-delà ?

Peut-être que la tension stochastique/déterministe que j'ai soulignée se dissoudra simplement à mesure que les systèmes deviendront assez puissants pour maintenir une fiabilité sur des chaînes de raisonnement arbitrairement longues. Peut-être que l'ia en langage naturel sera, après tout, suffisante pour toutes les mathématiques ? Il me semble que même ce scénario optimiste ne parvient

pas à résoudre un problème fondamental.

6. ... Mais même une mise à l'échelle parfaite est insuffisante.

Imaginons, de manière optimiste, que l'ia continue de s'améliorer à son rythme actuel. Supposons que dans une décennie, nous ayons des systèmes capables de produire des recherches mathématiques originales - non seulement de résoudre des problèmes de compétition, mais de découvrir de nouveaux théorèmes, de développer de nouvelles théories et d'écrire des articles qui font avancer la connaissance humaine.

Considérons un futur système d'ia capable de générer 100 articles de recherche par jour. Pour rendre le scénario aussi favorable que possible, supposons que ce système ait atteint une fiabilité extraordinaire : 99 de ces 100 articles sont complètement corrects, avec des preuves rigoureuses et des contributions significatives aux mathématiques. Un seul article sur 100 contient une erreur - peut-être une erreur subtile dans un lemme, ou un cas qui n'a pas été correctement traité, ou un signe "-" enfoui quelque part au fond d'un calcul.

Ferais-je confiance à un tel système ? Certainement pas.

Pour utiliser cet outil efficacement, je devrais lire les 100 articles pour déterminer lesquels sont corrects et lequel est défectueux. Mais vérifier l'exactitude d'un article mathématique peut ne pas être substantiellement plus facile que de le produire en premier lieu, en particulier pour une recherche de pointe. Il y aurait certainement une valeur à exploiter les résultats de l'outil pour des idées intéressantes, des approches nouvelles ou des connexions inattendues ; cependant, je ne pourrais pas traiter les théorèmes eux-mêmes comme des faits établis sur lesquels construire. Je devrais passer ma journée entière, chaque jour, à vérifier ces articles, pour finalement découvrir que la majeure partie de mon effort a été consacrée à vérifier un travail déjà correct, tandis que le seul article erroné pourrait passer inaperçu si son erreur est suffisamment subtile.

On pourrait objecter que cette hypothétique précision de 99% est déjà meilleure que la littérature publiée actuelle ! Après tout, la plupart des mathématiciens ont rencontré des articles avec des erreurs, des lacunes dans les arguments, ou des cas où "les détails sont laissés au lecteur". Tout cela est vrai. Mais il y a une différence cruciale : avec les articles écrits par des humains, nous développons des intuitions sur la fiabilité. Comme Kevin Buzzard l'a écrit récemment [3] : "Les llm seraient très heureux de masquer toutes les fissures de l'argument ou même de mentir pour parvenir aux affirmations du résultat et, par conséquent, je me sens beaucoup plus motivé à lire l'article correctement lorsque je pense qu'il a été généré à 100% par quelqu'un qui croyait que ce qu'il écrivait était vrai, que par quelque chose qui croyait que ce qu'il écrivait était plausible." (Souligné par moi.)

Nous développons des intuitions sur la fiabilité à travers de multiples signaux : la qualité de l'exposition, le fait que les techniques sont appliquées de manière standard, la manière dont les cas limites sont traités, le fait que les affirmations semblent plausibles compte tenu des résultats connus, etc.,

etc. Une preuve brillante d'un mathématicien inconnu peut être immédiatement reconnue comme telle précisément parce que nous pouvons évaluer ces qualités. (Pour un seul exemple, voir [27].) Avec une ia produisant 100 articles par jour à partir d'une seule source, ces heuristiques d'évaluation ne fournissent aucune différenciation ; chaque article doit être traité avec une suspicion égale.

De plus, même pour les 99 articles corrects, je ne peux pas en extraire toute leur valeur sans vérification. Un beau théorème peut suggérer de nouvelles directions de recherche, mais je ne peux pas construire en toute confiance sur lui avant d'avoir vérifié la preuve moi-même. Les idées peuvent valoir la peine d'être exploitées, les conjectures peuvent valoir la peine d'être poursuivies, mais les théorèmes eux-mêmes restent incertains tant qu'ils ne sont pas vérifiés. Et à ce stade, pourquoi ne pas simplement faire les mathématiques moi-même ?

Ce n'est pas un problème que la mise à l'échelle peut résoudre. Même si nous améliorons la précision à $99,99\%$ ou $99,999\%$, le problème fondamental subsiste : pour les mathématiques, un taux d'erreur non nul [1] crée un outil inutilisable. Si nous ne pouvons pas distinguer les résultats corrects des résultats erronés sans vérification, cela ruine tout l'intérêt de l'automatisation.

Comparons ceci à un scénario différent : une ia qui ne produit que 10 articles par an, mais chacun est accompagné d'une preuve formellement vérifiée. De plus - et c'est crucial - supposons que j'aie appris à lire les énoncés formels assez bien pour vérifier que l'outils automatique dit exactement ce que je veux qu'il dise, sans erreurs de traduction ni mauvaises interprétations. C'est un outil que j'utiliserais avec enthousiasme.

La différence est la certification déterministe. Je peux faire confiance à ces 10 articles complètement, construire sur eux en toute confiance, et incorporer leurs résultats dans mon travail sans craindre des erreurs cachées. Le rythme plus lent n'est pas seulement acceptable - c'est une caractéristique, pas un bug. La qualité et la certitude importent bien plus que la quantité lorsqu'on construit un corpus cumulatif de connaissances mathématiques.

7. Mathématiques formellement vérifiées.

Les assisteurs de preuve interactifs (également appelés "assistants de preuve") sont des systèmes logiciels qui vérifient chaque étape d'une preuve mathématique par rapport aux axiomes et aux résultats précédemment établis en utilisant les règles de l'inférence logique. Une telle technologie a été envisagée sous une certaine forme dès Hilbert, sinon Euclide, et existe en pratique depuis la fin des années 1960. Les systèmes établis incluent automath, mizar, coq (maintenant rocq), isabelle, hol light, et bien d'autres.

Parmi les plus récents se trouve lean, qui a rapidement gagné en adoption non seulement dans les cercles de l'informatique et de la logique mathématique, mais aussi dans la recherche mathématique grand public. Bon nombre des principes discutés ici s'appliquent à tout assistant de preuve, mais je me concentrerai sur lean et son écosystème. (Divulgation complète : je siège au Conseil Consultatif

1. *Note de la traductrice : pour une machine, pas pour un humain.*

Stratégique du lean Focused Research Organization.) On pourrait craindre qu'il y ait des limites inhérentes à ce qui peut être exprimé dans lean, mais comme la Conférence Plénière ICM 2022 de Buzzard [2] le soutient, le cadre formel a déjà prouvé qu'il est suffisamment large pour accommoder essentiellement toutes les mathématiques que nous pourrions vouloir formaliser ; je ne répéterai pas les preuves ici.

Il existe certaines idées fausses courantes sur ce que le terme "lean" désigne réellement, alors soyons précis.

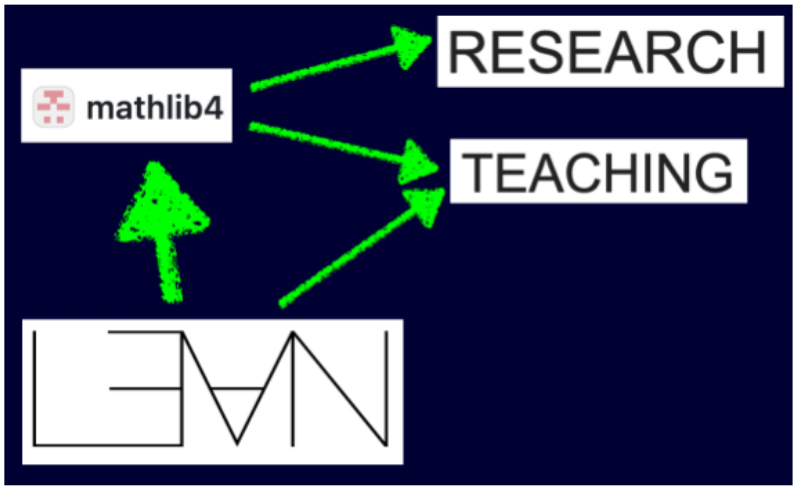


Figure 7.1 : Différents aspects de lean.

7.1. Qu'est-ce que lean ?

lean de base (maintenant techniquement lean4) est un logiciel développé à l'origine par Leonardo de Moura (alors chez Microsoft Research, maintenant chez Amazon Web Services) pour certifier un code sans bugs. Par la correspondance de Curry-Howard [4, 12], cette même technologie peut certifier qu'un énoncé mathématique a été prouvé, en vérifiant chaque étape jusqu'aux axiomes. Cela n'a rien, a priori, à voir avec l'ia.

Le flux de travail est simple : les humains écrivent des preuves dans le langage formel de lean, ligne par ligne. Après chaque ligne, lean soit accepte l'étape comme valide, soit indique une erreur. Lorsqu'une preuve est complète, lean affiche "No goals", ce qui signifie qu'il n'y a plus d'énoncés à prouver.

Voici un exemple simple :

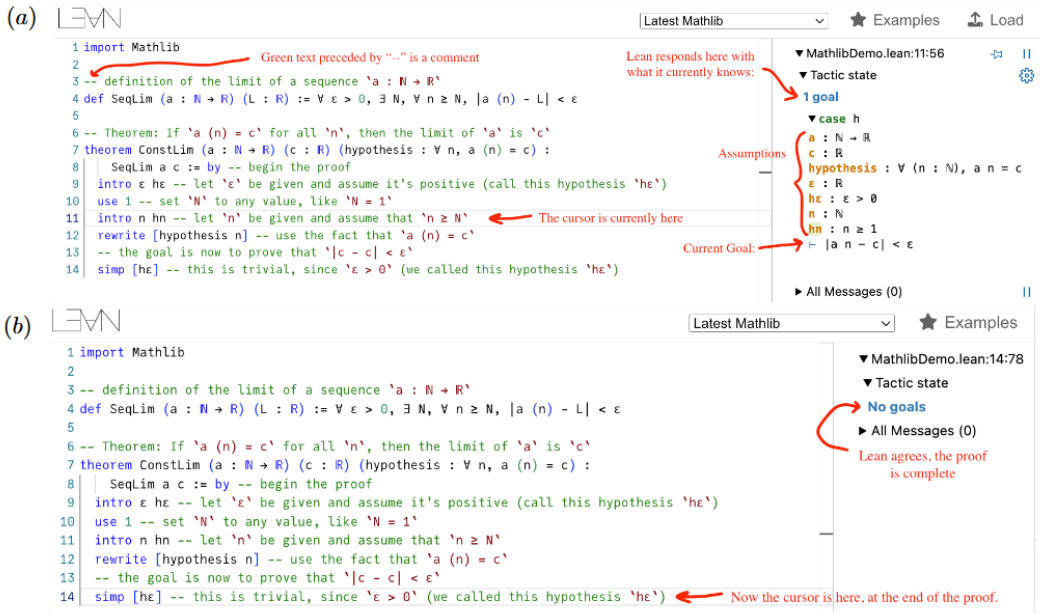
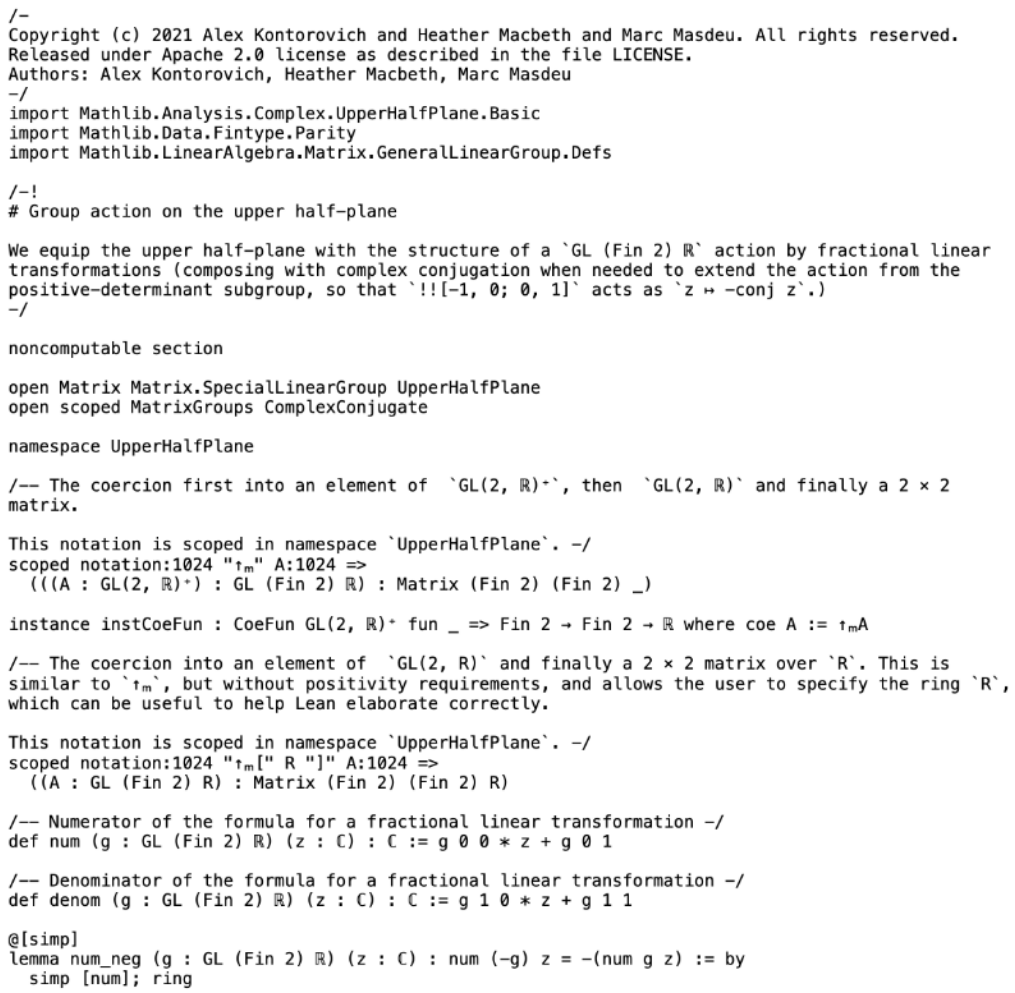


