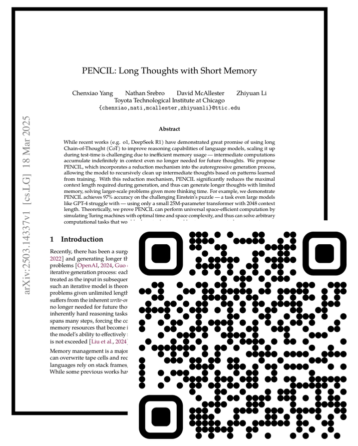
I have discovered a truly marvelous proof of this theorem, which this margin is too small to contain.

Fermat, 1637

NO! You can show it with a pencil (and an eraser)!



Yang et al., 2025

PENCIL : Long Thoughts with Short Memory

Chenxiao Yang, Nathan Srebro, David McAllester, Zhiyuan Li

Toyota Technological Institute at Chicago (TTIC)

Method

Scaling up the thoughts at test time can significantly improve the reasoning capability of LLMs (e.g. GPT-o1/o3, DeepSeek R1) !

BUT, very long CoT causes problems such as excessive memory, slow inference, attention dilution, etc. Is standard CoT fundamentally limited?

What is PENCIL ?

Model Generation (Write):

$$\mathbf{C}_1 \;\Rightarrow\; \mathbf{C}_1\ \mathbf{C}_2\ \text{[CALL]}\ \mathbf{T}\ \text{[SEP]}\ \mathbf{A}\ \text{[RETURN]}$$

Reduction Rule (Erase):

$$\mathbf{C}\ \text{[CALL]}\ \mathbf{T}\ \text{[SEP]}\ \mathbf{A}\ \text{[RETURN]} \;\Rightarrow\; \mathbf{C}\ \mathbf{A}$$

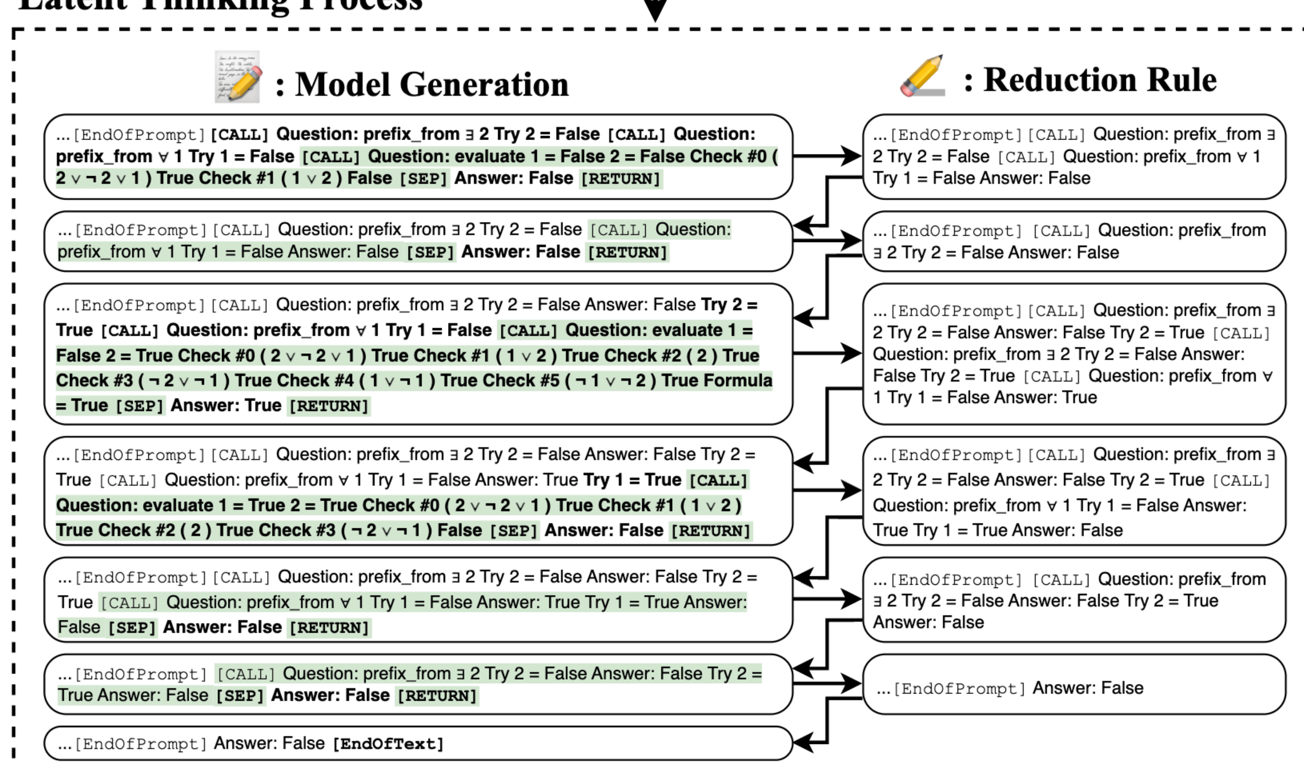
- [CALL], [SEP], [RETURN] are special tokens
- **C** = Context, **T** = Thoughts, **A** = Answer (*They can include special tokens!*)
- Reduction is triggered when the sequence matches the pattern
- PENCIL iteratively generate thoughts and triggers reduction

