CAC-CoT: Connector-Aware Compact Chain-of-Thought for Efficient Reasoning Data Synthesis Across Dual-System Cognitive Tasks

Sunguk Choi* **Yonghoon Kwon*** **Heondeuk Lee***
DATUMO
{sunguk.choi, yonghoon.kwon, heondeuk.lee}@selectstar.ai

Abstract

Long chain-of-thought (CoT) prompting helps Large Language Models (LLMs) solve difficult problems, but very long traces often slow or even degrade performance on fast, intuitive "System-1" tasks. We introduce Connector-Aware Compact CoT (CAC-CoT) — a method that deliberately restricts reasoning to a small, fixed set of connector phrases, steering the model toward concise and well — structured explanations. Despite its simplicity, our synthetic method with general-purpose LLMs yields a high-quality training quality. CAC-CoT achieves ≈ 85% on GSM8K and ≈ 40% on GPQA (System-2) while also achieving ≈ 85% on S1-Bench (System-1), surpassing the baseline by over 20%. Its reasoning traces average ≈ 300 tokens(ART), about one-third the length of baseline traces, delivering higher efficiency without loss of accuracy.[1]

1 Introduction

Large Language Models (LLMs) have achieved striking gains on reasoning tasks by producing explicit chain-of-thoughts (CoT) rationales (Wei et al., 2022). For complex problems that demand slow, analytical System-2 thinking, they deploy Long CoTs enriched with self-reflection, backtracking and budget-forcing — an approach adopted by frontier systems such as OpenAI's o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025) and the inference-time scaling method s1 (Muennighoff et al., 2025a) — and one that has markedly improved performance on challenging reasoning benchmarks.

Frontier Reasoning LLMs — e.g., from OpenAI, Anthropic and Google — internalize long-chain reasoning techniques and dynamically adjust CoT length, thereby achieving state-of-the-art results on demanding benchmarks like AIME and GPQA (Jaech et al., 2024; Anthropic, 2025; Google, 2025). Recent open-source initiatives follow the same trajectory. Representative efforts include RedStar (Xu et al., 2025b) and SkyThought (Li et al., 2025a), which boost reasoning performance by fine-tuning on large Long CoT corpora with reinforcement learning (RL), while LIMR (Li et al., 2025b) shows that carefully selected subsets can yield comparable gains at lower cost. Complementary work demonstrates that small and curated reasoning datasets alone can endow compact models with strong reasoning skills via supervised fine-tuning (SFT); notable examples include LIMO and the inference-time scaling method s1 (Ye et al., 2025a; Muennighoff et al., 2025a).

Despite their promise, integrating Long-CoT rea-

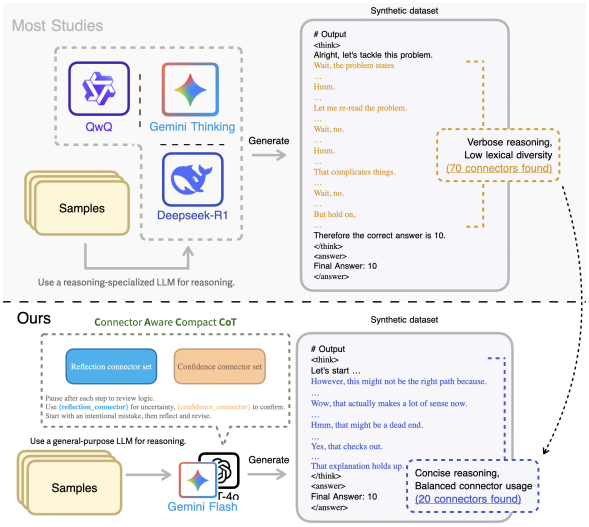

Figure 1: **Comparison of reasoning trace generation.** Most studies (top) use standard LLMs to generate verbose, repetitive reasoning with excessive connector usage. In contrast, our Connector-Aware Compact CoT framework (bottom) employs explicit connector control to produce concise, coherent reasoning traces with significantly fewer connectors, enabling efficient and high-quality data generation.

*Equal contribution

[1]Code is publicly available at https://github.com/selectstar-ai/CAC-CoT

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 19515–19530
November 4-9, 2025

soning into LLMs introduces significant drawbacks. Models fine-tuned on very long traces tend to generate equally lengthy chains at inference time, a phenomenon termed overthinking — they continue to reason well past the point of finding a correct answer (Chen et al., 2025). Such gratuitous deliberation inflates computation and may even degrade accuracy, as unnecessary steps accumulate noise and contradictions. Recent evaluations corroborate this trade-off: Long-CoT-optimized models excel on System-2 thinking benchmarks yet falter on intuitive, System-1 tasks (Zhang et al., 2025). These findings underscore a pressing need for balanced dual-system reasoning in LLMs — models must learn to invoke fast, heuristic reasoning for simple queries while reserving slow, analytical reasoning for genuinely hard problems.

In this paper, we propose **Connector-Aware Compact Chain-of-Thoughts** (CAC-CoT), a data-generation framework that bridges the gap between analytical System-2 reasoning and agile System-1 intuition. CAC-CoT prompts frontier LLMs to produce reasoning traces that (i) insert explicit connector sentences — e.g., "Let's pause and rethink this.", "Wow, that actually makes a lot of sense now." — as deliberate checkpoints for self-reflection, whether in correct or incorrect reasoning, and (ii) enforce structural compactness that discourages gratuitous length. These connectors cue the model to pause, verify, or backtrack only when necessary, preventing runaway verbosity while preserving logical coherence. In effect, it trains models to balance dual-system reasoning: they engage thorough analytical reasoning when warranted yet default to concise, intuitive responses on simpler queries. The proposed method is outlined in Figure 1.

To build a high-quality CAC-CoT dataset, we leverage general-purpose LLMs — cost-efficient but capable frontier models — using a simple single-turn prompt that elicits concise, connector-rich traces. Despite the dataset's modest size and simplicity of the prompting strategy, fine-tuning target models on this data yields strong reasoning performance. The result is an exceptionally economical training recipe that bypasses the need for large Long CoT corpora or complex pipelines.