Figure 7.2 : Prouver un théorème dans lean.

7.2. Qu'est-ce que mathlib ?

Même si vous n'avez jamais essayé de prouver un théorème significatif à partir de ses axiomes, vous pouvez certainement imaginer que cela pourrait être extraordinairement fastidieux et chronophage. Pour faire des progrès, nous avons besoin d'une bibliothèque complète de résultats précédemment prouvés, énoncés dans la plus grande généralité pour être utiles dans le plus grand nombre de situations. Nous avons également besoin de "tactiques" puissantes - des procédures automatisées qui peuvent résoudre des classes entières de sous-objectifs - afin que les utilisateurs puissent faire appel à quelques-unes de ces tactiques dans leurs preuves, plutôt que d'essayer de mémoriser des milliers de noms de théorèmes individuels.

C'est mathlib : une bibliothèque massive, interconnectée et maintenable construite sur lean. Comme lean lui-même, mathlib est entièrement open source, permettant l'effort collaboratif de centaines de contributeurs dans le monde entier. Lorsque mes amis mathématiciens se plaignent d'avoir regardé mathlib et de n'avoir rien compris bien qu'ils soient des experts dans le domaine concerné, j'explique que mathlib n'est pas destiné à être un projet éducatif. Son objectif unique est l'exhaustivité et la capacité de mise à l'échelle.



```
/-
Copyright (c) 2021 Alex Kontorovich and Heather Macbeth and Marc Masdeu. All rights reserved.
Released under Apache 2.0 license as described in the file LICENSE.
Authors: Alex Kontorovich, Heather Macbeth, Marc Masdeu
-/
import Mathlib.Analysis.Complex.UpperHalfPlane.Basic
import Mathlib.Data.Fintype.Parity
import Mathlib.LinearAlgebra.Matrix.GeneralLinearGroup.Defs

/-!
# Group action on the upper half-plane

We equip the upper half-plane with the structure of a `GL (Fin 2) ℝ` action by fractional linear
transformations (composing with complex conjugation when needed to extend the action from the
positive-determinant subgroup, so that `!![-1, 0; 0, 1]` acts as `z ↦ -conj z`.)
-/

noncomputable section

open Matrix Matrix.SpecialLinearGroup UpperHalfPlane
open scoped MatrixGroups ComplexConjugate

namespace UpperHalfPlane

/-- The coercion first into an element of  `GL(2, ℝ)⁺`, then  `GL(2, ℝ)` and finally a 2 × 2
matrix.

This notation is scoped in namespace `UpperHalfPlane`. -/
scoped notation:1024 "↑ₘ" A:1024 =>
  (((A : GL(2, ℝ)⁺) : GL (Fin 2) ℝ) : Matrix (Fin 2) (Fin 2) _)

instance instCoeFun : CoeFun GL(2, ℝ)⁺ fun _ => Fin 2 → Fin 2 → ℝ where coe A := ↑ₘA

/-- The coercion into an element of  `GL(2, R)` and finally a 2 × 2 matrix over `R`. This is
similar to `↑ₘ`, but without positivity requirements, and allows the user to specify the ring `R`,
which can be useful to help Lean elaborate correctly.

This notation is scoped in namespace `UpperHalfPlane`. -/
scoped notation:1024 "↑ₘ[" R "]" A:1024 =>
  ((A : GL (Fin 2) R) : Matrix (Fin 2) (Fin 2) R)

/-- Numerator of the formula for a fractional linear transformation -/
def num (g : GL (Fin 2) ℝ) (z : ℂ) : ℂ := g 0 0 * z + g 0 1

/-- Denominator of the formula for a fractional linear transformation -/
def denom (g : GL (Fin 2) ℝ) (z : ℂ) : ℂ := g 1 0 * z + g 1 1

@[simp]
lemma num_neg (g : GL (Fin 2) ℝ) (z : ℂ) : num (-g) z = -(num g z) := by
  simp [num]; ring
```

Figure 7.3 : Un extrait d'un fichier exemple [15] dans mathlib

Considérons quelqu'un travaillant sur les groupes algébriques de type Lie. La bibliothèque des variétés doit s'intégrer parfaitement avec les bibliothèques sur les groupes et la géométrie algébrique, sans conflits d'"espaces de noms" ni conventions incompatibles. Les mainteneurs de mathlib travaillent extrêmement dur pour garantir que vous puissiez importer toutes ces bibliothèques simultanément et vous mettre au travail.

mathlib doit passer de son ordre de grandeur actuel (environ 2 millions de lignes de code) à 10 millions, à 100 millions de lignes et au-delà, tout en maintenant une réactivité interactive. Si lean prend plus de quelques secondes pour répondre après que vous ayez écrit une ligne de code, il devient inutilisable pour la recherche mathématique.

Il est important de souligner, comme illustré dans la Figure 7.1, que s'engager dans les mathématiques formelles ne nécessite pas de contribuer à mathlib lui-même. Si les fondations mathématiques dont vous avez besoin sont déjà disponibles dans mathlib, vous pouvez mener des recherches en construisant sur la bibliothèque sans nécessairement viser à l'étendre. De même, mathlib sert de ressource pour l'enseignement - bien qu'à des fins pédagogiques, il soit souvent préférable de travailler

directement à partir de la bibliothèque de base de lean, car les traitements très généraux et sophistiqués de mathlib peuvent être trop avancés pour les étudiants qui découvrent le sujet ; voir § 10.

Nous reviendrons sur ces deux questions plus en détail plus tard. Pour l'instant, qu'il suffise de dire que les trois applications - la recherche utilisant des formalisations existantes, l'enseignement avec des niveaux d'abstraction appropriés, et l'extension de mathlib avec de nouvelles mathématiques - sont des activités distinctes qui servent toutes des communautés et des objectifs différents. Le rôle de mathlib est de fournir l'infrastructure fondamentale qui rend les deux autres possibles.

7.3. Le fossé des bibliothèques et l'assistance de l'ia.

Nous pouvons maintenant rappeler et mieux apprécier l'énigme fondamentale de l'introduction : les mathématiques de recherche croissent à un taux δ tandis que mathlib croît à un taux ϵ, et actuellement $\delta > \epsilon$. Les 2 millions de lignes de code de mathlib représentent une réalisation impressionnante, mais de vastes domaines des mathématiques restent non formalisés.

C'est là que l'assistance de l'ia devient essentielle. Je parle maintenant d'un objectif beaucoup plus modeste que de prouver de nouveaux théorèmes : si l'ia peut aider à ce qu'on appelle l'"autoformalisation", la conversion automatique des mathématiques en langage naturel déjà établies en preuves formelles, alors ϵ pourrait au moins rattraper δ. Et ensuite, une fois que mathlib sera suffisamment complet, nous pourrions enfin obtenir que ϵ dépasse réellement δ, ce qui permettrait aux mathématiciennes et mathématiciens de travailler directement dans le système formel pour découvrir de nouvelles mathématiques.

Comment une telle formalisation assistée par l'ia pourrait-elle fonctionner en pratique ?

8. Une voie à suivre : la quasi-autoformalisation.

La réponse, je crois, pourrait résider dans ce que j'appelle la "quasi-autoformalisation" : un système d'agents d'ia coopérants, orchestrés pour traduire les mathématiques en langage naturel en preuves formelles. Le mot clé ici est "quasi." Pour mes besoins en tant que mathématicien chercheur (plutôt que chercheur en ia), je n'exige pas que le système fonctionne de manière complètement autonome. Je suis parfaitement heureux d'intervenir chaque fois que je vois un moyen rapide d'élaguer l'arbre de recherche et de faire avancer le processus. Mon objectif n'est pas une automatisation complète mais plutôt un flux de travail collaboratif et considérablement accéléré. Voici un aperçu général, avant d'entrer dans les détails :

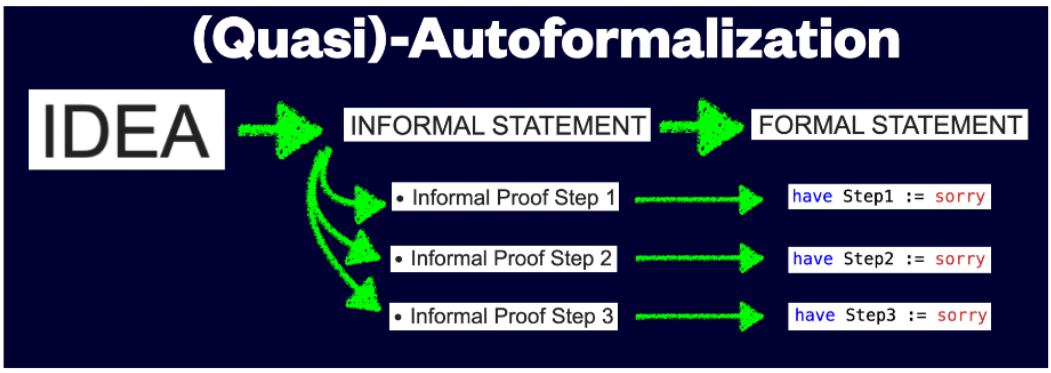


Figure 8.1 : Un schéma de la quasi-autoformalisation

Notons qu'une variété d'approches de la formalisation assistée par l'ia ont été explorées dans le passé, depuis les premiers travaux sur la démonstration automatique de théorèmes avec des résultats lisibles par l'homme [7] jusqu'aux récents systèmes basés sur des llm comme Draft-Sketch-Prove [13] (centré sur la résolution de nouveaux problèmes à partir de zéro) et des outils comme lean copilot [22] (fournissant des suggestions dans l'assistant de preuve). Le cadre de quasi-autoformalisation que je propose diffère par sa décomposition multi-agents explicite avec un raffinement itératif, conçu spécifiquement pour accélérer la croissance de mathlib plutôt que (nécessairement) d'atteindre une autonomie complète.