Important : *Special tokens can accumulate similar with nested function calls in functional programming!*

How PENCIL Works? (Example on QBF)

Prompt ∃ 2 ∨ 1 : #1 (2 ∨ ¬ 2 ∨ 1) #2 (1 ∨ 2) #3 (2) #4 (¬ 2 ∨ ¬ 1) #5 (1 ∨ ¬ 1) #6 (¬ 1 ∨ ¬ 2)

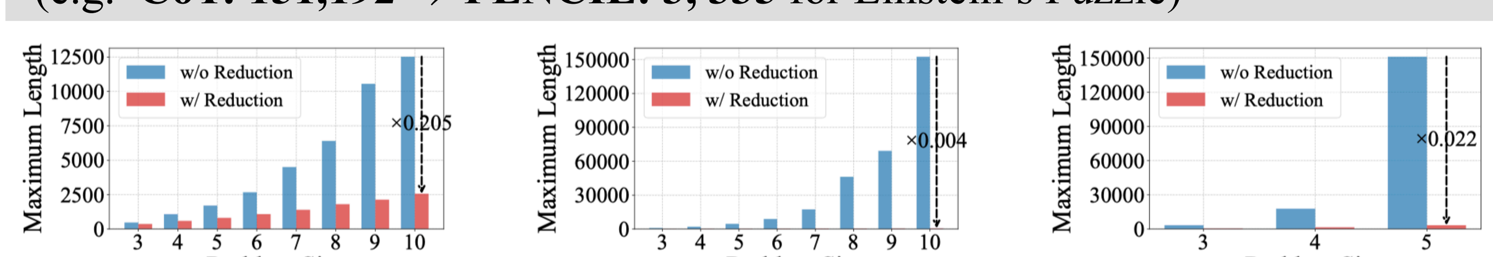
Latent Thinking Process



Response Answer: False

PENCIL can do (tail) recursion, backtrack, decomposition, etc.

PENCIL significantly reduces the maximal context length during inference (e.g. **CoT: 151,192 ⇒ PENCIL: 3, 335** for Einstein's Puzzle)



(a) SAT (b) QBF (c) Einstein's puzzle

Experiments

We train a small transformer (25M parameter, 2048 context length) to use PENCIL to solve computationally intensive reasoning tasks (SAT, QBF, Einstein's Puzzle).

$$\textbf{Training :}\ \mathcal{L}_{\textbf{CoT}} = -\sum \log p(\textbf{next token} \,|\, \textbf{previous sequence})$$

