Les LLMs réalisent-ils un *calculus ratiocinator* ?

Autour de *Claude's Cycles* et de la *Septimana Mirabilis*

Luc Pommeret

LISN (CNRS / Univ. Paris-Saclay)

M2 Lophisc, Univ. Paris 1 Panthéon-Sorbonne

15 avril 2026

Atelier de lecture philosophie des mathématiques, SPHERE

« Pour résoudre une question ou terminer une controverse, les adversaires n'auront qu'à prendre la plume, en s'adjoignant au besoin un ami comme arbitre, et à dire : « Calculons ! » »

— *Couturat, La Logique de Leibniz (1901), paraphrasant Leibniz*

Plan

I. Panorama des outils LLM

Le *calculus ratiocinator* de Leibniz

Leibniz distingue deux composantes d'un projet de mécanisation du raisonnement :

- *Characteristica universalis* : un langage formel universel encodant les concepts.
- *Calculus ratiocinator* : un calcul opérant sur ce langage, réduisant tout raisonnement à de la manipulation symbolique.

Héritiers naturels : la logique formelle (Frege, Russell), les prouveurs automatiques, l'IA symbolique classique.

Paradoxe : les LLMs ne procèdent **pas** par réécriture symbolique, mais par statistique sur des représentations latentes. Et pourtant, ils semblent "calculer". Réalisent-ils le *calculus ratiocinator* par des moyens que Leibniz n'avait pas envisagés ?

LLM et Transformers

Transformer (Vaswani et al., 2017) : architecture neuronale basée sur le mécanisme d'*attention*.

Principe : prédire le token suivant x_{t+1} étant donné le contexte $x_1, \ldots, x_t$. Prédiction d'une distribution de probabilité sur le vocabulaire, puis tirage.

LLM (Large Language Model) :

- Transformer à grande échelle (milliards de paramètres)
- Pré-entraîné sur des corpus massifs (web, livres, code)
- Capacités émergentes : raisonnement, few-shot learning, génération de code

Exemples :

$$\text{GPT}_n \ (n \in \{3, 3.5, 4, 4.5, 5\}), \quad \text{Claude}_n \ (n \in \{3, 3.5, 3.7, 4, 4.5\}), \quad \text{Gemini}_n \ (n = 1 + \tfrac{k}{2})$$

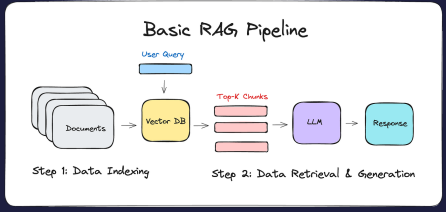
RAG (Retrieval-Augmented Generation)

Définition : architecture combinant un LLM et une base de connaissances externe.

Processus :

1. *Indexation* : le corpus est projeté dans un espace vectoriel
2. *Récupération* : recherche des documents pertinents par similarité
3. *Génération* : le modèle répond en s'appuyant sur le contexte récupéré

Objectif : réduire les hallucinations, étendre le contexte.



Agents, MCP et Vibe Coding

Agents (Wang et al., 2024) : le LLM comme contrôleur cognitif autonome.

- Planification (Chain of Thought, Reasoning)
- Mémoire : court terme (attention) + long terme (RAG)
- Outils via *MCP* (Model Context Protocol, Anthropic 2024)

Vibe Coding (Karpathy, fév. 2025) :

- Génération de code par dialogue avec un LLM
- Risque d'hallucination $\rightarrow$ besoin de vérification
- Brute force : le *Ralph Loop* (Huntley, 2025)

Synthèse : le LLM génère des esquisses, l'assistant de preuve (Lean, Rocq) vérifie.

II. *Claude's Cycles* (Knuth, 2026)

Claude's Cycles : le problème

Contexte : Knuth travaille sur les cycles hamiltoniens dirigés pour un futur volume de *The Art of Computer Programming*.

Le problème : soit le digraphe à m^3 sommets ijk ($0 \leq i, j, k < m$), avec trois arcs depuis chaque sommet (incrémentation de i, j ou k mod m). Décomposer l'ensemble des arcs en **trois cycles hamiltoniens dirigés**, pour tout $m > 2$.

État initial :

- Knuth avait résolu le cas $m = 3$.
- Filip Stappers avait trouvé empiriquement des solutions pour $4 \leq m \leq 16$.
- Conjecture : la décomposition existe pour tout $m > 2$.

Claude's Cycles : 31 explorations

Stappers soumet le problème à **Claude Opus 4.6**. En $\sim$**1 heure**, 31 explorations :

Reformulation Claude identifie un graphe de Cayley.