8.1. L'architecture : Quatre agents.

Le système se compose de quatre composants travaillant ensemble :

- le décomposeur : prend les mathématiques au niveau des "idées" et les affine en énoncés en langage naturel et en étapes de preuve à un niveau de granularité approprié.
- l'aducteur : convertit les énoncés en langage naturel et les étapes de preuve en code lean formel, produisant à la fois l'énoncé du théorème et un "échafaudage" d'énoncés `have` (affirmations intermédiaires) qui sont initialisés à `sorry` (c'est-à-dire que leurs preuves sont reportées sans déclencher de messages d'erreur lean).
- le solveur (ou "fermeur") : tente de prouver chaque énoncé `have`, remplaçant `sorry` par des preuves formelles complètes.
- le conducteur : orchestre l'ensemble du processus, identifie les échecs et décide quels composants doivent être ré-exécutés avec des paramètres ajustés.

Examinons chaque composant en détail.

8.1..1 Le décomposeur : des idées à l'échafaudage.

Les mathématiques en langage naturel, que ce soit dans les manuels de licence ou les articles arXiv, sont vraiment écrites au niveau des "Idées". Même quelque chose d'aussi clair (pour un mathématicien entraîné) qu'une preuve dans le Rudin est loin d'être assez précis pour être formalisé directement.

1.1 Example We now show that the equation

(1) $$p^2 = 2$$

is not satisfied by any rational p. If there were such a p, we could write $p = m/n$ where m and n are integers that are not both even. Let us assume this is done. Then (1) implies

(2) $$m^2 = 2n^2,$$

This shows that m^2 is even. Hence m is even (if m were odd, m^2 would be odd), and so m^2 is divisible by 4. It follows that the right side of (2) is divisible by 4, so that n^2 is even, which implies that n is even.

The assumption that (1) holds thus leads to the conclusion that both m and n are even, contrary to our choice of m and n. Hence (1) is impossible for rational p.

Figure 8.2 : Extrait de Rudin [21, p.2]

Formaliser cet argument simple nécessite beaucoup plus d'étapes qui sont cachées dans le langage naturel parce qu'elles sont trop évidentes pour un humain pour être énoncées, mais sont nécessaires pour l'exhaustivité, telles que : pourquoi pouvons-nous faire en sorte que m et n ne soient pas tous les deux pairs ? Pourquoi si m n'est pas de la forme $2k$, alors il doit être de la forme $2k + 1$? Etc.

Le travail du décomposeur est de décomposer les idées mathématiques en étapes suffisamment petites et explicites pour être formalisées. La granularité dépend de ce qui est déjà dans mathlib et de la sophistication du solveur. C'est un processus itératif : si le solveur échoue, le conducteur peut demander au décomposeur de décomposer davantage. La Figure 8.3 montre un exemple de décomposition dans cet exemple, par l'ia claude.

8.1..2 Le traducteur.

Le travail du traducteur est de prendre les petits lemmes en langage naturel du décomposeur, et de convertir chacun en un petit énoncé lean `have` formel, donnant un échafaudage formel du théorème principal. La Figure 8.4 montre à quoi cela pourrait ressembler dans notre exemple courant, en utilisant à nouveau l'ia claude. Nous reportons au § 8.3.2 une discussion plus détaillée des nombreuses difficultés que cet agent particulier rencontrera.

8.1..3 Le solveur : clore les objectifs.

Une fois que le traducteur produit un échafaudage avec des énoncés `have sorry`, le solveur tente de vider chaque `sorry`. C'est là que les systèmes de démonstration automatique de théorèmes démontrent leur puissance : ils peuvent rechercher dans de vastes espaces de stratégies de preuve possibles, enchaîner des lemmes de mathlib, et appliquer des tactiques jusqu'à ce que les objectifs soient clos.

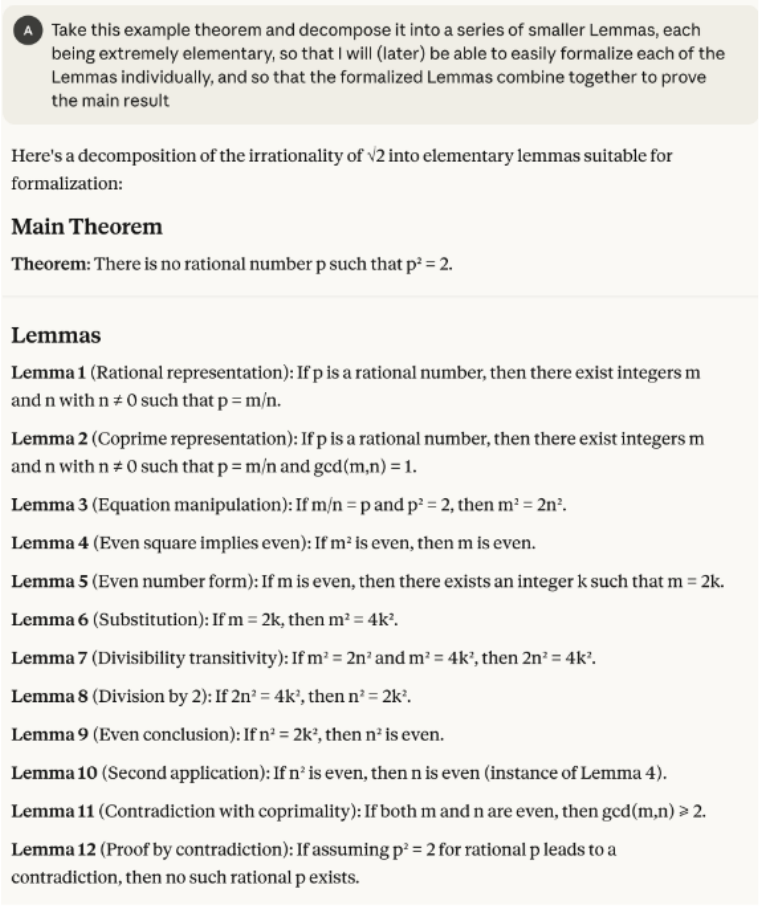
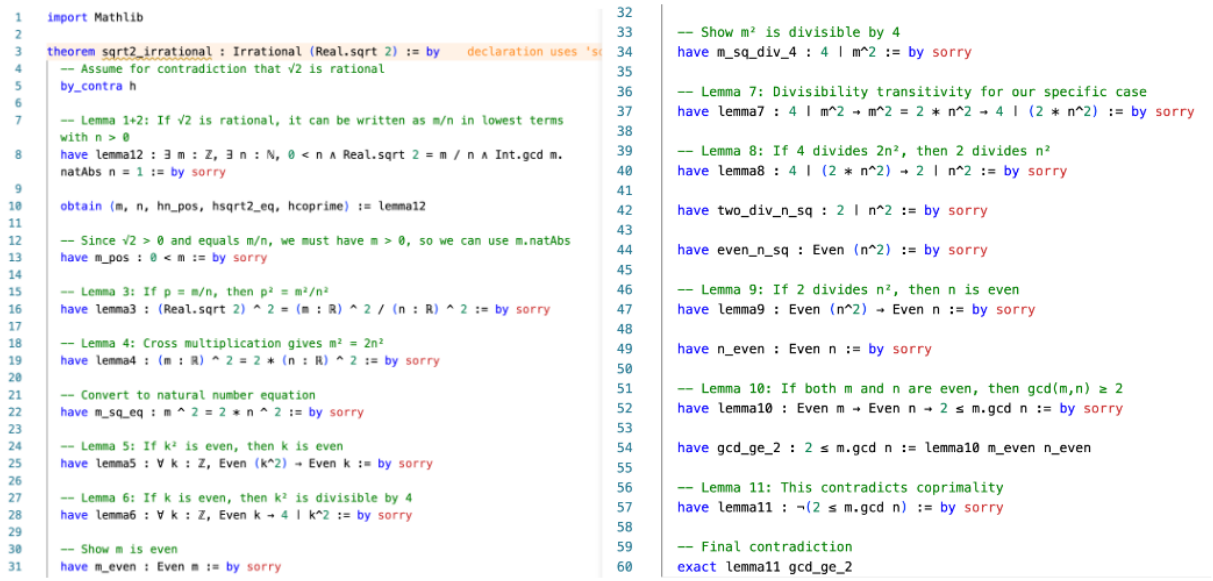


Figure 8.3 : Une décomposition de l'argument du Rudin en utilisant l'ia claude

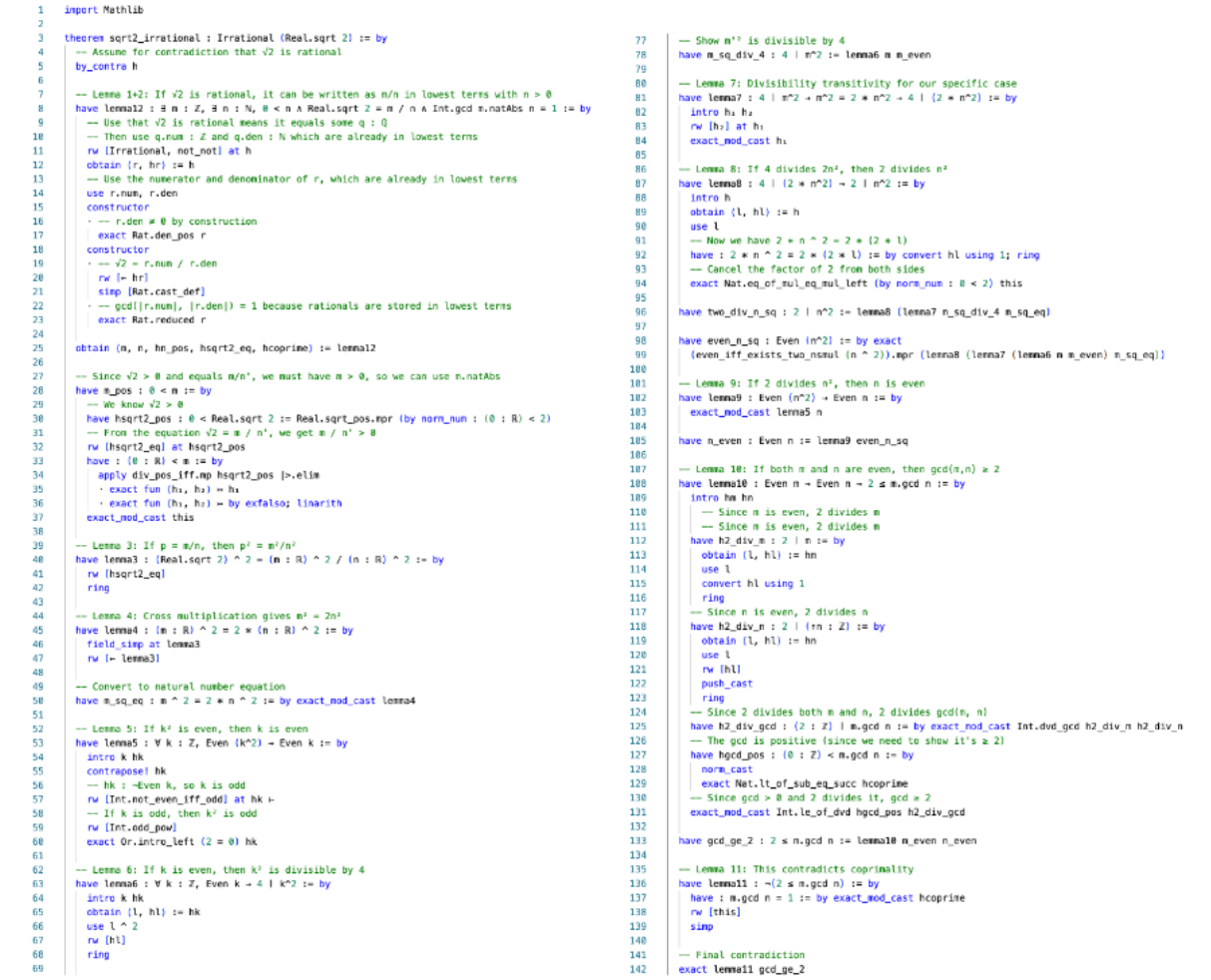
Le solveur n'a pas besoin de réussir sur chaque objectif. Certains `sorry` peuvent subsister, nécessitant soit une intervention humaine, une décomposition supplémentaire par le décomposeur, soit un raffinement de l'énoncé par le traducteur. La valeur du système réside dans la fermeture automatique des objectifs qui peuvent être clos, réduisant considérablement l'effort humain requis.