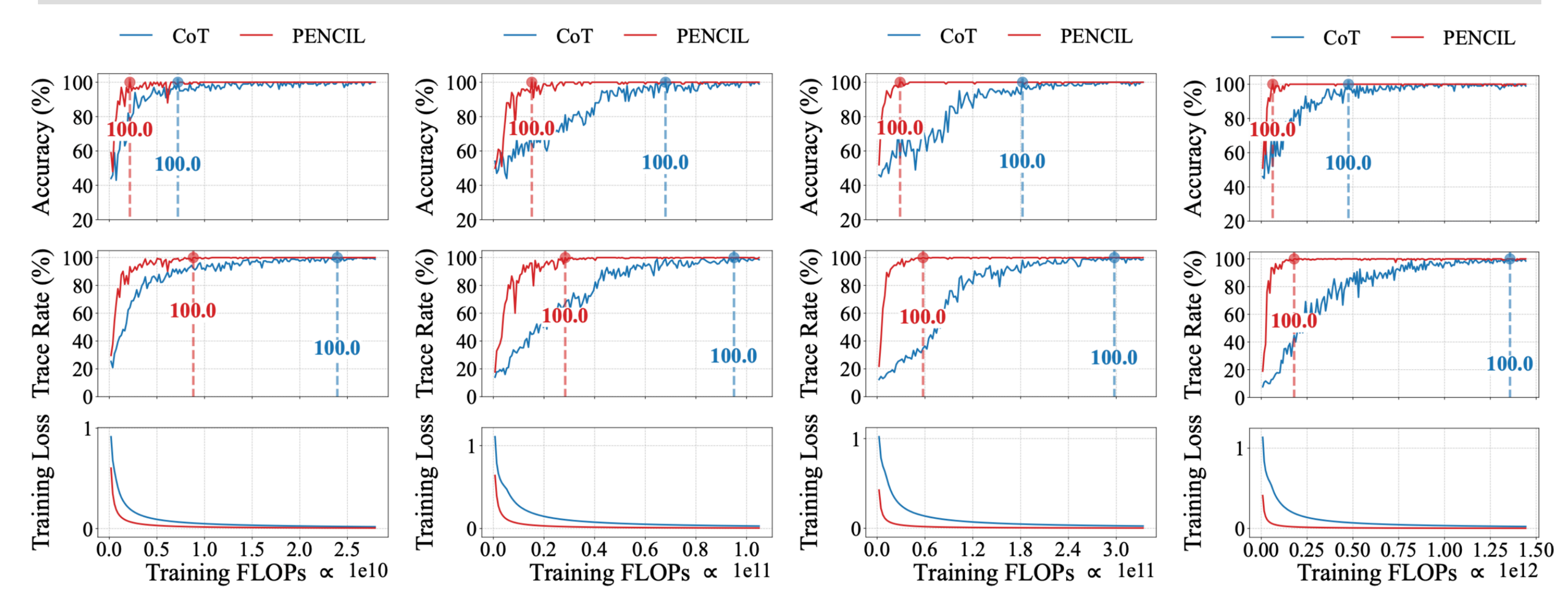
$$\mathcal{L}_{\textbf{PENCIL}} = -\sum \log p(\textbf{next token} \,|\, \textbf{reduced sequence})$$

Performance: PENCIL outperforms CoT on NP-hard tasks SAT and QBF by a large margin (i.e. almost perfect v.s. random guessing).

	$n=$	3	4	5	6	7	8	9	10
Baseline	Acc.	66	57	46	51	46	51	49	51
CoT	Acc.	100	100	100	99	84	63	54	50
	TR.	99.6	99.0	98.0	96.2	74.0	69.9	63.8	51.4
PENCIL	Acc.	100	100	100	99	99	100	100	100
	TR.	100	99.0	97.1	95.9	91.8	93.3	92.9	83.0

	$n=$	3	4	5	6	7	8	9	10
Baseline	Acc.	90	82	85	68	60	69	71	66
CoT	Acc.	100	100	97	94	74	72	69	73
	TR.	100	100	98.3	93.9	65.1	49.4	40.7	32.8
PENCIL	Acc.	100	100	100	100	100	100	100	100
	TR.	100	100	100	100	100	100	100	100

Time Efficiency: PENCIL significantly saves computations by reducing the context length and converges faster during training.