Expl. 1–2 σ cyclique avec g linéaire ou quadratique : échec. DFS brute-force : 6^{27}, trop lent.

Expl. 3 **Premier éclair** : redécouvre le *serpentin 2D*.

Expl. 4 Généralisation 3D via le *code de Gray m-aire* (Knuth, *TAOCP* 4A, p. 299) : un cycle, mais digraphe résiduel trop rigide.

Expl. 15 **Le déclic** : décomposition en *fibres* via $\pi(i,j,k) = i+j+k \bmod m$. Le problème 3D se ramène à un empilement de problèmes 2D.

Expl. 25 Auto-diagnostic : "*SA can find solutions but cannot give a general construction. Need pure math.*"

Expl. 31 **Succès** : dépendance en une seule coordonnée par fibre. Valide pour tout m **impair**, testé jusqu'à $m = 101$.

Rôle humain : Stappers relance, documente, guide.
"*After EVERY exploreXX.py run, IMMEDIATELY update plan.md before doing anything else.*"

Formalisation :

- Knuth rédige une preuve rigoureuse de la construction de Claude.
- **Kim Morrison** (communauté Lean) formalise cette preuve en Lean en 24h.

Le cas pair (m pair, $m \geq 8$) :

- Ho Boon Suan obtient une solution via **GPT-5.3 Codex**.
- La preuve de correction est produite par **GPT-5.4 Pro** : 14 pages, sans édition humaine.

Multi-agent : Exocija trouve une construction plus simple en alternant **GPT 5.4** et **Claude 4.6 Sonnet**.

Knuth : "*Hats off to Claude ! [...] We are living in very interesting times indeed.*"

III. IA et formalisation/découverte mathématique

Critères de la découverte (Tao, 2025)

Une contribution de l'IA est une découverte si :

1. La preuve est *nouvelle* (non présente dans la littérature mathématique)
2. Elle est *générée de manière autonome* (sans intervention humaine dans la fabrication)
3. Elle est *formalisée* et vérifiée (ici en Lean)

Dean et Naibo (2025)

"Artificial Intelligence and Inherent Mathematical Difficulty"
Philosophia Mathematica, juillet 2025.

Thèse (N0) : "Advances in computing theory or technology – inclusive of those currently understood to exemplify artificial intelligence – will not radically alter the following aspect of our current understanding of mathematics and its practice : Settling the status of open questions is an inherently difficult problem."

Thèse (N1) : "The application of artificial intelligence will not lead to a substantial reduction in the difficulty of one of the central goals of mathematical practice."

Arguments :

- La difficulté est structurelle (indécidabilité, NP-complétude)
- L'IA excelle dans la recherche (Σ_1^0), pas dans la conceptualisation

Complexité de découverte vs complexité de vérification

Complexité de vérification :

- Vérifier qu'une suite de symboles est une preuve valide
- Souvent polynomiale (P)

Complexité de découverte (Detlefsen, 1990) :

- Trouver le chemin vers la preuve
- Souvent exponentielle ou pire

La hiérarchie arithmétique reste la contrainte de l'IA (Dean et Naibo, 2025)

$\Sigma_1^0 : \exists x\, \phi(x)$

- Recherche existentielle
- Accessible à l'IA (recherche de témoin)

$\Pi_1^0 : \forall x\, \phi(x)$

- Propriété universelle
- Réfutation = trouver un contre-exemple (Σ_1^0)

$\Pi_2^0 : \forall x\, \exists y\, \phi(x, y)$

- Alternance de quantificateurs
- Nécessite un argument générique

La *Septimana Mirabilis* (4–12 janvier 2026)

4 janvier Kevin Barreto et “AcerFur” soumettent #728 à GPT-5.2 Pro.

4–6 janvier Aristotle traduit en Lean. Correction autonome.

8 janvier Terence Tao valide. Statut : **PROVED (LEAN)**.

10 janvier Résolution autonome du problème **#205** par Barreto avec GPT-5.2 Thinking et Aristotle.

11–12 janv. Réfutation du #397 et résolution du #729.

Le coup décisif : le Problème #205

Conjecture originale (Erdős) : tout n suffisamment grand peut-il s'écrire $2^k + m$ avec $\Omega(m) < \log\log m$?

Résultat IA (10 jan 2026) : Réfutation quantitative. Il existe une infinité de n tels que pour tout k avec $2^k < n$, $\Omega(n - 2^k) \gg \sqrt{\frac{\log n}{\log\log n}}$.