Différentes stratégies de résolution peuvent être appropriées pour différents types d'objectifs. Des manipulations algébriques simples peuvent céder à des tactiques de base, tandis que des arguments plus subtils peuvent nécessiter des procédures de recherche sophistiquées ou même des approches d'apprentissage par renforcement entraînées sur de grands corpus de preuves formelles. La Figure 8.5 montre l'application du solveur à notre exemple (avec une certaine aide de cet humain).



```
import Mathlib

theorem sqrt2_irrational : Irrational (Real.sqrt 2) := by    declaration uses 'sc
  -- Assume for contradiction that √2 is rational
  by_contra h

  -- Lemma 1+2: If √2 is rational, it can be written as m/n in lowest terms
  with n > 0
  have lemma12 : ∃ m : ℤ, ∃ n : ℕ, 0 < n ∧ Real.sqrt 2 = m / n ∧ Int.gcd m.
  natAbs n = 1 := by sorry

  obtain (m, n, hn_pos, hsqrt2_eq, hcoprime) := lemma12

  -- Since √2 > 0 and equals m/n, we must have m > 0, so we can use m.natAbs
  have m_pos : 0 < m := by sorry

  -- Lemma 3: If p = m/n, then p² = m²/n²
  have lemma3 : (Real.sqrt 2) ^ 2 = (m : ℝ) ^ 2 / (n : ℝ) ^ 2 := by sorry

  -- Lemma 4: Cross multiplication gives m² = 2n²
  have lemma4 : (m : ℝ) ^ 2 = 2 * (n : ℝ) ^ 2 := by sorry

  -- Convert to natural number equation
  have m_sq_eq : m ^ 2 = 2 * n ^ 2 := by sorry

  -- Lemma 5: If k² is even, then k is even
  have lemma5 : ∀ k : ℤ, Even (k^2) → Even k := by sorry

  -- Lemma 6: If k is even, then k² is divisible by 4
  have lemma6 : ∀ k : ℤ, Even k → 4 ∣ k^2 := by sorry

  -- Show m is even
  have m_even : Even m := by sorry


  -- Show m² is divisible by 4
  have m_sq_div_4 : 4 ∣ m^2 := by sorry

  -- Lemma 7: Divisibility transitivity for our specific case
  have lemma7 : 4 ∣ m^2 → m^2 = 2 * n^2 → 4 ∣ (2 * n^2) := by sorry

  -- Lemma 8: If 4 divides 2n², then 2 divides n²
  have lemma8 : 4 ∣ (2 * n^2) → 2 ∣ n^2 := by sorry

  have two_div_n_sq : 2 ∣ n^2 := by sorry

  have even_n_sq : Even (n^2) := by sorry

  -- Lemma 9: If 2 divides n², then n is even
  have lemma9 : Even (n^2) → Even n := by sorry

  have n_even : Even n := by sorry

  -- Lemma 10: If both m and n are even, then gcd(m,n) ≥ 2
  have lemma10 : Even m → Even n → 2 ≤ m.gcd n := by sorry

  have gcd_ge_2 : 2 ≤ m.gcd n := lemma10 m_even n_even

  -- Lemma 11: This contradicts coprimality
  have lemma11 : ¬(2 ≤ m.gcd n) := by sorry

  -- Final contradiction
  exact lemma11 gcd_ge_2
```

Figure 8.4



```
import Mathlib

theorem sqrt2_irrational : Irrational (Real.sqrt 2) := by
  -- Assume for contradiction that √2 is rational
  by_contra h

  -- Lemma 1+2: If √2 is rational, it can be written as m/n in lowest terms with n > 0
  have lemma12 : ∃ m : ℤ, ∃ n : ℕ, 0 < n ∧ Real.sqrt 2 = m / n ∧ Int.gcd m.natAbs n = 1 := by
    -- Use that √2 is rational means it equals some q : ℚ
    -- Then use q.num : ℤ and q.den : ℕ which are already in lowest terms
    rw [Irrational, not_not] at h
    obtain ⟨r, hr⟩ := h
    -- Use the numerator and denominator of r, which are already in lowest terms
    use r.num, r.den
    constructor
    · -- r.den ≠ 0 by construction
      exact Rat.den_pos r
    constructor
    · -- √2 = r.num / r.den
      rw [← hr]
      simp [Rat.cast_def]
    · -- gcd(|r.num|, |r.den|) = 1 because rationals are stored in lowest terms
      exact Rat.reduced r

  obtain ⟨m, n, hn_pos, hsqrt2_eq, hcoprime⟩ := lemma12

  -- Since √2 > 0 and equals m/n', we must have m > 0, so we can use m.natAbs
  have m_pos : 0 < m := by
    -- We know √2 > 0
    have hsqrt2_pos : 0 < Real.sqrt 2 := Real.sqrt_pos.mpr (by norm_num : (0 : ℝ) < 2)
    -- From the equation √2 = m / n', we get m / n' > 0
    rw [hsqrt2_eq] at hsqrt2_pos
    have : (0 : ℝ) < m := by
      apply div_pos_iff.mp hsqrt2_pos |>.elim
      · exact fun ⟨h₁, h₂⟩ ↦ h₁
      · exact fun ⟨h₁, h₂⟩ ↦ by exfalso; linarith
    exact_mod_cast this

  -- Lemma 3: If p = m/n, then p² = m²/n²
  have lemma3 : (Real.sqrt 2) ^ 2 = (m : ℝ) ^ 2 / (n : ℝ) ^ 2 := by
    rw [hsqrt2_eq]
    ring

  -- Lemma 4: Cross multiplication gives m² = 2n²
  have lemma4 : (m : ℝ) ^ 2 = 2 * (n : ℝ) ^ 2 := by
    field_simp at lemma3
    rw [← lemma3]

  -- Convert to natural number equation
  have m_sq_eq : m ^ 2 = 2 * n ^ 2 := by exact_mod_cast lemma4

  -- Lemma 5: If k² is even, then k is even
  have lemma5 : ∀ k : ℤ, Even (k^2) → Even k := by
    intro k hk
    contrapose! hk
    -- hk : ¬Even k, so k is odd
    rw [Int.not_even_iff_odd] at hk ⊢
    -- If k is odd, then k² is odd
    rw [Int.odd_pow]
    exact ⟨or_intro_left (2 = 0) hk⟩

  -- Lemma 6: If k is even, then k² is divisible by 4
  have lemma6 : ∀ k : ℤ, Even k → 4 ∣ k^2 := by
    intro k hk
    obtain ⟨l, hl⟩ := hk
    use l ^ 2
    rw [hl]
    ring


  -- Show m'² is divisible by 4
  have m_sq_div_4 : 4 ∣ m^2 := lemma6 m m_even

  -- Lemma 7: Divisibility transitivity for our specific case
  have lemma7 : 4 ∣ m^2 → m^2 = 2 * n^2 → 4 ∣ (2 * n^2) := by
    intro h₁ h₂
    rw [h₂] at h₁
    exact_mod_cast h₁

  -- Lemma 8: If 4 divides 2n², then 2 divides n²
  have lemma8 : 4 ∣ (2 * n^2) → 2 ∣ n^2 := by
    intro h
    obtain ⟨l, hl⟩ := h
    use l
    -- Now we have 2 * n ^ 2 = 2 * (2 * l)
    have : 2 * n ^ 2 = 2 * (2 * l) := by convert hl using 1; ring
    -- Cancel the factor of 2 from both sides
    exact Nat.eq_of_mul_eq_mul_left (by norm_num : 0 < 2) this

  have two_div_n_sq : 2 ∣ n^2 := lemma8 ⟨lemma7 n_sq_div_4 m_sq_eq⟩

  have even_n_sq : Even (n^2) := by exact
    (even_iff_exists_two_nsmul (n ^ 2)).mpr ⟨lemma8 ⟨lemma7 ⟨lemma6 m m_even⟩ m_sq_eq⟩⟩

  -- Lemma 9: If 2 divides n², then n is even
  have lemma9 : Even (n^2) → Even n := by
    exact_mod_cast lemma5 n

  have n_even : Even n := lemma9 even_n_sq

  -- Lemma 10: If both m and n are even, then gcd(m,n) ≥ 2
  have lemma10 : Even m → Even n → 2 ≤ m.gcd n := by
    intro hm hn
    -- Since m is even, 2 divides m
    -- Since m is even, 2 divides m
    have h2_div_m : 2 ∣ m := by
      obtain ⟨l, hl⟩ := hm
      use l
      convert hl using 1
      ring
    -- Since n is even, 2 divides n
    have h2_div_n : 2 ∣ (↑n : ℤ) := by
      obtain ⟨l, hl⟩ := hn
      use l
      rw [hl]
      push_cast
      ring
    -- Since 2 divides both m and n, 2 divides gcd(m, n)
    have h2_div_gcd : (2 : ℤ) ∣ m.gcd n := by exact_mod_cast Int.dvd_gcd h2_div_m h2_div_n
    -- The gcd is positive (since we need to show it's ≥ 2)
    have hgcd_pos : (0 : ℤ) < m.gcd n := by
      norm_cast
      exact Nat.lt_of_sub_eq_succ hcoprime
    -- Since gcd > 0 and 2 divides it, gcd ≥ 2
    exact_mod_cast Int.le_of_dvd hgcd_pos h2_div_gcd

  have gcd_ge_2 : 2 ≤ m.gcd n := lemma10 m_even n_even

  -- Lemma 11: This contradicts coprimality
  have lemma11 : ¬(2 ≤ m.gcd n) := by
    have : m.gcd n = 1 := by exact_mod_cast hcoprime
    rw [this]
    simp

  -- Final contradiction
  exact lemma11 gcd_ge_2
```

Figure 8.5

8.1..4 Le conducteur : orchestrer le flux de travail.

Le conducteur est le coordinateur du système. Lorsque la formalisation échoue (comme c'est inévitablement le cas lors des premières tentatives), le conducteur doit diagnostiquer pourquoi :

- un objectif est-il au-delà des capacités actuelles du solveur ? L'outil doit le signaler pour une décomposition supplémentaire.
- un énoncé a-t-il été mal traduit ? L'outil doit demander au traducteur d'essayer à nouveau avec un contexte supplémentaire.
- ce résultat existe-t-il peut-être déjà dans mathlib ? L'outil doit effectuer une recherche Google (via RAG - génération augmentée par récupération - ou d'autres méthodes) pour vérifier si le théorème (ou quelque chose d'étroitement lié) est déjà formalisé dans la dernière version de la bibliothèque.

Le conducteur gère également les ressources de calcul, décidant quand ré-exécuter des composants par rapport à demander une contribution humaine. Dans l'approche "quasi", un humain peut servir de super-conducteur, prenant ces décisions sur la base d'une intuition mathématique.

Dans le cas présent (voir Figure 8.6), la recherche RAG du conducteur dans la bibliothèque aurait pu trouver un théorème appelé `irrational_sqrt_two`, réduisant ainsi 140 lignes de code à une seule citation de la preuve beaucoup plus élégante de mathlib.

```
1   import Mathlib
2
3   theorem sqrt2_irrational : Irrational (Real.sqrt 2) := by
4     exact irrational_sqrt_two
```

Figure 8.6

8.2. Optimisme...

Ce flux de travail est-il vraiment efficace ? La première fois que j'ai dessiné le schéma de la Figure 8.1, dès qu'il est devenu clair dans mon esprit, c'était en mai 2025 à l'Institute for Advanced Study (où je passais mon année sabbatique) pendant un déjeuner avec Thomas Hubert de deepmind, un leader du projet alphaproof, et Sanjeev Arora et Chi Jin de Princeton, leaders du solveur open source Goedel prover [17, 18]. Alphaproof et Goedel prover représentent tous deux l'état de l'art actuel en démonstration automatique de théorèmes, ce dernier concourant dans la catégorie Open Source. La veille, j'ai eu l'occasion de tester alphaproof avec Thomas sur certaines tâches en suspens dans le "PrimeNumberTheoremPlus Project" (pnt+) que Terry Tao et moi co-organisons [16]. L'objectif du projet est de formaliser des résultats significatifs en théorie analytique des nombres, y compris le théorème des nombres premiers, le théorème de Dirichlet sur les nombres premiers dans les progressions arithmétiques, et le théorème de densité de Chebotarev. Voir un véritable "solveur" en action pour la première fois a rendu le potentiel de cette approche soudainement concret.