(a) QBF $n=3$ (b) QBF $n=4$ (c) QBF $n=5$ (d) QBF $n=6$

Applicability : PENCIL works as well on a natural-language reasoning task called Einstein's puzzle – a logic puzzle that even GPT-4 struggles with.

Einstein's Puzzle

- **Constraint 1 :** The green house is immediately to the right of the one who keeps birds
- **Constraint 2 :** The Brit is immediately to the right of the German
- **Constraint 3 :** The one who keeps dogs is the same house as the red house
- **Constraint 4 :** The one who keeps birds is immediately to the right of the Swede

Question: who owns the fish?

→

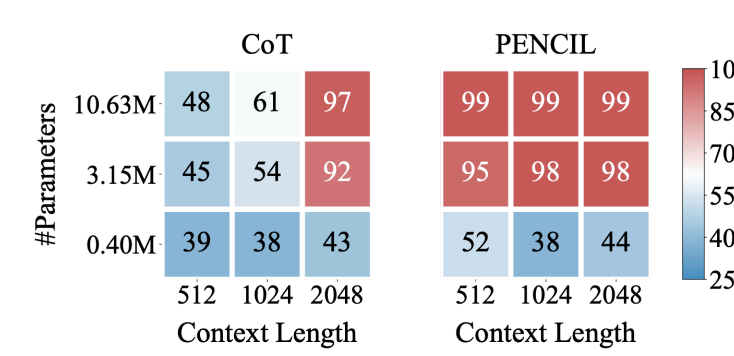
PENCIL

Solution :

House #	1	2	3
Color	Red	Blue	Green
Nationality	Swede	German	Brit
Pet	Dogs	Birds	Fish

Answer: the Brit owns the fish

Puzzle Size		CoT	PENCIL
5 × 5	Accuracy (%)	25	97
	Trace Rate (%)	2.97	78.27
4 × 4	Accuracy (%)	34	100
	Trace Rate (%)	8.33	86.52
3 × 3	Accuracy (%)	99	99
	Trace Rate (%)	99.37	99.66

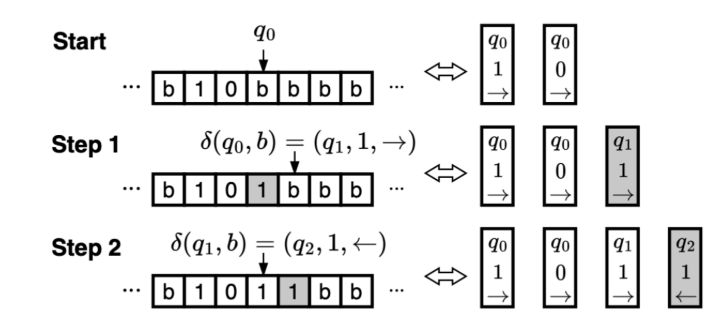


Theory

Theorem (Main, Informal). For any Turing machine, there exists a fixed finite-size transformer such that for any input, on which Turing machine uses T steps and S space to compute, **PENCIL** with this transformer computes the same output with $\mathcal{O}(T)$ **generated tokens** and using $\mathcal{O}(S)$ **maximal context length**. *(This is impossible for CoT !)*

