How LLMs got better at reasoning

A few advances from 1906 to 2021, and then from 2022 to 2026 (Part 2)

Pierre Vandenhove

AI in Math & CS Research — UMONS
26 October 2026

UMONS
Université de Mons

Overview

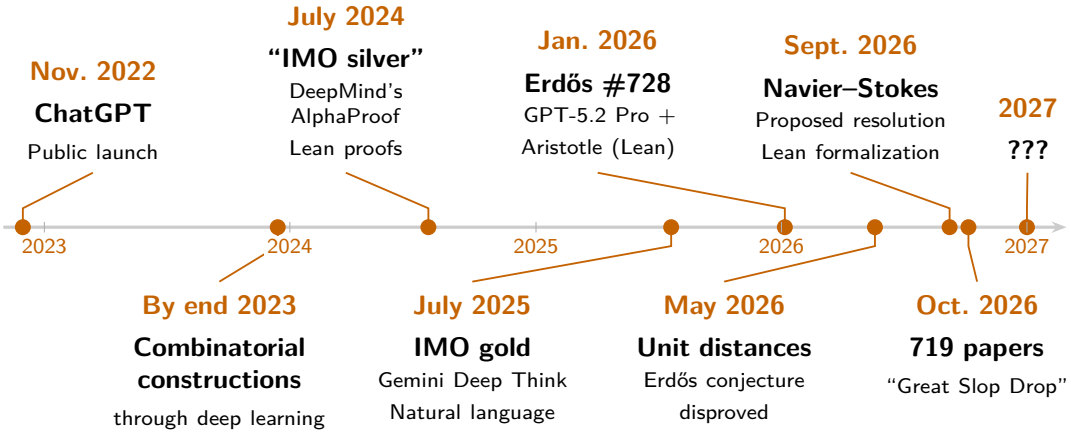
Objective

Present ideas that have contributed to improvements in LLM **reasoning**.

Three limitations of this talk:

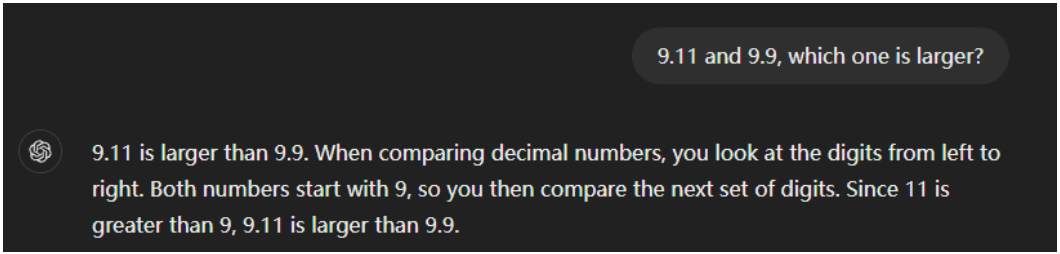
- We don't know everything: the **training recipes** of reasoning models have largely remained **secret** for competitive reasons.
- This talk is biased towards **maths** ("proof-based") research.
  But many techniques also apply to **code**.
- I don't do research on LLMs, but I've tried to follow recent developments.

Some mathematical milestones



OpenAI, Introducing ChatGPT (2022); Wagner 2021; Romera-Paredes et al. 2023; DeepMind, IMO silver (2024); Hubert et al. 2025 (*Nature*); DeepMind, IMO gold (2025); Sothanaphan 2026; Alon et al. 2026; OpenAI, Navier–Stokes (2026); Clay, Navier–Stokes statement (2026); OpenAI, Mathematical results (Oct. 2026).

Where we come from



GPT-4o, reported in August 2024. Same answer with Llama3. Two years ago!

September 2024: o1, an "extended reasoning model", is launched.

What changed in how we train models and how we use them?

Xie 2024, *Order Matters in Hallucination: Reasoning Order as Benchmark and Reflexive Prompting for Large-Language-Models*, Section 3
OpenAI, o1 launch (Sept. 2024).

Two stages: training and inference

During training

(Updating model/verifier weights)

- Process supervision (2023)
  Learn to score reasoning steps.
- RL with verifiable rewards (2024–25)
  Learn from automatically checked attempts.

During inference

(Use the trained model to solve a problem, with fixed weights)

- Chain-of-thought prompting (2022)
  (*Think step by step*, cf. previous talk)
- Self-consistency (2022)
- Tool use (2022)
- Search and test-time scaling (2024)
- Formal checking (AlphaProof, 2024)

Self-consistency

Self-consistency[1] can improve performance at inference time.

Simple idea: for a given question,

- generate **several solutions** to the **same** question, using random sampling;
- return the most frequent **final answer**.

Intuition for why this works well (at the cost of increased computation):

- Different **correct reasoning paths** can converge on the **same correct answer**.
- **Wrong paths** often lead to **distinct wrong answers**.

[1] Wang et al. 2022, *Self-Consistency Improves Chain of Thought Reasoning in Language Models*.

2022 Tool use

LLMs can recognise **when they need a tool** and choose to use it.[2]

- **Write** a program → **run** it → **observe** the output → **repeat**.
- Execution avoids many arithmetic errors.