Comprehensive experiments confirm CAC-CoT's effectiveness. On demanding System-2 thinking benchmarks, our model attains $\approx 85\%$ on GSM8K and $\approx 40\%$ on GPQA (Rein et al., 2024) with Long-CoT fine-tuned specialists. On S1-Bench (Zhang et al., 2025), which stresses intuitive System-1 thinking, Ours reaches ≈ 85 %, representing an improvement of over 20% points compared to the s1.1 and LIMO baselines. Efficiency follows suit: the model averages an ART of ≈ 300, surpassing nearly all prior systems in computational economy. Collectively, these results demonstrate that connector-based, compact reasoning traces are a promising foundation for future reasoning data sets and models.

2 Related works

2.1 Long Chain-of-Thought

Recent advancements in CoT prompting have significantly enhanced the reasoning abilities of Large Language Models (LLMs) by decomposing complex tasks into intermediate steps (Wei et al., 2022). Early work mainly relied on static, human-written exemplars, but subsequent studies introduced self-reflection and backtracking mechanisms to generate longer yet more adaptive reasoning traces. For instance, Reflexion prompts an agent to store episodic self-feedback between trials (Shinn et al., 2023), whereas Self-Refine iteratively rewrites its own output until self-criticism converges (Madaan et al., 2023). Building on this idea, Pang et al. (2024) directly optimizes pairwise preferences between good and bad traces, and Adarsh et al. (2024) distills high-quality rationales into smaller models via self-guided cycles of error detection and regeneration.

In other research directions on Long CoT, there is ongoing debate regarding the optimal length and granularity of reasoning traces. (Yao et al., 2023a) argue that concise, structured traces improve efficiency without harming accuracy, whereas (Shen et al., 2025) and (Snell et al., 2024) present scaling laws showing that longer reasoning often improves performance on difficult tasks, particularly for large models. Recent observations suggest that overly lengthy reasoning traces can result in diminishing returns or even performance degradation due to noise accumulation and increased inference costs (Chen et al., 2025). This phenomenon, often termed overthinking, disproportionately hampers smaller models, which struggle to assimilate and faithfully reproduce the excessively long reasoning chains that arise when they are trained on Long CoT data (Li et al., 2025c; Wu et al., 2025; Xu et al., 2025a; Zhang et al., 2025).

2.2 Data Efficiency and Overthinking

While architectural advances have driven recent gains in reasoning performance, growing evidence highlights that the structure and quality of training data play an equally critical role (Li et al., 2025a; Swayamdipta et al., 2020). In particular, Long Chain-of-Thought (Long CoT) supervision has proven effective for inducing structured reasoning. However, such traces are often verbose, increasing both training cost and the risk of overfitting to unnecessary steps (Chen et al., 2025).

Recent studies attempt to mitigate this verbosity by emphasizing logical sufficiency over exhaustiveness. For instance, Sky-T1 (Li et al., 2025a) enhances reasoning robustness by filtering out incorrect reasoning traces rather than compressing them. Meanwhile, LIMO (Ye et al., 2025a) curates high-quality datasets and applies multi-step supervision to highlight only the essential logical steps. Although neither approach directly targets brevity, both demonstrate the potential to improve training efficiency by focusing on core reasoning content. These strategies reflect a broader trend toward data-centric LLM development, where the structure and quality of supervision play a decisive role in downstream reasoning behavior.

Despite improvements, overthinking remains a failure mode. Models often produce overly detailed reasoning even for intuitive tasks, leading to redundancy, inconsistencies, and degraded performance—particularly on System-1 benchmarks (Sui et al., 2025; Zhang et al., 2025). This issue is pronounced in smaller models, where excessive elaboration introduces noise or contradictions (Yao et al., 2023b; Chen et al., 2024). Even compact prompting strategies can backfire when misapplied to simple problems, underscoring the need for task-aware reasoning generation that adjusts explanation depth to problem complexity.

2.3 Dual-System Reasoning

Dual-system theory distinguishes between two types of cognition: fast, intuitive System-1 reasoning, and slow, analytical System-2 reasoning (Kahneman, 2011; Kannengiesser and Gero, 2019). Although large reasoning models excel at complex System-2 tasks, their performance often deteriorates on simpler System-1 tasks due to unnecessarily elaborate reasoning strategies (Zhang et al., 2025). This highlights a critical limitation—current models often lack the cognitive flexibility needed to generalize across varied reasoning demands.

Early attempts to address this limitation took divergent approaches. LIMO curated a minimal yet challenging question set requiring detailed reasoning and manually verified each reasoning step for high-quality supervision. However, its scalability is limited by the labor-intensive nature of manual annotation. In contrast, Sky-T1 distilled reasoning patterns into fixed-length traces via offline analysis and trained LoRA adapters, sacrificing adaptability and stylistic diversity for inference simplicity.

CAC-CoT adopts a different approach by removing reliance on large reasoning models (LRMs) during trace generation. Instead of imitating model-generated reasoning trajectories, CAC-CoT explicitly injects connector phrases into the generation process. This strategy leverages existing high-quality question datasets (such as those from LIMO) while guiding models to produce concise, cognitively aligned reasoning traces. Notably, CAC-CoT achieves this without requiring chain-level annotations or reinforcement learning, thereby promoting both training efficiency and dual-system adaptability. Its connector-driven approach allows for controllable variation in reasoning form while maintaining semantic coherence, offering a scalable path toward more flexible and cognitively grounded reasoning systems.

3 Methodology

Building on recent advances in reasoning models — which demonstrate significant performance gains through mechanisms such as self-reflection, backtracking, and self-correction — ours, **CAC-CoT**, adopts a structured approach that explicitly injects diverse connector phrases into the training data. By deliberately guiding the model's reasoning behavior through these connectors and tightly controlling the length of reasoning traces compared to conventional Long-CoT datasets, CAC-CoT enables the emergence of a robust dual-system reasoning capability. This design allows the model to adaptively balance concise, intuitive responses for System-1 tasks with deeper, structured reasoning for System-2 challenges, thereby achieving strong and generalizable performance across both cognitive regimes. The synthetic data generation prompts and the corresponding generation logic can be found in Appendix B, in Table 9 and Algorithm 1, respectively.