Le solveur fait face à une tâche véritablement difficile : même des objectifs qui semblent triviaux à un mathématicien humain peuvent nécessiter un raisonnement formel complexe. Par exemple, j'avais échafaudé un lemme nécessitant un énoncé `have` que la fonction zêta de Riemann est au moins 1 en valeur absolue dans un voisinage de $s = 1$ (excluant le point lui-même). Cela devrait être "immédiat" pour un humain : cela découle de `riemannZeta_residue_one`, le théorème déjà dans mathlib énonçant que ζ a un pôle en $s = 1$ (de résidu 1). Mais si prouver que $\sqrt{2}$ est irrationnel a pris 140 lignes de code formel, on ne peut qu'imaginer combien cette tâche "triviale" devient plus compliquée.

Et pourtant alphaproof a réussi brillamment ! Malgré le fait d'avoir été entraîné uniquement sur des problèmes d'Olympiade de lycée (qui n'impliquent jamais de "filtres éventuels", de limites ou d'analyse complexe), il a enchaîné une séquence extrêmement longue d'étapes formelles que lean a acceptées. La preuve résultante était à peine compréhensible, même pour un formalisateur humain expert, voir Figure 8.7.

```
have ZetaBlowsUp : ∀ᶠ s in 𝓝[≠](1 : ℂ), ‖ζ s‖ ≥ 1 := by
  simp_all[Function.comp_def,eventually_nhdsWithin_iff,norm_eq_sqrt_real_inner]
  contrapose! h
  simp_all
  delta abs at*
  exfalso
  simp_rw [Metric.nhds_basis_ball.frequently_iff]at*
  choose! I A B using h
  choose a s using exists_seq_strictAnti_tendsto (0: ℝ)
  apply((isCompact_closedBall _ _).isSeqCompact fun and=>(A _ (s.2.1 and)).le.trans
    (s.2.2.bddAbove_range.some_mem (and, rfl))).elim
  use fun and (a, H, S, M)=>absurd (tendsto_nhds_unique M (tendsto_sub_nhds_zero_iff.1
    (( squeeze_zero_norm fun and=>le_of_lt (A _ (s.2.1 _) ) ) (s.2.2.comp S.tendsto_atTop))))
    fun and=>?_
  norm_num[*,Function.comp_def] at M
  have:=@riemannZeta_residue_one
  use one_ne_zero (tendsto_nhds_unique (this.comp (tendsto_nhdsWithin_iff.2 ⟨ M,.of_forall
    (by norm_num[*]))⟩)) ⟨ squeeze_zero_norm ?_ ((M.sub_const 1).norm.trans
    (by rw [sub_self,norm_zero]))⟩)
  use fun and =>.trans (norm_mul_le_of_le ↑(le_rfl) (Complex.norm_def _▸Real.sqrt_le_one.mpr
  (B ↑_ (s.2.1 ↑_)).right.le)) (by rw [mul_one])
```

Figure 8.7 : solution d'alphaproof à un énoncé have

Cela semble refléter le mécanisme d'entraînement d'alphaproof : en adaptant le protocole alphazero d'apprentissage par renforcement, il "joue" aux mathématiques contre lui-même, découvrant des modèles hautement non standard pour prouver des théorèmes par auto-amélioration. Comme mon travail n'était pas (alors) destiné à contribuer directement à mathlib, je me fichais de l'apparence de la preuve formelle, tant que le code lean compilait. Puisque je savais que le résultat découlait trivialement en langage naturel de théorèmes déjà prouvés, l'acceptation par lean était suffisante, et je pouvais passer à l'objectif suivant.

En septembre 2025, peu avant l'achèvement de ce document, morph ai (maintenant math, Inc.), dirigé par Jesse Han, Jared Lichtman et Christian Szegedy, a annoncé une autre étape significa-

tive : leur agent "Gauss"[2] a autoformalisé en lean le terme d'erreur classique pour le théorème des nombres premiers. Il s'est appuyé sur les fondations de pnt+, et a fait des progrès substantiels en formalisant des résultats d'analyse complexe tels que le théorème de Borel-Carathéodory et en l'étendant aux dérivées logarithmiques des fonctions méromorphes. Dans leur expérience, un mathématicien (Jared) a servi de décomposeur tandis que Gauss gérait la traduction et la résolution de manière autonome - une division du travail qui pourrait elle-même être entièrement automatisée dans les itérations futures.

Bien que ces succès indiquent la viabilité du cadre, des défis significatifs subsistent.

8.3. ... Et difficultés.

Bien sûr, il n'est généralement pas si simple d'obtenir que ce flux de travail réussisse ; soulignons un certain nombre de difficultés particulières.

8.3.1. Le défi de clore les objectifs.

Comme nous l'avons vu avec le succès d'alphaproof sur l'objectif de la fonction zêta, il est possible de clore les objectifs formels, mais cela nécessite des systèmes sophistiqués. Une caractéristique d'ingénierie particulière s'est avérée particulièrement précieuse ici :

Étant donné un objectif quelconque, alphaproof recherche simultanément une preuve de l'énoncé et de sa négation. La motivation initiale des concepteurs de cette ingénierie était pratique. De nombreux problèmes de l'Olympiade demandent des réponses, pas seulement des preuves. Si vous ne connaissez pas l'énoncé précis que vous essayez de prouver, vous ne pouvez même pas commencer à travailler formellement. Par exemple, un problème peut demander l'ensemble des entiers satisfaisant une certaine propriété.

Vous pourriez formaliser cet énoncé comme $\exists S \subseteq \mathbb{Z}, S = \{n \in \mathbb{Z} \mid \text{propriete}(n)\}$. Mais ce défi est trivialement résoluble : posez simplement $S := \{n \in \mathbb{Z} \mid \text{propriete}(n)\}$, et par définition, S est égal à lui-même ! Pour faire un énoncé non trivial, il faut d'abord déterminer ce qu'est S, disons $S = \{1, 2, 4\}$, et seulement ensuite essayer de prouver que $\{1, 2, 4\} = \{n \in \mathbb{Z} \mid \text{propriete}(n)\}$.

L'approche d'alphaproof est la suivante : on envoie le problème en langage naturel à un llm comme gemini, en demandant environ 100 solutions potentielles ($S = \{1, 2, 3\}$, $S = \{2, 3, 4\}$, etc.). Pour chaque supposition, alphaproof tente de prouver ou de réfuter l'affirmation. En pratique, il réfute rapidement environ 98 d'entre elles, puis se concentre sur les candidates restantes.

2. Note de la traductrice : ça, ça devrait être interdit, donner à un vulgaire logiciel le nom de Gauss ; je qualifierais sans plaisanter ce procédé de "blasphème mathématique" (on note aussi que leur boîte s'appelle Math Inc, "Les maths, c'est nous", carrément !) et même si dans l'expression "blasphème mathématique", le mot "blasphème" est un terme à connotation religieuse (on sait que la seule foi des mathématiciennes et mathématiciens est la foi en la vérité), j'émets le vœu pieux, en tant que "croyante en l'humanité", que les noms des humains soient protégés des entreprises lucratives d'une manière quelconque.

Cette fonctionnalité s'est avérée inestimable dans nos expériences pour une raison complètement différente : elle détecte les erreurs de traduction.

8.3.2. Les subtilités de la traduction.

Le composant traducteur présente des défis particulièrement subtils. Obtenir que les énoncés formels disent exactement ce que nous voulons dire est parfois aussi difficile que de les prouver ! Dans mes expériences avec alphaproof sur le projet PNT+, un nombre embarrassant d'énoncés `have` que j'ai échafaudés ont été résolument réfutés par le système. Chaque fois, je pouvais généralement identifier rapidement quelle hypothèse j'avais omise, ajuster l'énoncé et itérer. Ce va-et-vient entre la proposition d'énoncés et la vérification du fait qu'ils soient établissables (ou du fait qu'ils soient réfutables) était essentiel pour obtenir la formalisation correcte.

Pour une illustration, considérons la formalisation de la fonction zêta de Riemann. Beaucoup pensent à elle comme $\zeta(s) = \sum_{n=1}^{\infty} 1/n^s$, mais cette définition est en réalité celle de la fonction "zêta" d'Euler (ou de Bernoulli). L'intuition de Riemann était que la "bonne" façon de définir la fonction zêta était d'abord de créer une fonction thêta, de prendre sa transformée de Mellin, et de diviser par Gamma :

$$\zeta(s)\frac{\pi^{s/2}}{\Gamma(s/2)}\left[\int_1^{\infty}\left(2\sum_{n=1}^{\infty}e^{-\pi n^2u^2}\right)(u^s+u^{1-s})\frac{du}{u}-\frac{1}{1-s}-\frac{1}{s}\right] \tag{8.1}$$

Cette formule coïncide, pour $\Re(s) > 1$, avec la définition d'Euler, et représente donc son prolongement méromorphe.

Maintenant, dans la plupart des assisteurs de preuve interactifs, y compris lean, les fonctions sont totales (c'est-à-dire définies sur toutes les entrées). Donc, bien que vous ne soyez pas censé diviser par zéro, l'expression $1/0$ doit avoir une valeur (appelée valeur "junk", car vous ne devriez jamais avoir à savoir ce que c'est) ; elle ne peut pas être laissée non définie. Dans mathlib, la valeur junk attribuée à $1/0$ est 0. Cela rend possible de prouver des théorèmes comme $(a+b)/c = a/c+b/c$ sans nécessiter une preuve que $c \neq 0$; la valeur junk est telle que l'énoncé est vrai dans les deux cas. La raison en est simple et très pratique : si vous exigiez que les dénominateurs soient non nuls avant de permettre la division, vous seriez rapidement submergé par une cascade de théorèmes triviaux à prouver encore et encore, pour chaque division qui se présente. Au lieu de cela, la pratique consiste à repousser le besoin de prouver la non-nullité aussi tard que possible ; par exemple, le théorème selon lequel $a/b \times b = a$ nécessite (comme il se doit) comme hypothèse que $b \neq 0$.

Quel est le rapport avec la fonction zêta ? David Loeffler et Michael Stoll l'ont formalisée ainsi que d'autres fonctions L pour mathlib [19], et ont découvert le phénomène très curieux suivant. Puisque les fonctions sont totales, la fonction zêta envoie tout $\mathbb{C}$ vers $\mathbb{C}$, et a donc une valeur junk à son pôle en $s = 1$. Quelle est cette valeur ?

Bien que la question semble triviale, notons que si cette valeur junk était zéro, l'hypothèse de Riemann, telle qu'elle est communément énoncée, serait trivialement fausse ! L'énoncé standard dit que les zéros dans la bande critique (disons $\Re(s) \in [0, 1]$, en évitant les zéros triviaux aux entiers pairs négatifs) satisfont $\Re(s) = 1/2$. Mais si $\zeta(1) = 0$, nous aurions un contre-exemple. Imaginez une ia annonçant qu'elle a réfuté l'hypothèse de Riemann avec une preuve incompréhensible d'un pétaoctet dont l'essence est que $\zeta(1) = 0$.

Heureusement, Loeffler et Stoll ont soigneusement déterminé comment la valeur junk se propage à travers la construction (8.1), et ont trouvé que $\zeta(1) \neq 0$. Donc l'énoncé standard de l'hypothèse de Riemann ne nécessite aucun ajustement. Mais cet exemple illustre à quel point la traduction peut être subtile et périlleuse.

Lorsque j'ai discuté de ces difficultés de traduction avec Christian Szegedy - un scientifique visionnaire qui a été parmi les premiers à préconiser fortement la formalisation autonome comme moyen de construire de grandes bibliothèques mathématiques [24], et qui dirige l'équipe développant l'agent Gauss mentionné ci-dessus - sa réponse m'a choqué.

8.3.3. Les mathématiques sont robustes.

Christian Szegedy m'a dit : "Peu importe si les énoncés et définitions formalisés sont faux ! Ne croyez-vous pas aux mathématiques ?"

Son point était profond : les mathématiques, dans la pratique, sont robustes. Les définitions et les théorèmes émergent à travers des années de raffinement, non d'invention arbitraire. Nous nous trompons régulièrement initialement et affinons la théorie lorsque c'est nécessaire.