Demo: without and with Python **[Example run at this link]**

```
Let f(n) = n^2 - 41n + 461 for nonnegative integers n.
Determine the smallest n such that f(n) is not prime.
```

Let's test it with Gemini 3.1 Flash Lite (May 2026, fast but not the smartest).

[2] Yao et al. 2022, *ReAct: Synergizing Reasoning and Acting in Language Models.*
Gao et al. 2022, *PAL: Program-aided Language Models.*

Process supervision (1/2)

Train a separate model to **score reasoning steps**:
"*Let's Verify Step by Step*": *process supervision*[3]

- A wrong final answer tells us little about **why** it failed.
- **Labelling individual steps** helps locate the first mistake.

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $2/5$, what is the numerator of the fraction? (Answer: $\boxed{14}$)

⋮

☹ 😐 😎 5x = 6x - 14.

☹ 😐 😎 So x = 7.

Here, training this verifier requires **human labels for individual reasoning steps**.

[3] Lightman et al. 2023, *Let's Verify Step by Step*, Sections 2–3.

2023 Process supervision (2/2): Selecting among attempts

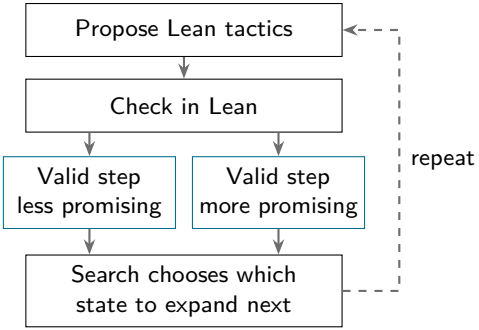
Once the verifier is trained, use it to choose a solution.
The generator model is not updated.

1. Generate several candidate solutions.
2. Score their steps with the verifier.
3. Combine the step scores and return the highest-scoring candidate.

$\rightsquigarrow$ More fine-grained than self-consistency: not just aggregating final answers, but evaluating the reasoning process with a learned verifier.

Formal checking during search

DeepMind's AlphaProof searches for formal proofs in **Lean**.



- Lean checks whether proposed formal steps are valid.
- Several valid continuations may exist.
- Search continues from the most promising proof states.

Solved 4/6 problems at IMO'24; required manual formalisation of statements and three days of compute.

DeepMind, IMO silver (2024).
Hubert et al. 2025, *Olympiad-level formal mathematical reasoning with reinforcement learning* (*Nature*).

Recap: Scaling test-time compute

We've seen several ways to spend more computation at inference time: chain of thought, self-consistency, tool use, learned or automated selection...

Research question: How does performance improve as we spend more computation on a problem? **How** should we spend it?[4]

[4] Snell et al. 2024, *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*.

RL with verifiable rewards

Instead of just selecting good solutions, can we use them to improve the model?

$$\text{Generate attempts} \longrightarrow \text{Check answers} \longrightarrow \text{Update model}$$

For some tasks, the solution can be **checked automatically**.

- **Mathematics:** does the final answer match the known answer? Does the Lean proof compile?
- **Programming:** does the code pass the tests?

$$\rightsquigarrow \text{Reinforcement learning with verifiable rewards (RLVR)}.^{5}$$

No need for a human to evaluate every attempt or step.

[5] DeepSeek-AI 2025, *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*.

DeepSeek-R1-Zero

Pretrained LLM $\longrightarrow$ RLVR
without supervised reasoning examples[6]

Two motivations:

- Human-written reasoning traces are expensive to collect at scale.
- Imitating human demonstrations may limit the exploration of other reasoning strategies.

The model generates its own reasoning attempts. The reward checks the final answer (and the output format), *not the reasoning*.

Not starting from zero! The base model was pretrained on human-produced data.

[6] Guo et al. 2025, *DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning*.

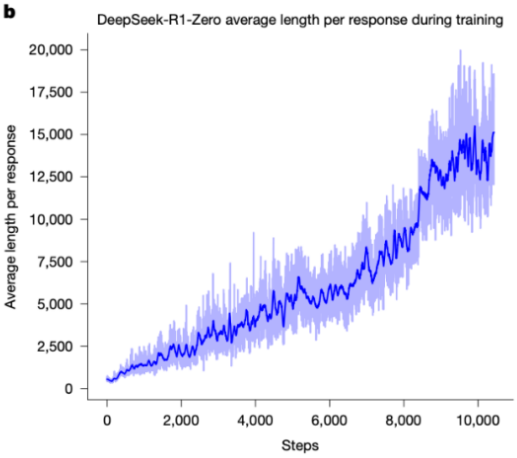
What **emerged** during R1-Zero training?

During RL, the model starts to:[7]

- Generate longer reasoning sequences.
- Revisit earlier steps.
- Explore alternative approaches.

"Wait, wait. Wait... Let's reevaluate this step-by-step"

But R1-Zero has issues: poor readability (e.g., repetitions, mixing languages).
$\rightsquigarrow$ DeepSeek-R1 then combines supervised fine-tuning and RL.