3.1 Connector-Aware CoT

In reasoning models, a variety of connectors (e.g., Wait, Hmm, Alternatively) are often used to facilitate the flow of reasoning, particularly through mechanisms such as self-reflection and backtracking. Inspired by this observation, we propose a data construction method that explicitly injects connectors into the reasoning process. This encourages the model to maintain or even enhance its reasoning performance by following a structured and reflective approach. The key components of our method are as follows: (1) Insert checkpoints after each reasoning step for reassessment; (2) use reflection connectors at checkpoints to signal uncertainty and enable revision of faulty logic; (3) use confidence connectors when prior logic is valid to confirm reasoning and move forward; (4) begin with an intentionally flawed reasoning path to promote reflective correction and generate extended traces.

By prompting verification at every reasoning step and initially inducing errors to ensure that reasoning chains remain appropriately maintained, and by applying confidence and reflection connectors as needed, we steer the process to achieve both exploration and convergence. Details on connector usage are provided in Section4.4, and the types of connectors are summarized in Table 10 of Appendix B.

3.2 Compact CoT

Reasoning models often suffer from excessively long reasoning traces, which can lead to inefficiency and even performance degradation on System-1 thinking tasks. To mitigate this issue, impose explicit termination constraints by limiting the number of validations, bounding the trace length, and defining clear stopping conditions. Moreover, the two types of connectors — reflection connector and confidence connector — further enhance the compactness of the reasoning steps, with the confidence connector in particular facilitating concise reasoning process.

By alternating these two types of connectors, selectively guide the model to either expand or converge its reasoning process. This prevents unnecessary elaboration and encourages timely termination, preserving overall reasoning performance while significantly improving performance on System-1 tasks. The termination strategy consists of the following rules: (1) Disallow consecutive use of connectors to avoid incoherent chaining of reasoning steps; (2) skip further validation if the same answer is produced more than once; (3) invoke the termination condition if the reasoning becomes unclear or overly repetitive; and (4) trigger termination if the reasoning trace exceeds a predefined length or the number of validations surpasses a threshold.

By preventing reasoning steps from expanding excessively, we ensure suitably concise reasoning traces. In particular, by imposing length limits and instructions to avoid overly repetitive steps, we achieve efficient reasoning. While triggering the termination condition does not directly improve accuracy, it prevents trace length blow-up during data generation. As shown in Section 4.4, the confidence connector terminates further progression upon successful inference, thereby promoting compactness.

4 Experiments

4.1 Experimental Setup

In this section, we describe our experimental setup and present the main findings of the dual-system benchmark.

4.1.1 Benchmarks

For analytical System-2 thinking, we select widely used math-centric datasets of escalating difficulty — AMC, AIME, GPQA, GSM8K and MATH. Collectively, these benchmarks pose problems that resist easy solutions, driving the model to explore multiple angles and engage in deep reasoning, making them suitable for validation analytical System-2 thinking. For intuitive System-1 thinking, we follow the evaluation metrics of S1-Bench (Zhang et al., 2025), a collection of commonsense and rapid-inference questions that can usually be solved with minimal deliberation. By contrasting performance across these two suites, we can measure whether CAC-CoT preserves fast, concise intuition while enhancing deep analytical skill.

4.1.2 Baselines

To contextualize the impact of CAC-CoT, we benchmark against three recent models that rely solely on SFT over Long-CoT data to achieve substantial gains in reasoning performance via System-2 thinking. s1.1 (Muennighoff et al., 2025a), LIMO (Ye et al., 2025a), and Bespoke-Stratos (Labs, 2025) all rely on SFT over a compact Long-CoT corpus synthesized by powerful reasoning engines (R1, R1-Distill-Qwen-32B, and QwQ, respectively) (Guo et al., 2025; DeepSeek-AI, 2025;

Table 1: **System 1 thinking performance across models on S1-Bench.** Comparison of evaluation metrics (Acc@5, Pass@1, Success, ART) for English (EN) and Chinese (ZN) tasks. Qwen-2.5-7B-Instruct is evaluated under loose formatting, while all other models are evaluated under strict formatting. CAC-CoT-7B (Ours) achieves strong accuracy with the lowest average reasoning length (ART), demonstrating superior efficiency and balanced performance across all categories. **Formatting:** The AVG row is highlighted in **bold**, while notable values in all other rows are underlined for emphasis.

Models	Task Type	EN				ZN			
		Acc@5 ↑	Pass@1 ↑	Success ↑	ART ↓	Acc@5 ↑	Pass@1 ↑	Success ↑	ART ↓
Qwen2.5-7B-Instruct	analysis_question	100.0	100.0	100.0	49.8	94.44	96.39	100.0	37.76
	instruction_following	26.47	57.06	100.0	6.79	13.79	21.38	100.0	10.54
	knowledge_question	62.75	80.00	100.0	48.40	13.21	25.28	100.0	46.02
	reasoning_question	66.67	74.67	100.0	67.08	41.67	62.08	100.0	51.61
	AVG	63.97	77.93	100.0	43.02	40.78	51.28	100.0	36.48
Bespoke-Stratos-7B	analysis_question	100.0	100.0	100.0	830.43	75.0	95.24	99.17	408.97
	instruction_following	58.82	96.69	88.82	1026.77	65.52	95.59	93.79	771.93
	knowledge_question	100.0	100.0	100.0	830.62	88.68	97.36	100.0	460.63
	reasoning_question	93.33	98.66	99.67	836.27	77.08	94.98	99.58	545.82
	AVG	**88.04**	98.84	97.12	881.02	76.57	95.79	**98.14**	546.84
s1.1-7B	analysis_question	74.67	99.16	94.93	573.77	63.89	99.39	91.11	299.44
	instruction_following	47.06	98.55	81.18	2041.02	41.38	99.19	84.83	1109.58
	knowledge_question	80.39	100.0	94.12	848.53	84.91	100.0	96.23	329.6
	reasoning_question	70.0	99.28	92.67	1088.92	41.67	99.48	80.83	490.29
	AVG	68.03	**99.25**	90.73	1138.06	57.96	**99.25**	88.25	557.23
LIMO-7B-reproduced	analysis_question	49.33	85.91	96.53	806.91	5.56	60.79	63.06	368.22
	instruction_following	17.65	89.52	61.76	1633.84	41.38	90.29	71.03	1111.96
	knowledge_question	72.55	92.21	95.69	975.48	35.85	96.24	70.19	573.94
	reasoning_question	56.67	81.56	94.0	1144.3	45.83	78.0	83.33	643.32
	AVG	49.05	87.30	87.00	1140.13	32.16	81.33	71.90	674.36
CAC-CoT-7B (Ours)	analysis_question	97.33	99.2	100.0	273.97	90.28	98.33	99.72	174.12
	instruction_following	67.65	98.12	94.12	306.82	65.52	96.35	94.48	287.83
	knowledge_question	84.31	99.18	96.08	256.12	84.91	99.61	97.36	177.47
	reasoning_question	95.00	98.67	100.0	308.13	85.42	97.49	99.58	226.16
	AVG	86.07	98.79	**97.55**	**286.26**	**81.53**	97.95	97.78	**216.39**