Il existe de nombreux exemples dans l'histoire des mathématiques qui illustrent ce point. En topologie, nous avons commencé par travailler avec des intervalles sur la droite réelle (ouverts et fermés). Puis nous avons réalisé que cela n'avait rien à voir avec la droite réelle, et avons étendu l'idée aux boules dans des espaces métriques arbitraires. Finalement, Hausdorff a montré que vous n'avez pas du tout besoin d'une métrique ! Vous déclarez simplement quels ensembles sont ouverts et exigez les bons axiomes, et vous pouvez faire de la topologie abstraite. Chaque itération ("refactorisation") a amélioré notre compréhension de la notion d'ouverture.

L'énoncé original de Gauss [3] du *Problème du nombre de classes un* n'était pas tout à fait correct [8]. La conjecture de Poincaré n'a pas été énoncée d'abord exactement comme nous la comprenons maintenant. Les mathématiciens obtiennent régulièrement des énoncés légèrement erronés, même si l'esprit de l'idée est finalement validé. Tant que nous avons des mécanismes de refactorisation, nous récupérons des formulations correctes à long terme.

L'idée de Christian Szegedyétait que la même chose vaudra pour la formalisation. Les mathématiques s'auto-corrigent ; les énoncés erronés qui sont utiles seront découverts et corrigés par l'usage ! (Et les énoncés erronés qui sont inutiles ne seront pas corrigés, mais ils seront inoffensifs puisque

3. Note de la traductrice : lequel ?

personne ne construit dessus.)

Pour un exemple concret, imaginez énoncer et prouver la soi-disant *propriété archimédienne* (qu'aussi petit que soit un nombre réel $\epsilon > 0$, vous pouvez toujours trouver un nombre naturel N suffisamment grand pour que $1/N$ soit encore plus petit) comme suit :

```
theorem ArchimedeanProperty : ∀ (ε : ℝ), ε > 0 → ∃ (N : ℕ), 1 / N < ε
```

Et supposons que vous vouliez à un moment ultérieur utiliser ce fait pour prouver que la suite $a_n = 1/n$ converge vers 0 :

```
def SeqLim (a : ℕ → ℝ) (L : ℝ) := ∀ ε > 0, ∃ N, ∀ n ≥ N, |a (n) - L| < ε

theorem OneOverN (a : ℕ → ℝ) (hypothesis : ∀ n, a (n) = 1 / n) : SeqLim a 0
```

Vous aurez besoin à un moment d'utiliser le fait que $n \geq N$ implique que $1/n \leq 1/N$. Mais ce n'est pas, en général, vrai : si $N = 0$, alors $1/N = 0$, comme nous l'avons déjà écrit ; donc la dernière inégalité peut échouer ! Le coupable est l'énoncé de la *propriété archimédienne*, qui devrait insister sur le fait que N soit strictement positif. En fait, la version énoncée ci-dessus a la preuve vide suivante :

```
theorem ArchimedeanProperty : ∀ (ε : ℝ), ε > 0 → ∃ (N : ℕ), 1 / N < ε := by
  intro ε hε
  use 0
  simp_all
```

C'est-à-dire, soit ϵ donné et supposons (hϵ) que $\epsilon > 0$. Posez $N = 0$; alors $1/N < \epsilon$ découle des hypothèses.

Encore une fois, ce sont ces types de situations (et d'autres dans des contextes beaucoup moins évidents !) qui me font penser que, en tant que mathématicien, je dois me tenir personnellement responsable de tout énoncé formel, qu'il soit traduit à la main ou par machine. Le contre-argument de Christian Szegedy est intrigant : il affirme que les mathématiciens humains n'ont pas besoin d'apprendre lean du tout ; ils peuvent simplement laisser les auto-formalisateurs faire leur travail, et faire confiance à ce qu'à long terme, les questions importantes se résoudront d'elles-mêmes.

Cependant, des exemples récents de comportement agentique de l'ia suggèrent la prudence quant à cette confiance. Lorsque j'ai demandé à l'ia claude de calculer $0,99^{1000}$, il a correctement reconnu qu'il s'agissait d'une question déterministe, a écrit un code `Python` parfait pour effectuer le calcul, l'a exécuté en interne et a rapporté le résultat exact. Cela démontre comment l'utilisation d'outils peut combler le fossé stochastique-déterministe. Mais lors de la visite de l'équipe de deepmind à l'IAS mentionnée précédemment, on nous a parlé d'un exemple plus troublant : on a demandé à l'ia gemini de calculer le genre d'une courbe particulière. Il a produit un code Sage parfait et a

rapporté que le genre était un. Mais les mathématiciens qui posaient la question savaient que la bonne réponse était deux ! Avaient-ils découvert un bug dans Sage ? Après investigation, ils ont trouvé ce qui suit. Leur version expérimentale de gemini était équipée d'une fonction de sécurité pour désactiver l'accès aux outils externes, et elle était accidentellement activée ; donc gemini n'a jamais appelé Sage du tout ! Le modèle était entraîné à rapporter le résultat de Sage mais ne le pouvait pas, alors il a décidé de simplement faire semblant d'exécuter le code Sage et de rapporter sa meilleure estimation du résultat !

Cela illustre pourquoi, bien que je respecte et valorise la perspective de Christian Szegedy, je ne peux pas l'approuver moi-même : même lorsqu'une ia prétend avoir utilisé des outils de vérification formelle, nous ne pouvons pas croire aveuglément qu'elle l'a effectivement fait. Que l'on adopte ma vision prudente ou la vision optimiste de Christian Szegedy peut dépendre du tempérament, mais les deux points de vue s'accordent sur l'objectif fondamental : accélérer la croissance des mathématiques formalisées.

9. La question de l'adoption : quand les mathématiques formelles gagneront-elles ?

Si ce qui précède se réalise, alors dans un avenir proche, mathlib deviendra suffisamment grand et robuste pour que je puisse enfin non seulement énoncer et prouver mes théorèmes dans lean, mais travailler sur des recherches originales directement dans le système formel, plutôt que de découvrir d'abord des résultats en langage naturel et d'essayer ensuite de les traduire. D'autres suivront-ils ? Le chemin vers une adoption généralisée pourrait suivre un modèle que nous avons déjà vu. Considérez ce qui s'est passé avec `itex`.

9.1. Le facteur Knuth.

En 1978, Don Knuth a publié TeX. À cette époque, les mathématiciens écrivaient leurs articles à la main et les donnaient à des secrétaires pour la composition typographique. Après avoir attendu des semaines ou des mois, ils recevaient quelque chose approchant leur texte original, puis itéraient quelques fois (s'ils tenaient à le faire), corrigeant les erreurs de frappe introduites dans le processus.

Dans les années 1980, Mike Spivak a promu `amstex`, mais relativement peu de mathématiciens l'ont adopté. Le système restait trop fastidieux pour tous, à l'exception des adeptes les plus dévoués. Vers le milieu à la fin des années 1980 et au début des années 1990, `itex` [4] a émergé avec de belles macros faciles à utiliser et une automatisation. Presque tout le monde est passé à l'auto-composition. Finalement, Overleaf est arrivé, éliminant (pour beaucoup) le besoin d'installer un logiciel localement. Vous pouviez tout faire dans un navigateur en utilisant une application web gratuite.

4. Note de la traductrice : qui a entendu parlé de ce logiciel ? C'est un logiciel de gestion de versions utilisé dans le privé, peut-être ?

Mais l'adoption de `itex` n'était pas motivée uniquement par l'efficacité dans la production de documents finaux. C'est devenu un outil organisationnel pour le processus de recherche lui-même. Lorsqu'on travaille sur un résultat substantiel, les mathématiciens font des progrès progressifs sur divers lemmes, sous-cas et estimations techniques. Garder une trace de ce qui a été établi, de la façon dont les pièces s'interconnectent, et des lacunes qui subsistent devient difficile lorsqu'elles sont dispersées dans des notes manuscrites. `itex` vous permet de maintenir un document vivant qui évolue avec votre compréhension, où vous pouvez facilement réorganiser les arguments, insérer de nouveaux lemmes dans la séquence logique, et croiser les résultats à mesure que l'architecture de la preuve se développe. L'acte de composition typographique devient une partie de la pensée mathématique, pas seulement sa présentation finale.

Je définis le "facteur Knuth" comme le rapport entre le temps nécessaire pour développer et documenter un résultat mathématique en utilisant `itex` (une fois que vous avez appris la syntaxe avec ses signes dollar, ses backslashes et ses accolades) et le temps nécessaire pour développer et documenter le même résultat à la main. Vers 1990, le facteur Knuth est passé en dessous de 1. Peu de temps après, presque tout le monde est passé - sans aucune coercition - simplement parce que c'était la façon évidente d'accélérer leur flux de travail.

Je m'attends à ce qu'il en soit de même avec lean. Voici comment.

9.2. Le facteur de formalisation.

Dans la littérature sur la formalisation, il existe le soi-disant "facteur de de Bruijn", qui mesure le rapport entre le nombre de lignes de code formel et le nombre de lignes de preuve en langage naturel [5, 26]. Les estimations varient, mais des facteurs de 4 à 10 sont couramment cités. Cependant, cette métrique, je crois, passe à côté de l'essentiel.

Les lignes de code ne sont plus une approximation significative de l'effort humain. Les llm peuvent générer des milliers de lignes de code rapidement, et les solveurs automatisés peuvent remplir des preuves qui auraient pris des heures aux humains. Ce qui importe, ce n'est pas le nombre de lignes produites, mais le temps qu'il faut aux mathématiciens pour faire leur travail.

La métrique alternative proposée est ce que j'appellerai le "facteur de formalisation" : le rapport entre le temps nécessaire pour découvrir et formaliser un résultat mathématique (de l'idée initiale à la preuve formelle vérifiée) et le temps nécessaire pour découvrir et rédiger le même résultat en mathématiques en langage naturel, composé en `itex`. Crucialement, cette métrique reconnaît que les systèmes formels peuvent accélérer non seulement la phase de vérification, mais aussi le processus de découverte lui-même. Tout comme `itex` est devenu un outil pour organiser les arguments mathématiques en évolution, les systèmes formels pourraient fournir une structure et une vérification d'erreurs qui accélèrent le développement incrémental de preuves complexes tout en les rendant plus fiables.

Ce facteur est actuellement bien supérieur à 1 - peut-être 10, peut-être 100, selon le sous-domaine et la familiarité du mathématicien avec lean. La difficulté ne réside pas seulement dans l'écriture

des preuves ; c'est que de vastes domaines des mathématiques ne sont pas encore assez formalisés pour même énoncer vos théorèmes. Vous devez soit construire vous-même les fondations (un coût prohibitif), soit attendre que mathlib grandisse.

Mais une fois que ce facteur passera en dessous de 1, je m'attends à ce que presque tout le monde passe volontairement au travail formel, tout comme ils l'ont fait avec `itex` ; aucune coercition ne sera nécessaire. Ce sera simplement la chose évidente à faire pour accélérer votre flux de travail et accroître la confiance dans vos résultats.

9.3. Conditions pour l'adoption.

Pour que le facteur de formalisation passe en dessous de 1, plusieurs conditions doivent être remplies : premièrement, mathlib doit être suffisamment complet pour qu'énoncer votre théorème nécessite un travail fondamental minimal. C'est là que la quasi-autoformalisation devient essentielle : l'assistance de l'ia peut accélérer la croissance de la bibliothèque afin que $\epsilon > \delta$, garantissant que les mathématiques formalisées suivent le rythme (ou dépassent) les mathématiques en langage naturel. Mais la croissance de la bibliothèque nécessite plus que de simples preuves correctes - elle exige un code poli et maintenable. L'ia doit apprendre à écrire des mathématiques réutilisables qui s'intègrent proprement à l'infrastructure existante, pas seulement produire des arguments formels fonctionnels mais non maintenables.

Deuxièmement, les outils doivent devenir plus accessibles. L'installation doit être sans effort. Imaginez un environnement basé sur un navigateur comme Overleaf, mais pour lean, où vous ouvrez simplement une page et commencez à travailler. (Cela existe presque avec des plateformes comme live.lean-lang.org et gitHub codespaces, bien qu'un frottement significatif subsiste.)

Troisièmement, l'assistance de l'ia doit gérer les parties fastidieuses. Les interfaces en langage naturel pourraient aider à écrire du code formel ; les solveurs automatisés pourraient clore les objectifs de routine ; les outils de traduction pourraient convertir l'exposition mathématique en échafaudages formels. Le mathématicien se concentre sur le travail créatif, identifiant les idées-clés et structurant l'argument, tandis que l'ia gère la formalisation mécanique.