AIME 2024: 15.6% $\longrightarrow$ 71.0% after RL training.
Then 86.7% with self-consistency over 64 attempts.

[7] Guo et al. 2025, *DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning*, Fig. 1b.

2019 Sutton's *Bitter Lesson*

"The two methods that seem to scale arbitrarily in this way are search and learning."

Richard Sutton, *The Bitter Lesson* (2019)[8]

The idea: general methods exploiting more computation have repeatedly outperformed methods based on explicitly encoding human expertise.

$\rightsquigarrow$ Rather than prescribing how to reason, let the model discover strategies through feedback.

[8] Sutton 2019, *The Bitter Lesson*. `https://bitterlesson.ai/`.

Conclusion: What's next?

No clear plateau yet. Progress comes from:

- Better hardware and software (*increasingly guided by AI itself*), possibly new architectures (e.g., *looped transformers*).
- Training: more compute and data, including self-generated problems.
- Inference: more search, verification, increased context size, parallel agents.
  - Massive multi-agent systems: ∼10,000 concurrent agents in OpenAI's Navier–Stokes effort (∼88 hours).[9]

$\rightsquigarrow$ Improving both **models** and **the systems around them**.

Thank you for your attention!

[9] OpenAI 2026, *On the Navier–Stokes Millennium Prize Problem*.

Sources for the methods

Gao, L. et al. (2022). *PAL: Program-aided Language Models*. arXiv: 2211.10435 [cs.CL].

Wang, X. et al. (2022). *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. arXiv: 2203.11171 [cs.CL].

Yao, S. et al. (2022). *ReAct: Synergizing Reasoning and Acting in Language Models*. arXiv: 2210.03629 [cs.CL].

Lightman, H. et al. (2023). *Let's Verify Step by Step*. arXiv: 2305.20050 [cs.LG].

Snell, C. et al. (2024). *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*. arXiv: 2408.03314 [cs.LG].

Xie, Z. (2024). *Order Matters in Hallucination: Reasoning Order as Benchmark and Reflexive Prompting for Large-Language-Models*. arXiv: 2408.05093 [cs.CL].

Yang, H. et al. (2024). *Number Cookbook: Number Understanding of Language Models and How to Improve It*. arXiv: 2411.03766 [cs.CL].

DeepSeek-AI (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv: 2501.12948v1 [cs.CL].

Guo, D. et al. (2025). "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning". In: *Nature* 645, pp. 633–638.

OpenAI (12 September 2024). Learning to reason with LLMs.

Backup: sources for the milestones

OpenAI (30 November 2022). Introducing ChatGPT.

Google DeepMind (25 July 2024). IMO silver-medal standard.

Google DeepMind (21 July 2025). IMO gold-medal standard.

Wagner, A. Z. (2021). *Constructions in combinatorics via neural networks*. arXiv: 2104.14516 [math.CO].

Romera-Paredes, B. et al. (2023). "Mathematical discoveries from program search with large language models". In: *Nature*.

Alon, N. et al. (2026). *Remarks on the disproof of the unit distance conjecture*. arXiv: 2605.20695 [math.CO].

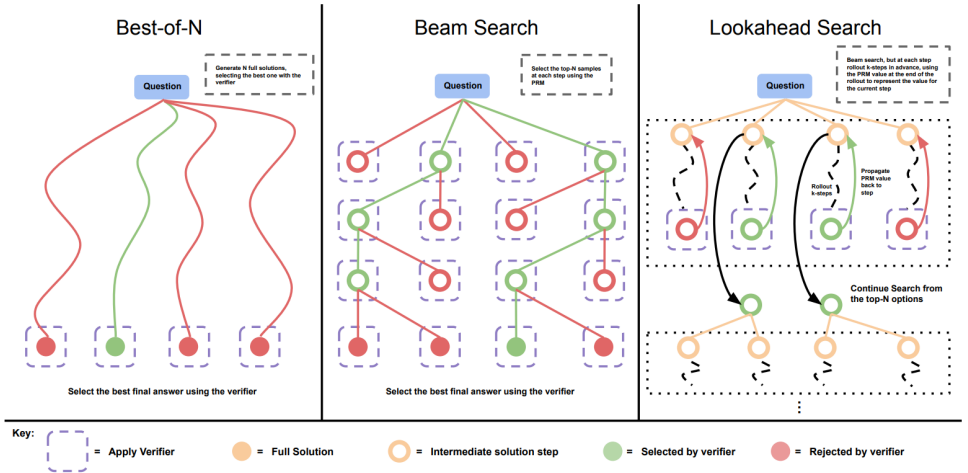
Sothanaphan, N. (2026). *Resolution of Erdős Problem #728: a writeup of Aristotle's Lean proof*. arXiv: 2601.07421 [math.NT].

OpenAI (8 September 2026). On the Navier–Stokes Millennium Prize Problem.

Clay Mathematics Institute (11 September 2026). Navier–Stokes announcement.

OpenAI (6 October 2026). Sharing AI progress in mathematics.

Backup: Test-time compute



Snell et al. 2024, *Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters*, Fig. 2.