Team, 2025). Despite the modest data volume, each baseline is reported to extract unexpectedly strong reasoning ability from its target model. s1.1 and Bespoke-Stratos are evaluated exactly as released in huggingface (SimpleScaling Team, 2024; Labs, 2024), maintaining full reproducibility. but, LIMO was experimented on the 32B model, and no experimental results were reported for the 7B model; we therefore reproduce the same method in ours (s1.1 training method), and perform an identical SFT run on Qwen-2.5-7b-Instruct (Team, 2024) checkpoint to ensure a fair comparison.

4.1.3 Training Details

Training. For fine-tuning, we adopted Qwen-2.5-7B-Instruct as the base model. Our training followed the same hyperparameter configuration used in s1.1 to ensure comparability. Further implementation details, including batch size, optimizer settings, and training duration, are provided in the Appendix A.

Datasets. We generated our training data using general-purpose LLMs, Gemini-2.0-Flash and GPT-4o (OpenAI et al., 2024), cost-effective yet capable frontier models. The performance of our method (Ours) reported in Section 4 is based on synthetic data generated with Gemini-2.0-Flash. To examine whether training signals may have been influenced by the style and bias of a single model, we additionally report results using GPT-4o generated data in Appendix C.1. During the data generation process, we filtered out instances that exhibited generation errors or failed to follow our specified formatting instructions. As a result, the total usable output was reduced. To ensure sufficient coverage and diversity, we supplemented this data with samples from the LIMO and s1 datasets (Ye et al., 2025b; Muennighoff et al., 2025b). Specifically, we removed any duplicates between LIMO and s1 and excluded all corrupted or invalid generations. After this filtering process, we finalized a training set consisting of **1,391** examples. Details of the data generation process are provided in Appendix B.

4.2 System 1 Thinking

As summarized in Table 1, our experiments reveal that CAC-CoT consistently achieves strong, state-of-the-art results on System-1 tasks.

We evaluate our method on the S1-Bench suite, which consists of tasks solvable via quick, intuitive reasoning ("System-1" thinking). These tasks generally require little to no step-by-step deduction, so an effective effective should answer accurately without over-elaborating. On this benchmark, it performs on par with or slightly better than the baselines in terms of accuracy: all methods achieve high scores on these easier queries (as expected), but importantly, ours does not sacrifice performance on simple tasks despite its emphasis on CoT. In fact, CAC-CoT attains the highest overall ACC@5 and PASS@1 on S1-Bench (Table 1), albeit with marginal improvement since the baseline performance is near-saturated.

Crucially, its responses on System-1 tasks are compact and to-the-point, highlighting adaptability. Whereas naive CoT prompting might introduce unnecessary steps or verbiage for trivial questions, our approach generates minimal reasoning — or sometimes goes straight to the answer — when extensive explanation is unnecessary. Our average reasoning trace length is the shortest among all evaluated methods on S1-Bench (Table 1, ART), indicating that it avoids "overthinking" simple problems.

This connector-aware strategy compresses reasoning traces to roughly one-third the length of baselines while preserving — and in some cases improving — accuracy. By constraining unnecessary expansion of the CoT, it avoids overthinking pitfalls and shows that instruction-following architectures — when guided by targeted connector signals — can overcome the compliance weakness of seen in previous studies on Long CoT reasoning. Empirically, this design delivers over 20% accuracy gain compared to both s1.1 and LIMO, confirming its strength on straightforward System-1 tasks without sacrificing deliberative performance.

4.3 System 2 Thinking

CAC-CoT maintains strong performance on System-2 benchmarks, demonstrating comparable reasoning ability to prior Long-CoT baselines. While there is a slight performance penalty relative to baseline models — whose results are detailed in Table 2 — the overall degradation is minor, and Ours successfully preserves analytical reasoning capabilities without relying on large-scale reasoning-specialized traces.

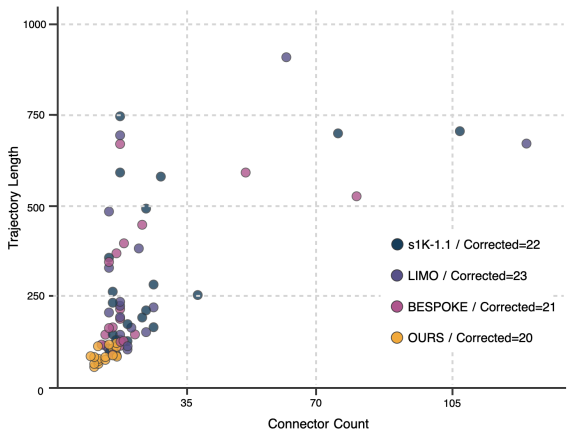

Figure 2: **Compactness of Chain-of-Thought Traces by Model.** Scatter plot of reasoning-trace length (y-axis) versus connector count (x-axis) for AMC23 outputs. Points farther toward the lower-left denote shorter, more compact traces.

A key distinction lies in the construction of our training data. Unlike existing approaches that depend heavily on dedicated reasoning models to generate Long-CoT corpora — often resulting in verbose outputs with excessive connector usage — it employs a connector-aware generation strategy based on Gemini-2.0-flash, a general-purpose but instruction-aligned model. This allows us to retain the reasoning efficacy of connector-based CoT without inflating either the length of reasoning traces or the frequency of connector insertions. As shown in Figure 2, our outputs are significantly more compact, with the CAC-CoT cluster consistently positioned in the lower-left region, indicating fewer connectors and greater efficiency than competing models. In summary, CAC-CoT offers a principled trade-off: it slightly underperforms the strongest reasoning-optimized baselines, yet does so with far more economical prompting and without the bloat of overly long reasoning chains. This makes it a compelling choice for efficient, scalable reasoning under constrained data and compute regimes.