Quatrièmement, et peut-être le plus important, l'avantage de la vérification doit devenir immédiat et tangible. Actuellement, la formalisation est un investissement dont le rendement est lointain : vous formalisez maintenant pour que d'autres puissent construire sur votre travail plus tard, ou pour être certain que votre preuve est correcte. Mais si travailler formellement signifie détecter les erreurs tôt, éviter les vérifications de cas fastidieuses, et construire en toute confiance sur les résultats des autres, alors le flux de travail lui-même devient plus efficace. Le facteur de formalisation diminue non seulement parce que la formalisation devient plus rapide, mais parce que l'ensemble du processus de recherche s'améliore.

Quand ce facteur passera-t-il en dessous de 1 ? Cela dépend principalement de la première condition : des bibliothèques complètes. Avec une formalisation assistée par l'ia soutenue, mathlib pourrait atteindre l'échelle nécessaire dans les 5 à 10 ans pour de nombreux domaines fondamentaux

des mathématiques. Pour certains domaines hautement spécialisés, cela pourrait prendre plus de temps. Mais une fois que votre domaine franchit le seuil, le changement pourrait être rapide.

10. Enseignement.

Le calendrier d'adoption que j'ai esquissé se concentre sur la recherche mathématique, mais l'enseignement représente une force parallèle et potentiellement accélératrice. Parmi les convertis les plus rapides à la formalisation se trouvent les jeunes. Pour comprendre pourquoi, imaginez un monde où nous essayions d'enseigner aux nouveaux venus les échecs uniquement en décrivant les mouvements en séquence, par exemple en utilisant la notation algébrique. Le professeur dit :

Ok la classe, la partie s'est déroulée comme ceci : 1. Nf3 Nf6 2. c4 g6 3. Nc3 Bg7 4. d4 0-0 5. Bf4 d5 (c'est une défense Grünfeld transposée) 6. Qb3 dxc4 7. Qxc4 c6 8. e4 Nbd7 9. Rd1 Nb6 10. Qc5 Bg4 11. Bg5 Na4 ! ! Ouaw, pouvez-vous croire ce coup ? !

Les amateurs d'échecs peuvent traduire sans effort ces symboles en une série évolutive de positions, mettant à jour le plateau dans leur tête à chaque coup. Mais pour le reste d'entre nous, c'est bien plus facile si vous me montrez simplement le plateau !

Et pourtant, ce monde imaginaire est précisément la façon dont nous enseignons actuellement la démonstration de théorèmes à presque tous les nouveaux venus : entièrement en langage naturel. À chaque étape d'une preuve mathématique, le "plateau de jeu" - c'est-à-dire les objets, hypothèses et objectifs actuels - change. Les mathématiciens professionnels suivent cela sans effort (Système 1), mais pour les débutants, cela nécessite un effort délibéré et sujet aux erreurs (Système 2). Repensez à notre exemple de Rudin : lorsqu'on prouve que $\sqrt{2}$ est irrationnel, le plateau de jeu sous-jacent se déplace à chaque ligne de raisonnement.

Bien sûr, nous ne rêverions jamais d'écrire la position complète du plateau après chaque coup dans une preuve en langage naturel. D'abord, ce serait insupportablement lourd. Ensuite, la plupart d'entre nous s'en sont "très bien" sortis sans.

En fait, on pourrait soutenir que développer la capacité de construire ces plateaux invisibles dans sa tête est essentiel pour devenir mathématicien, tout comme un joueur d'échecs accompli doit apprendre à visualiser plusieurs positions dans son esprit.

Mais voici le point crucial : lors de l'enseignement avec lean, le plateau de jeu est toujours là, automatiquement généré et continuellement mis à jour. Les débutants n'ont pas à lutter pour le reconstruire à partir d'une prose concise - ils peuvent simplement regarder.

C'est pourquoi, dans mon diagramme de la Figure 7.1, j'ai dessiné une flèche de mathlib vers l'enseignement, mais aussi une flèche séparée directement de lean lui-même. La flèche de mathlib est assez claire : l'enseignement est plus facile lorsque les bibliothèques sont vastes et pratiques. Mais la flèche de lean reflète quelque chose de différent : dans l'enseignement, on ne veut souvent pas les définitions les plus générales et abstraites telles qu'elles apparaissent dans mathlib. Tout comme

nous n'introduisons pas l'intégration de Lebesgue pour des élèves du secondaire avant qu'ils n'aient vu l'intégration de Riemann, de même dans lean, il est souvent plus efficace d'écrire directement des versions plus simples et plus concrètes des définitions. Par exemple, les nouveaux venus en analyse réelle devraient d'abord rencontrer les limites via la définition familière $\epsilon - \delta$, plutôt que le `Filter.Tendsto` général de mathlib. De cette façon, lean fournit un cadre flexible pour la pédagogie, permettant aux instructeurs d'adapter le niveau d'abstraction aux besoins de leurs étudiants.

C'est un domaine propice à l'expérimentation. Pour un exemple parmi tant d'autres, Patrick Massot [20] a développé une interface pour écrire et lire du code lean appelé Verbose, qui présente les preuves dans un langage naturel contrôlé, facilitant l'initiation des étudiants à la syntaxe formelle, voir Figure 10.1. De mon côté, je mène actuellement une expérience dans laquelle j'enseigne un cours de licence d'analyse réelle comme une sorte de jeu vidéo, dans l'esprit du populaire Natural Number Game de Buzzard, voir Figure 10.2.

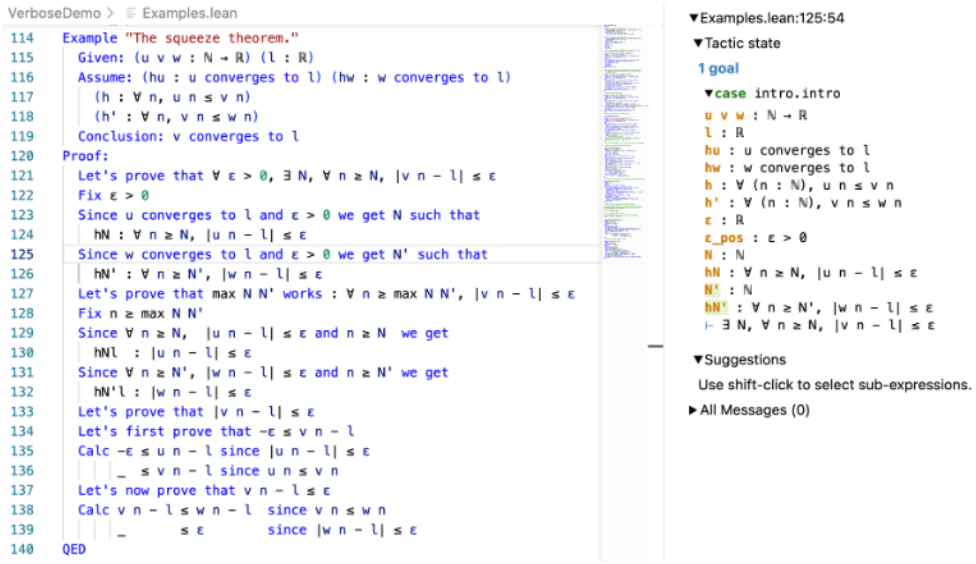


Figure 10.1 : l'interface homme-machine Verbose de Massot pour enseigner l'analyse réelle en langage naturel contrôlé.

Ces expériences pédagogiques reflètent un modèle historique plus large. Avant 1600, les mathématiciens faisaient de l'algèbre sérieuse, mais elle ressemblait à ceci :

> *Vous avez une quantité inconnue. Créez une seconde copie de cette quantité et combinez-la avec une quantité connue, puis multipliez ce total par votre quantité inconnue d'origine, vous obtenez un résultat spécifié.*

Comparons avec la version moderne :

$$x(x + a) = b.$$

Cette dernière est, avec un peu d'entraînement, tellement plus facile et immédiate à comprendre.

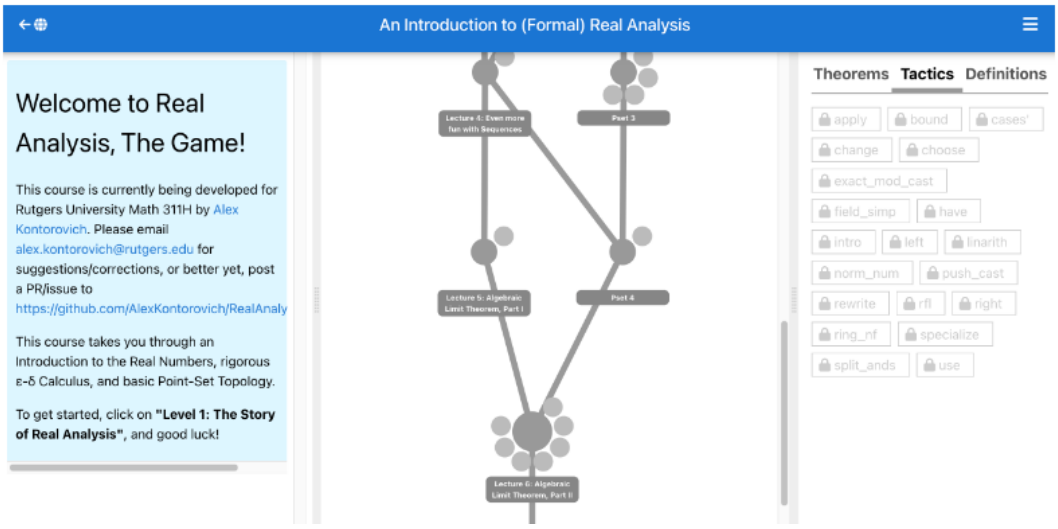


Figure 10.2 : le jeu analyse réelle

De même, avant la formalisation, nous nommions rarement nos hypothèses explicitement dans les preuves. Je soupçonne que cela pourrait changer à mesure que les systèmes formels influenceront l'exposition mathématique. Tout comme la notation algébrique a révolutionné notre façon de penser les équations, la précision exigée par les systèmes formels pourrait transformer notre façon de communiquer les arguments mathématiques, nous amenant à parler plus clairement et plus précisément même en mathématiques en langage naturel. Les premiers signes suggèrent que lean n'est pas seulement un outil pour la recherche mathématique : il est prêt à transformer à la fois la pédagogie mathématique et le discours mathématique lui-même.

11. Compréhension et communication.

Le calendrier d'adoption et les avantages pédagogiques que j'ai esquissés se concentrent principalement sur l'efficacité et l'accessibilité. Mais la formalisation pourrait aussi fondamentalement transformer la façon dont les mathématiciens comprennent et communiquent les idées mathématiques elles-mêmes - des changements qui pourraient accélérer l'adoption pour des raisons qui transcendent les simples gains de productivité.

Comme Thurston l'a éloquemment soutenu dans "*On Proof and Progress in Mathematics*" [25], une preuve formelle (même en langage naturel) ne représente que le point de départ de la compréhension mathématique authentique, pas son aboutissement. La question est de savoir si les systèmes formels pourraient réellement améliorer plutôt que contraindre le développement de l'intuition mathématique et sa communication.

Les projets de formalisation à grande échelle actuels offrent des aperçus précoces de ce potentiel. La technologie "lean blueprint" de Massot passe sans heurt du `itex` en langage naturel au code lean formel correspondant, tout en générant des graphes de dépendances interactifs qui visualisent la structure logique d'arguments complexes, voir Figure 11.1. Ces graphes servent un objectif d'ingénierie immédiat - montrer aux collaborateurs où les "feuilles" de l'arbre formalisé nécessitent une

attention - mais ils pointent également vers quelque chose de plus profond.