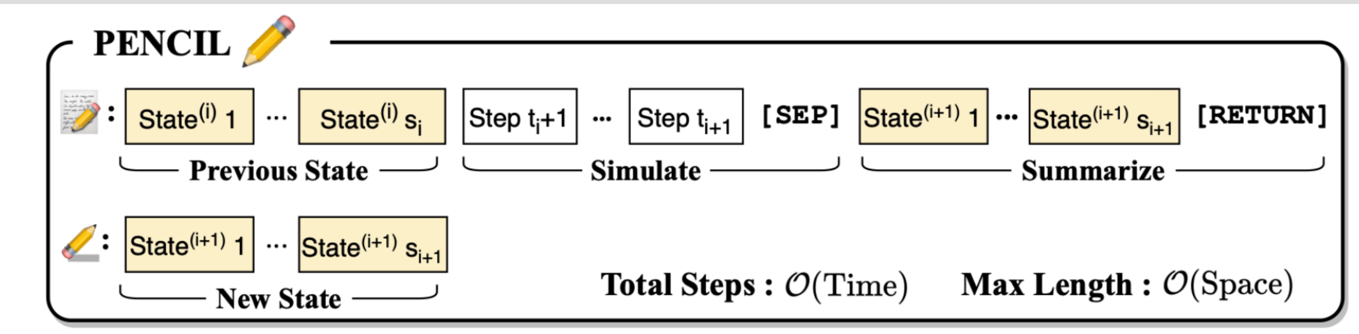
Corollary (Informal). With poly(n) context length, PENCIL can solve all problems in PSPACE, while standard CoT can only solve problems in P.

Step 1 : Turing Machine computation ⇔ Next-token generation (CoT)



- Each generated token represents a TM transition step, encoding (*state*, *symbol*, *movement direction*).
- **Total Steps** = $\mathcal{O}$(**Time**)
- **Max Context Length** = $\mathcal{O}$(**Time**)

Step 2 : Simulating TM space-efficiently with PENCIL



- **Simulation:** generate tokens to simulate the step-by-step running of Turing machine, starting from the previous state (i.e. TM's configuration).
- **Summarization:** summarize all previously tokens into the new state using the PENCIL reduction rule.
- Trigger the summarization whenever the length of current sequence exceeds twice the actual space needed to store the information.
- **Total Steps** = $\mathcal{O}$(**Time**), **Max Context Length** = $\mathcal{O}$(**Space**)

Step 3 : Can Transformers express the algorithm ?

Full-Access Sequence Processing (FASP)

A program in FASP is a process of building a sequence of increasingly complex functions (i.e. Transformers) $\Sigma^* \to \mathbb{R}^d$.

- **Each Variable** = a Transformer.
- **Each Line of Code** = an operator from some simpler Transformers to a more complex Transformer.
- **FASP** predefines a set of local and non-local operators, and supports custom operators.

Theorem (Informal). FASP = $\mathcal{H}_{\mathrm{TF}}$

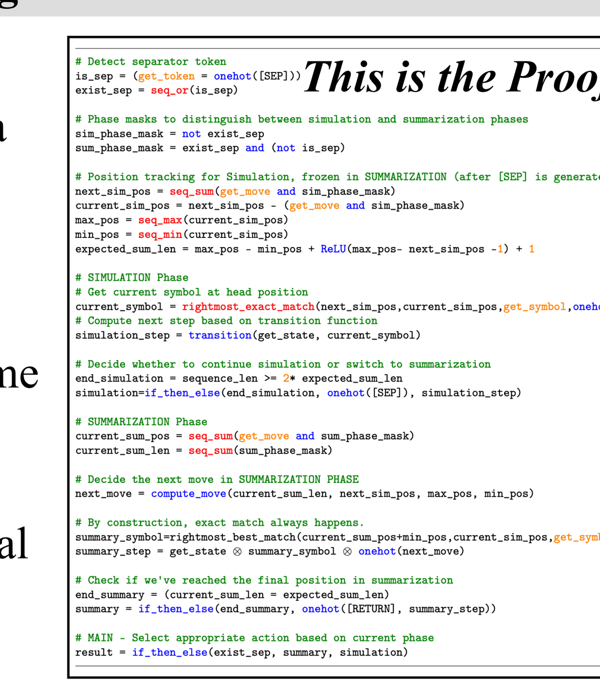


Fig. A program written in FASP

👉 : ∃ a Transformer expresses Step 2 ⇔ ∃ a FASP program implements Step 2