4.4 Connector Usage

To better understand the efficiency and structure of reasoning under CAC-CoT, we jointly analyze the distribution of connector usage and output length across both the training data and inference outputs, comparing against baselines (Table 3 , Figure 2). These two aspects — connector count and sequence

Table 2: **System 2 thinking performance across benchmarks.** Accuracy comparison on five reasoning benchmarks, including AMC23, AIME24, GSM8K, GPQA Diamond and Math500. The row in-between highlights the difference between CAC-CoT-7B (Ours) and the best baseline in each column (orange: negative, blue: positive).

Models	AMC23	AIME24	GSM8K	GPQA Diamond	Math500	AVG
Qwen2.5-7B-Instruct	55.00	6.67	79.98	33.84	75.00	50.09
s1.1-7B	55.00	13.33	90.67	39.39	79.40	55.55
LIMO-7B-reproduced	57.50	13.33	88.55	35.35	78.20	54.58
Bespoke-Stratos-7B	52.50	23.33	88.25	43.94	80.20	57.64
CAC-CoT-7B (Ours)	–5.0 50.00	+3.33 10.00	+5.39 85.37	+4.54 38.38	–7.0 68.00	+0.26 50.35

Table 3: **Connector usage comparison between open-source LongCoT corpora and our corpus.** Our corpus yields significantly shorter reasoning traces with lower connector density compared to widely used open-source LongCoT datasets. **Len**: average trajectory length (tokens). **Conn/1K**: connectors per 1,000 tokens. **# Samples**: number of samples.

Dataset	Len	Conn/1K	# Samples
S1K-1.1	9291.62	5.55	1k
LIMO	6984.09	2.97	0.8k
BESPOKE	4452.22	5.13	16.7k
OURS	1843.43	2.65	1.4k

length — serve as proxies for reasoning verbosity and structural control, and offer insight into how different models manage the trade-off between expressiveness and conciseness. Table 3 summarizes connector usage and token lengths for our training corpora versus the baselines. The Bespoke dataset averages approximately 4500 tokens per reasoning trace with 5 connectors per 1000 tokens, while s1K averages around 9000 tokens and 5.5 connectors per 1000 tokens. In contrast, our connector-aware data exhibits a far more controlled structure, with an average trace length of only 1850 tokens and just 2.5 connectors per 1000 tokens. This demonstrates that our data construction yields an efficient reasoning corpus, which in turn benefits inference. Figure 2 shows connector counts and token lengths at inference time. As evident, the compact structure of the training data carries over to outputs: on average the model uses only about 20 connectors and produces fewer than 200 tokens per response. Baseline models, by comparison, sometimes exceed 500 tokens or include over 70 connectors. Together with the accuracy efficiency improvements reported in Sections 4.2 and 4.3, these results confirm that our connector-aware approach delivers both concise and efficient reasoning.

Additionally, we perform a connector-aware qualitative analysis on the generated training data. Table 4 displays how the data patterns are structured before and after each connector. The first is the reflection connector, which reflects known mechanisms such as self-reflection and backtracking. Theses connectors encourage exploratory or verification-oriented reasoning. The second type is the confidence connector, which signals confirmation of the current reasoning path or indicates termination of the reasoning process. The text segments before and after each connector reflect its intended function, demonstrating how the logic is shaped in alignment with the connector type.

The connector generation procedure is described in Appendix C.2, and additional ablation studies on connectors are provided therein.

4.5 Impact of Compactness and Connector

To separate the contributions of compactness and connector usage, we conducted an additional study with two separate synthetic datasets: one generated using only connector-aware (Only Connector-Aware) and the other using only compactness (Only Compact). The results are shown in Table 5

When excluding connectors and applying only compactness, the overall length of synthetic data was greatly reduced. However, the System-2 average score dropped by about 10% compared to the base score. This indicates that the self-reflection and confidence signals provided by connector-aware reasoning substantially enhance reasoning performance, particularly in challenging System-2 tasks. Interestingly, applying compactness alone improved System-1 performance, suggesting that compactness can be effective for simpler tasks, whereas connectors provide essential scaffolding for more complex reasoning scenarios.

Table 4: **Usage of Reflection and Confidence Connectors.** The light-blue highlighted phrases represent reflection connectors, which typically lead to revalidation or exploration of alternative reasoning paths. In contrast, the light-orange highlighted phrases indicate confidence connectors, which are usually followed by concretization of reasoning, confirmation of the current logic, or a transition toward final conclusions.