Méthode :

- Théorème des restes chinois (CRT) pour construire n.
- Utilisation du Théorème des Nombres Premiers (PNT).
- Argument par cas universel.

`https://www.erdosproblems.com/205`

Analyse logique du #205

L'énoncé formel exact, validé en Lean (`infinitely_many_counterexamples`) :

$$\exists c > 0, \forall N, \exists n > N, \forall k, (2^k \leq n \to \Omega(n - 2^k) \geq c \cdot \text{pntRate}(n))$$

Classification : Π_3^0 (ou Σ_4^0 avec le paramètre réel c).

- L'alternance interne $\exists n \forall k$ est irréductible.
- La preuve ne procède **pas** par force brute ou énumération, mais par construction analytique.
- Réfute la thèse selon laquelle seuls des problèmes Σ_1^0 (force brute déguisée) seraient accessibles aux LLMs (et donc aux IAs).

IV. Un cas limite : de la traduction à la compilation

Dépasser la notion de compilation ?

Projet expérimental : Traducteur de pseudo-code vers Python

- Finetuning d'un petit LLM (Qwen3.5 0.8B).
- Mode **déterministe** : exécution sur CPU avec température 0.
- Tâche : convertir du pseudo-code en anglais naturel vers du code Python exécutable.

Question : est-ce un compilateur ?

- La « formalisation » est régie par des probabilités linguistiques plutôt que des règles syntaxiques strictes.
- Cas limite : reconsidérer l'opposition entre compilation déterministe et traduction automatique.

`https://gitlab.lisn.upsaclay.fr/pommeret/pseudo-code`

Espace latent vs formalisation sémantique

Correspondance de Curry-Howard : dans un assistant de preuve (Lean, Rocq), un programme **est** une preuve. La formalisation est *sémantique* — chaque terme a une signification logique vérifiable.

Le LLM à température 0 :

- Le pseudo-code est projeté dans un **espace latent** (représentation vectorielle continue).
- La « compréhension » est *géométrique*, pas logique : proximité dans l'espace $\neq$ équivalence formelle.
- Pourtant : le code produit est correct et déterministe.

Tension : le LLM produit une formalisation *de facto* (du code exécutable) sans formalisation *de jure* (pas de correspondance preuves-programmes). Une sémantique latente plutôt que logique.

Démonstration

lancer la commande "pc exemple.pc"

Le compilateur interprète et traduit l'intention structurelle.

Conclusion

Méta-induction pessimiste et IA

L'argument classique (Stanford, 1a inst., Laudan 1981) :

> *Nos meilleures théories scientifiques passées se sont toutes avérées fausses. Donc nos théories actuelles le seront probablement aussi.*

Version appliquée à l'IA :

> *L'IA n'a jamais pu accomplir des tâches de complexité X. Donc elle ne le pourra probablement jamais.*

Asymétrie : chaque franchissement de seuil est **irréversible**.

- On ne peut pas "désapprendre" que le #205 (Π_3^0) a été résolu par un LLM.
- On ne peut pas ignorer que Claude a résolu un problème ouvert de TAOCP en 1h.
- La méta-induction pessimiste appliquée à l'IA se *réfute elle-même* à chaque nouveau seuil franchi.

Vers une hybridation ?

Bilan :

- Résolution autonome d'énoncés hautement complexes (Π_3^0 pour le #205).
- Collaboration itérative humain-machine (*Claude's Cycles* : 31 explorations).
- Continuité entre sémantique latente (compilateur de pseudocode) et preuve formelle (Lean).

Quelles perspectives pour le chercheur ?

- **Le rôle humain** : formulation de problèmes, architecture, jugement esthétique.
- **Le LLM comme *exosomatisation* mathématique** (Stiegler) : délégation partielle de la charge combinatoire.
- **Explication** : comment expliquer une preuve trouvée et certifiée de manière purement computationnelle ?

Références

- Couturat, L. (1901). *La Logique de Leibniz d'après des documents inédits*. Paris : Alcan.
- van Heijenoort, J. (1967). "Logic as Calculus and Logic as Language." *Synthese* 17, 324–330.
- Knuth, D. E. (2026). *Claude's Cycles*. Stanford CS.
- Dean, W. & Naibo, A. (2025). *Artificial Intelligence and Inherent Mathematical Difficulty*. Philosophia Mathematica.
- Henkel, J. (2025). *The Mathematician's Assistant*. Math Semesterber 72, 117–144.
- Pommeret, L. (2026). Dépôt *pseudo-code* : `https://gitlab.lisn.upsaclay.fr/pommeret/pseudo-code`