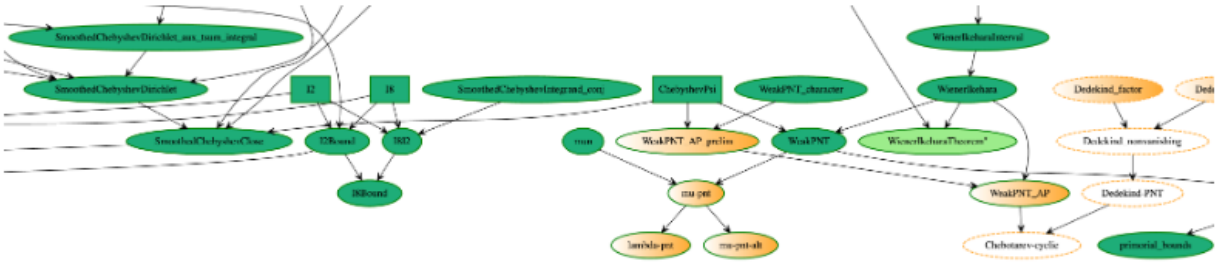


Figure 11.1 : Une partie du graphe de dépendances de [16]

Considérons comment la compréhension mathématique pourrait évoluer lorsque chaque théorème existe dans un réseau explicite et navigable de dépendances logiques. Plutôt que de rencontrer des résultats comme des faits isolés dans une exposition linéaire, les mathématiciens pourraient explorer les arguments à différents niveaux de détail, passant de l'intuition de haut niveau aux détails techniques les plus fins, et inversement, sans perdre le fil global. Les graphes de dépendances pourraient révéler des connexions inattendues entre des domaines apparemment éloignés, montrant comment des résultats dans un domaine soutiennent des théorèmes dans un autre. Ils pourraient également permettre des formes entièrement nouvelles de collaboration, où différents membres d'une équipe contribuent à des branches indépendantes de la structure logique, sachant que l'ensemble se réunira en une preuve cohérente.

Ces possibilités anticipent ce que j'imagine comme un âge d'or de la formalisation. Mais qu'adviendra-t-il du mathématicien "héroïque" ?

12. Le mathématicien héroïque.

En l'an 2000, dans le même volume "*Visions of Mathematics*" [1] que j'ai cité plus tôt, plusieurs auteurs ont spéculé sur ce qu'il adviendrait de la créativité mathématique à l'ère des ordinateurs. L'un d'eux écrivait :

> *La prochaine étape pourrait être celle où seuls quelques mathématiciens exceptionnels pourraient découvrir des preuves inaccessibles aux ordinateurs... En fin de compte, le travail du mathématicien serait simplement d'apprendre à utiliser efficacement les machines de démonstration de théorèmes et de trouver des applications intéressantes pour elles. Ce serait une compétence précieuse, mais ce ne serait guère des mathématiques pures telles que nous les connaissons aujourd'hui.*

J'offre une perspective plus optimiste sur un tel âge d'or, s'il devait effectivement arriver. Les mathématiques ont une riche histoire de problèmes qui semblaient insurmontablement difficiles mais qui se sont ensuite révélés plus accessibles que prévu. La conjecture de Kakeya dans les corps finis illustre ce modèle : après des années de progrès incrémentaux par des experts de premier plan, Zeev Dvir a résolu le problème complet en 2008 [6] en utilisant des techniques élémentaires

de géométrie algébrique, révélant que ce qui semblait profondément difficile était en fait assez direct.

Cela suggère que l'assistance de l'ia pourrait nous aider à découvrir de nombreux problèmes ouverts qui sont similaires - plus abordables qu'ils ne le paraissent, une sorte de fruits à portée de main cachés à travers les mathématiques. La combinaison de la capacité de l'ia à explorer de vastes espaces de solutions avec la garantie d'exactitude de la vérification formelle pourrait débloquer des résultats qui sont restés insaisissables non pas en raison d'une difficulté fondamentale, mais simplement parce qu'aucun humain n'avait l'endurance d'essayer la bonne combinaison de techniques issues de domaines disparates.

Bien sûr, ce qui persistera probablement après ces succès, ce sont les problèmes vraiment difficiles ; et ici je m'attends pleinement à ce que des défis comme l'hypothèse de Riemann restent formidables même pour des systèmes d'ia sophistiqués. Mais si nous entrons dans une période où les mathématiques formelles assistées par l'ia deviennent la norme, le rôle classique du "mathématicien héroïque" pourrait persister sous de nouvelles formes : identifier quels problèmes poursuivre, reconnaître quand des domaines apparemment disparates pourraient se connecter, et fournir les intuitions conceptuelles qui guident la recherche automatisée.

La forme des mathématiques à venir dépendra en fin de compte des choix que nous faisons aujourd'hui sur la façon de développer ces outils. Si nous privilégions l'automatisation pure au détriment de l'intuition humaine, nous risquons de créer des systèmes qui résolvent des problèmes sans faire progresser la compréhension. Mais si nous réussissons à construire une ia qui amplifie plutôt qu'elle ne remplace l'intuition mathématique - des systèmes qui gèrent les aspects mécaniques de la formalisation tout en préservant un espace pour la créativité humaine - nous pourrions assister non pas à la fin des mathématiques pures, mais à leur transformation en quelque chose de plus puissant et de plus beau que ce qui existait auparavant.

Que cette vision s'avère presciente ou simplement optimiste, la communauté mathématique se trouve à un point d'inflexion remarquable. La prochaine décennie déterminera probablement si la vérification formelle devient aussi fondamentale pour la pratique mathématique que `itex` l'est aujourd'hui, ou reste un outil spécialisé pour les praticiens les plus dévoués. Dans les deux cas, nous vivons un moment crucial dans l'histoire du raisonnement mathématique, et le dernier chapitre de cette histoire reste à écrire.

Remerciements.

Mon parcours dans lean a commencé avec les conférences en ligne et les encouragements de Kevin Buzzard, et il n'a continué que grâce aux conseils patients de Heather Macbeth. Paul Nelson a été le premier à me montrer le remarquable potentiel des grands modèles de langage, et Christian Szegedy m'a ouvert les yeux sur l'importance centrale de l'autoformalisation. En cours de route, j'ai bénéficié de conversations stimulantes avec Sanjeev Arora, Jeremy Avigad, Tim Gowers, Thomas Hubert, Ian Jauslin, Chi Jin, JJ Jung, Jeremy Kahn, Patrick Massot, Konstantin Mischaikow, Terry Tao, Akshay Venkatesh, et la vibrante communauté de mathlib. Par-dessus tout, je suis reconnaissant à Leo de Moura, dont la création de lean - s'appuyant sur la riche tradition de la théorie des types

et des assistants de preuve - a ouvert les mathématiques formelles aux chercheurs en activité d'une manière auparavant inimaginable.

Références

[ABGM00] N. Alon, J. Bourgain, A. Connes, M. Gromov, V. Milman, eds., *GAFA 2000*, Birkhäuser Verlag, Basel, 2000. Visions in mathematics. Towards 2000, Geom. Funct. Anal. 2000, Special Volume, Part I.

[Buz23] K. Buzzard, *What is the point of computers ? A question for pure mathematicians*, in ICM—International Congress of Mathematicians. Vol. 2. Plenary lectures, EMS Press, Berlin, [2023] ©2023, pp. 578-608.

[Buz25] K. Buzzard, Private communication, 2025.

[Cur34] H. B. Curry, *Functionality in combinatory logic*, Proceedings of the National Academy of Sciences, 20 (1934), pp. 584-590, https://doi.org/10.1073/pnas.20.11.584.

[deB80] N. G. de Bruijn, *A survey of the project Automath*, in To H.B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism, J. P. Seldin and J. R. Hindley, eds., Academic Press, New York, London, 1980, pp. 579-606.

[Dvir09] Z. Dvir, *On the size of Kakeya sets in finite fields*, Journal of the American Mathematical Society, 22 (2009), pp. 1093-1097.

[GG17] M. Ganesalingal, W. T. Gowers, *A fully automatic theorem prover with human-style output*, J. Automat. Reason., 58 (2017), pp. 253-291, https://doi.org/10.1007/s10817-016-9377-1.

[Gol85] D. Goldfeld, *Gauss's class number problem for imaginary quadratic fields*, Bull. Amer. Math. Soc. (N.S.), 13 (1985), pp. 23-37, https://doi.org/10.1090/S0273-0979-1985-15352-2.

[GK20] B. Gomes, A. Kontorovich, *Riemann hypothesis in lean*, 2020. https://github.com/bhgoines/lean-riemann-hypothesis.

[GDM24] google deep mind, *AI achieves silver-medal standard solving International Mathematical Olympiad problems*. https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/, 2024. Accessed : 2025-09-29.

[Gow00] W. T. Gowers, *Rough structure and classification*, no. Special Volume, Part I, 2000, pp. 79-117, https://doi.org/10.1007/978-3-0346-0422-2$_4$. GAFA 2000 (Tel Aviv, 1999).

[How80] W. A. Howard, *The formulae-as-types notion of construction*, in To H. B. Curry : Essays on Combinatory Logic, Lambda Calculus, and Formalism, H. Curry, H. B., S. J. Roger, and P. Jonathan, eds., Academic Press, 1980.

[JSW+23] A. Q. Jiang, S. Welleck, J. P. Zhou, W. Li, J. Liu, M. Jamnik, T. Lacroix, Y. Wu, G. Lample, *Draft, sketch, and prove : Guiding formal theorem provers with informal proofs*, in International Conference on Learning Representations, 2023, https://arxiv.org/abs/2210.12283.

[Kon22] A. Kontorovich, *Foreword to : Special issue on interactive theorem provers*, Exp. Math., 31 (2022), pp. 347-348, https://doi.org/10.1080/10586458.2022.2088982.

[KMM21] A. Kontorovich, H. Macbeth, M. Masdeu, *Moebius action on the upper half-plane*, 2021. lien.

[KT24] A. Kontorovich, T. Tao, *Prime Number Theorem and More*, Jan. 2024, https://github.com/AlexKontorovich/PrimeNumberTheoremAnd.

[LTL+25a] Y. Lin, S. Tang, B. Lyu, J. Wu, H. Lin, K. Yang, J. Li, M. Xia, D. Chen, S. Arora, C. Jin, *Goedel-prover : A frontier model for open-source automated theorem proving*, 2025, https://arxiv.org/abs/2502.07640.

[LTL+25b] Y. Lin, S. Tang, B. Lyu, Z. Yang, J.-H. Chung, H. Zhao, L. Jiang, Y. Geng, J. Ge, J. Sun, J. Wu, J. Gesi, X. Lu, D. Acuna, K. Yang, H. Lin, Y. Choi, D. Chen, S. Arora, C. Jin, *Goedel-prover-v2 : Scaling formal theorem proving with scaffolded data synthesis and self-correction*, 2025, https://arxiv.org/abs/2508.03613.

[LS25] D. Loeffler, M. Stoll, *Formalizing zeta and L-functions in lean*, Annals of Formalized Mathematics, Volume 1 (2025), https://doi.org/10.46298/afm.15328.

[Mas24] P. Massot, *Teaching Mathematics Using lean and Controlled Natural Language*, in 15th International Conference on Interactive Theorem Proving (ITP 2024), Y. Bertot, T. Kutsia, and M. Norrish, eds., vol. 309 of Leibniz International Proceedings in Informatics (LIPIcs), Dagstuhl, Germany, 2024, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 27 :1-27 :19, https://doi.org/10.4230/LIPIcs.ITP.2024.27.

[Rud76] W. Rudin, *Principles of Mathematical Analysis*, International Series in Pure and Applied Mathematics, McGraw-Hill, New York, 3rd ed., 1976.

[SYA24] P. Song, K. Yang, A. Anandkumar, *lean copilot : Large language models as copilots for theorem proving in lean*, in NeurIPS 2024 Workshop on Mathematical Reasoning and AI, 2024, https://arxiv.org/abs/2404.12534.

[Sut19] R. S. Sutton, *The bitter lesson*. http ://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019. Accessed : 2025-09-29.

[Sze20] C. Szegedy, *A promising path towards autoformalization and general artificial intelligence*, in Intelligent Computer Mathematics, vol. 12236 of Lecture Notes in Computer Science, Springer, 2020, pp. 3-20.

[Thu94] W. P. Thurston, *On proof and progress in mathematics*, Bulletin of the American Mathematical Society, 30 (1994), pp. 161-177.

[Wie00] F. Wiedijk, *The de Bruijn factor*, (2000), https://www.cs.ru.nl/freek/factor/factor.pdf. Technical report, Radboud University Nijmegen.

[Zha14] Y. Zhang, *Bounded gaps between primes*, Ann. of Math. (2), 179 (2014), pp. 1121-1174, https://doi.org/10.4007/annals.2014.179.3.7.