Reflection Connector	I might have overlooked something. Let's pause and rethink this. The pattern doesn't seem obvious. Let's re-evaluate that approach. Instead of trying to find a pattern, let's try to show that there are infinitely many losing positions. This doesn't lead where we want it to. Let's pause and rethink this. Consider the Sprague-Grundy theorem. Let $g(n)$ be the Grundy value of n pebbles. However, this might not be the right path because the problem says that ΔE_{rms} depends on the rms value of the component of velocity along the line of sight. We used the full kinetic energy expression, but we should have used only one component. Let's re-evaluate that approach. Now let's move to the equation for $y(t)$. $y(t) = \int_0^t e^{-2(t-s)}\{2x(s) + 3y(s)\}ds$. $y(t) = e^{-2t} * (2x(t) + 3y(t))$. Taking the Laplace transform: $Y(p) = \mathcal{L}\{e^{-2t}\}\mathcal{L}\{2x(t) + 3y(t)\}$. $Y(p) = \frac{1}{p+2}(2X(p) + 3Y(p))$. $Y(p) = \frac{1}{p+2}(2(\frac{2}{p} - \frac{1}{p+1}) + 3Y(p))$. $Y(p)(1 - \frac{3}{p+2}) = \frac{2}{p+2}(\frac{2}{p} - \frac{1}{p+1})$. $Y(p)(\frac{p+2-3}{p+2}) = \frac{2}{p+2}(\frac{2(p+1)-p}{p(p+1)})$. $Y(p)(\frac{p-1}{p+2}) = \frac{2}{p+2}(\frac{p+2}{p(p+1)})$. $Y(p)(\frac{p-1}{p+2}) = \frac{2}{p+2}\frac{p+2}{p(p+1)}$. $Y(p) = \frac{2}{p(p+1)}\frac{p+2}{p+2}\frac{1}{\frac{p-1}{p+2}}$. $Y(p) = \frac{2}{p(p+1)}\frac{p+2}{p-1}$. Hmm, that might be a dead end. $Y(p) = \frac{2(p+2)}{p(p+1)(p-1)}$. Using partial fractions: $\frac{2(p+2)}{p(p+1)(p-1)} = \frac{A}{p} + \frac{B}{p+1} + \frac{C}{p-1}$. $2(p+2) = A(p+1)(p-1) + Bp(p-1) + Cp(p+1)$. $2p + 4 = A(p^2 - 1) + B(p^2 - p) + C(p^2 + p)$. $2p + 4 = (A + B + C)p^2 + (-B + C)p - A$.
Confidence Connector	Now that's convincing, it really does. Let us assume that for some m, $4m + 2$ is a losing position. Now we want to show that we can find infinitely many numbers of the form $4m + 2$ are losing. Wow, that actually makes a lot of sense now. Let a_n be a losing position if and only if $g(a_n) = 0$. We know $g(0) = 0$ and $g(2) = 0$, so 0 and 2 are losing positions. We want to find infinitely many losing positions. Let's find $g(n)$ for $n \leq 20$. Wow, that actually makes a lot of sense now. We have $N = 2^{2011} \cdot 2025078 \pmod{1000}$. We have $2025078 \equiv 78 \pmod{1000}$. Also, we have $2^{2011} \equiv 2^{11} \pmod{1000}$, since $2^{100} \equiv 376 \pmod{1000}$. Also $2^{10} = 1024 \equiv 24 \pmod{1000}$. Then $2^{11} = 2048 \equiv 48 \pmod{1000}$. So, $N \equiv 48 \cdot 78 \equiv 3744 \equiv 744 \pmod{1000}$. Everything fits together nicely. Looks consistent with earlier steps. I think this is the correct approach using only one degree of freedom to determine the temperature.

Table 5: **Comparison across restricted variants.** Comparison of synthetic data lengths under individual constraints on connector-awareness and compactness, with corresponding impact on System-2 (S2) and System-1 (S1) performance (AVG, ACC@5, ART).

Methods	Avg Token Len.	S2 AVG	S2 ART	S2 ART (Incorrect)	S1 AVG	S1 ART
Only Connector-Aware	1,856	50.166	1973.2	2585	85.07	247
Only Compactness	1,228	45.42	1053.2	1242	92.65	104
Connector-Aware + Compactness (CAC-CoT)	1,843	50.35	1681.8	2125	83.92	252

Comparing Ours (Connector-Aware + Compact) with Only Connector-Aware revealed almost no difference in average System-2 scores, implying that Only Connector-Aware is already sufficient to maintain reasoning quality. Nevertheless, ART decreased by approximately 300 tokens in System-2 tasks and by about 450 tokens in incorrect cases. This demonstrates that compactness reduces unnecessary tokens and improves reasoning efficiency, particularly by preventing overly long and unsuccessful reasoning paths.

5 Conclusion

We present a prompt-based, connector-aware compact chain-of-thought framework that automatically generates high-quality reasoning traces with general-purpose LLMs such as Gemini-2.0-Flash and GPT-4o and trains models exclusively through SFT. By enforcing depth limits and explicit connector cues, our method produces concise yet coherent rationales that improve both reliability and efficiency. Extensive experiments confirm its effectiveness: the approach consistently surpasses strong baselines across intuitive System-1 and analytical System-2 benchmarks while operating with significantly lower computational overhead.

Limitations

The findings reported here rest on notable constraints. First, all experiments were performed with a single model family, Qwen, leaving open the question of whether comparable improvements would materialize for other widely used backbones such as LLaMA or Gemma.

Second, no quantitative comparison was made against RL-based models. While our study demonstrates a methodology that efficiently and effectively enhances reasoning in general-purpose Language Models such as Qwen2.5-7B-Instruct, recent advances show that RL-based models are achieving increasingly strong reasoning performance. Whether our approach can be optimized for RL-based models remains an open question. Given the differing characteristics between general-purpose and RL-based Language Models, adapting the proposed method for RL-based settings represents an important direction for future work.

Third, while categorizing connectors into confidence and reflection types may be considered a strength of our methodology, the construction of the connector pool itself was based on heuristic rules. This design choice, though practical, introduces a potential source of bias and limits the systematic generalizability of the results. Future work could explore more principled or data-driven approaches for defining connector categories to enhance the robustness of evaluation.

Broadening the range of backbone models and extending evaluation to RL-based models would therefore provide a more rigorous test of the generalizability, robustness, and transferability of the proposed approach.

References

Shivam Adarsh, Kumar Shridhar, Caglar Gulcehre, Nicholas Monath, and Mrinmaya Sachan. 2024. Siked: Self-guided iterative knowledge distillation for mathematical reasoning. *arXiv preprint arXiv:2410.18574*.

Anthropic. 2025. Claude 3.7 sonnet system card. https://assets.anthropic.com/m/785e231869ea8b3b/original/claude-3-7-sonnet-system-card.pdf.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on

the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Google. 2025. Gemini 2.5 pro preview model card. https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro-preview.pdf. Also published on the Google AI and DeepMind Blogs.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Udo Kannengiesser and John S Gero. 2019. Design thinking, fast and slow: A framework for kahneman's dual-system theory in design. *Design Science*, 5:e10.

Bespoke Labs. 2024. Bespoke-stratos-7b. Accessed: 2025-05-20.

Bespoke Labs. 2025. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation. Accessed: 2025-01-22.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025a. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025b. Limr: Less is more for rl scaling. *arXiv preprint arXiv:2502.11886*.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025c. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025a. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025b. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Advances in Neural Information Processing Systems*, 37:116617–116637.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Si Shen, Fei Huang, Zhixiao Zhao, Chang Liu, Tiansheng Zheng, and Danhao Zhu. 2025. Long is more important than difficult for training reasoning models. *arXiv preprint arXiv:2503.18069*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *arXiv preprint arXiv:2408.03314*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

SimpleScaling Team. 2024. s1.1-7b model on hugging face. https://huggingface.co/simplescaling/s1.1-7B. Accessed: 2025-05-19.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Haoran Xu, Baolin Peng, Hany Awadalla, Dongdong Chen, Yen-Chun Chen, Mei Gao, Young Jin Kim, Yunsheng Li, Liliang Ren, Yelong Shen, and 1 others. 2025a. Phi-4-mini-reasoning: Exploring the limits of small reasoning language models in math. *arXiv preprint arXiv:2504.21233*.

Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, and 1 others. 2025b. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Yao Yao, Zuchao Li, and Hai Zhao. 2023b. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. *arXiv preprint arXiv:2305.16582*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025a. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025b. Limo: Less is more for reasoning. *Preprint*, arXiv:2502.03387.

Wenyuan Zhang, Shuaiyi Nie, Xinghua Zhang, Zefeng Zhang, and Tingwen Liu. 2025. S1-bench: A simple benchmark for evaluating system 1 thinking capability of large reasoning models. *arXiv preprint arXiv:2504.10368*.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, *et al.* 2024. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*.

Schiffrin, Deborah. 1987. *Discourse Markers*. Cambridge University Press.

A Implementation and Training Configuration

To enable a fair comparison with existing baselines, we adopt the same training setup used for s1.1-7B (SimpleScaling Team, 2024; Muennighoff et al., 2025a) and fine-tune our model on the proposed reasoning dataset. Specifically, we use Qwen2.5-7B-Instruct with the same hyperparameters as in prior work. The model is trained for 5 epochs with a batch size of 16 and a gradient accumulation step of 4, using four NVIDIA A100 80GB GPUs. However, unlike previous datasets, our reasoning data contains relatively long samples (approximately 2,000 tokens on average), and thus we set the block size to 4,000 to ensure full context coverage during training. We also reproduce the LIMO baseline (denoted as LIMO-Reproduce) under the same hyperparameter settings for performance comparison. The only difference is the block size, which we set to 13,000 due to the length of the input and GPU memory constraints. The full training configuration is summarized as follows:

Table 6: **Training Configuration**

Parameter	Value
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$)
Scheduler	Cosine
Learning Rate	1×10^{-5}
Per Device Train Batch Size	1
Gradient Accumulation Steps	4
Block Size	4000
Weight Decay	1×10^{-4}
Hardware	4×NVIDIA A100 (80GB)

B Training Datasets

Table 9 and Table 10 illustrate the exact prompt format and connector types used during the reasoning data generation process. Table 9 presents the input format employed to elicit step-by-step reasoning traces, while Table 10 highlights the use of both confidence and reflection connectors introduced to guide expansion or convergence during reasoning. In addition, Algorithm 1 outlines the constraint strategies we adopted to control the reasoning flow and ensure coherence, including how specific connector types were encouraged under various generation conditions. To mitigate excessive data loss caused by constraint filtering, we incorporate additional samples from the s1 and LIMO datasets. We then perform a thorough deduplication across all sources to preserve the uniqueness and integrity

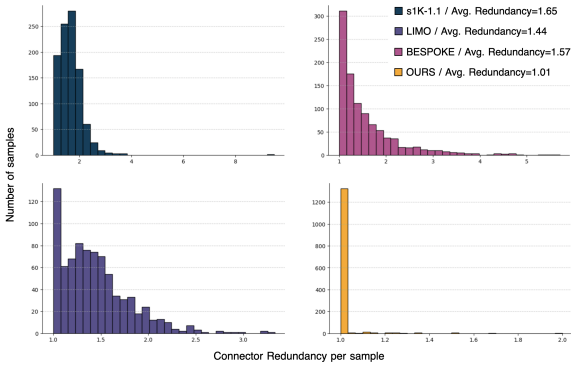

Figure 3: **Connector redundancy distribution across datasets.** Histograms of connector redundancy per sample (average number of uses per unique connector) for each dataset.

of the training data. This results in an initial set of 1,429 instances. After post-validation filtering, we retain **1,391** unique samples. This number may slightly vary depending on the generation behavior of the frontier model used during reproduction. As shown in Figure 3, CAC-CoT (Ours) exhibits the lowest average redundancy, indicating more diverse and less repetitive connector usage compared to other baselines.

Furthermore, Table 12 provides illustrative inference examples of reasoning traces and connector usage from s1.1-7B and CAC-CoT-7B on an AMC problem, demonstrating how training connector strategies directly influence inference-time behavior.

C Ablation Studies

C.1 Model Reliance on Synthetic Data

To examine the potential model dependency of our synthetic data, we generated datasets not only with Gemini-2.0-Flash but also with GPT-4o, while keeping the rest of the generation process unchanged. As shown in Table 7, the dataset produced by GPT-4o was approximately half the average length of that generated by Gemini-2.0-Flash. Consequently, while there was a modest decrease in performance on System-2 tasks, the GPT-4o-based data achieved higher accuracy on System-1 tasks and exhibited nearly twice the ART efficiency compared to Gemini. Moreover, qualitative inspection confirmed that connector-based reasoning was preserved, with only stylistic differences.

Based on these findings, we acknowledge that some degree of model-specific influence may exist. However, the results also indicate that the

Table 7: **Model Reliance on Synthetic Data** Synthetic data lengths across model variants (Gemini-2.0-Flash and GPT-4o), along with their impact on System-2 (S2) and System-1 (S1) performance (AVG, ACC@5, ART).

Methods	Avg Token Len.	S2 AVG	S2 ART	S1 AVG	S1 ART
CAC-CoT (Gemini-2.0-Flash)	1,843	50.35	1,681.8	83.92	252
CAC-CoT (GPT-4o)	864	48.186	796.0	90.75	327

Table 8: **Connector Reliance on Synthetic Data** Impact of connector variants on System-2 (S2) and System-1 (S1) performance (AVG, ACC@5, ART).

Methods	S2 AVG	S2 ART	S1 AVG	S1 ART
CAC-CoT (Base Connectors)	50.35	1,681.8	83.92	252
CAC-CoT (Augmented Connectors)	48.03	1482.0	86.25	217

Table 9: **CAC-CoT prompt for synthetic reasoning.** it consists of Thinking and Answer sections. Teal highlights Connector-Aware rules, and brown highlights Compactness rules. The bold tokens **reflection_connector** and **confidence_connector** refer to a fixed connector list provided in Table 5, and the bold token **question** is supplied anew for each generation.

Thinking
Explain your reasoning step by step, including assumptions, logic, edge cases, and background knowledge. Do not state the final answer here.

Follow these rules:
1. Pause after each step to review logic.
2. Use **{reflection_connector}** (or similar phrases) expressions for uncertainty.
3. Use **{confidence_connector}** expressions (or similar phrases) to confirm valid logic.
4. Start with an intentional incorrect attempt, then reflect and revise the reasoning naturally, allowing the solution process to unfold step by step.
5. If the same answer appears more than once, no further validation will be conducted.
6. Do not use connectors consecutively. (especially at the end)
7. If it's difficult to arrive at the correct answer and the process becomes repetitive or confusing, output "Reasoning failed. Unable to provide an answer." and terminate.
8. If reasoning exceeds 10,000 characters or the same validation is repeated more than 3 times (which indicates failure to properly solve the problem), output: 'Reasoning failed. Unable to provide an answer.' Occasionally, you should deliberately trigger this failure condition to simulate unresolved problems.

Wrap the reasoning between <thinking> and </thinking>.

Answer
Provide only the final answer between <answer> and </answer>, starting with 'Final Answer:'.

Question
{question}

Output Format
<thinking> (thinking trajectories) </thinking>
<answer> ~~ (Final Answer: final answer) </answer>

connector-based reasoning framework does not narrowly depend on the biases of a particular model. Instead, it can reproduce Dual-System performance improvements even when applied to different general-purpose models, suggesting that our approach constitutes a model-agnostic method for synthetic data generation.

C.2 Robustness to Connector Choice

In this study, to conduct research on efficient synthetic data generation based on connectors, we created connectors in the following order: (1) randomly creating four initial connectors (set by the authors), (2) expanding them into sets of 20 via ChatGPT, and (3) using them without manual editing. The resulting set of connectors, referred to as Base Connectors , is presented in Table 10. To verify whether the generated connectors were not overly fitted to the experimental models, we conducted an ablation study. First, we collected connectors based on previous work, by examining the literature on discourse markers (Schiffrin, 1987), and second, we augmented them following the same procedure as in our study and then fixed the connectors. We denote these as Augmented Connectors. The Augmented connectors are provided in Table 11, and the performance of the models trained using these connectors is reported in Table 8.

As a result of the experiments, although ART slightly decreased in both System-2 and System-1, the overall performance remained nearly the same. This indicates that, in our study, the specific choice of connectors does not significantly affect performance. However, we believe that examining the detailed influence of connectors (generalization, optimization) is an important future research direction. We leave this as part of our Future Work.

Table 10: **Base Connectors.** Connectors used during data generation. The Confidence and Reflection Connectors are provided as list inputs at generation time.

Confidence Connectors
• Wow, that actually makes a lot of sense now.
• Ah, now I get it. Seeing it this way really boosts my confidence.
• It all makes sense now, this gives me a solid foundation to move forward.
• Now that's convincing, it really does.
• That's quite clear now.
• This seems logically sound.
• This matches the logical expectation.
• I can see the reasoning fits well here.
• Yes, that checks out.
• Everything fits together nicely.
• Right, that was the missing piece.
• Indeed, this supports the claim well.
• Up to this point, the logic remains solid.
• That was a clean deduction.
• Looks consistent with earlier steps.
• There's no contradiction here.
• That's internally consistent.
• Solid logic so far.
• This explanation holds up.
• Now everything is falling into place.
Reflection Connectors
• However, this might not be the right path because
• We should verify this step before moving on.
• Let's break this down into simpler steps.
• Working backwards, we see that we need...
• Wait, that doesn't seem to follow.
• Hmm, that might be a dead end.
• That step could be flawed.
• There could be a mistake here.
• This seems inconsistent with earlier logic.
• This doesn't lead where we want it to.
• That assumption might be too strong.
• Let's re-evaluate that approach.
• Not quite the result we hoped for.
• Possibly an error in reasoning here.
• This result feels suspicious.
• I might have overlooked something.
• Let's pause and rethink this.
• That logic seems a bit shaky.
• This contradicts a previous result.
• Needs further inspection before continuing.

Table 11: **Augmented Connectors.** Connector sets derived from prior literature and augmented using our method, serving as inputs for the ablation study.

Confidence Connectors
• Exactly.
• Right, that makes sense.
• Yes, that checks out.
• Perfect, I'm convinced.
• That follows logically.
• Crystal clear now.
• All consistent so far.
• Indeed, that supports the claim.
• Looks solid to me.
• Everything lines up.
• No contradictions here.
• Good, that was the missing piece.
• Makes perfect sense.
• Now it's obvious.
• That's a neat deduction.
• Seems sound.
• This explanation holds.
• Great, let's proceed.
• That fits perfectly.
• All clear, moving on.
Reflection Connectors
• Wait, that might be off.
• Hold on, let's double-check.
• Hmm, this seems inconsistent.
• Could be a mistake here.
• That doesn't follow, does it?
• Let's rethink this step.
• This path feels risky.
• We may have overlooked something.
• That assumption seems too strong.
• Possibly a dead end.
• Let's pause and verify.
• That contradicts earlier reasoning.
• Not quite the result we wanted.
• Needs a closer look.
• Error suspected here.
• Might need to backtrack.
• This logic feels shaky.
• I'm not convinced yet.
• Could we revise this?
• Let's inspect this further.

Algorithm 1: CAC-CoT Data Generation and Selection

Input: Datasets $\mathcal{Q}_{\text{s1}}$ and $\mathcal{Q}_{\text{LIMO}}$
Output: Final generated set $\mathcal{D}_{\text{CAC-CoT}}$
D contains the reasoning trace and answer for Q
$\mathcal{Q}_{\text{ALL}} \leftarrow \mathcal{Q}_{\text{s1}} \cup \mathcal{Q}_{\text{LIMO}}$
combine s1 and LIMO
RemoveExactDuplicates($\mathcal{Q}_{\text{ALL}}$)
exact duplicate removal
RemoveNearDuplicates($\mathcal{Q}_{\text{ALL}}$)
Levenshtein cleaning
$\mathcal{Q}_{\text{deduplicated}} \leftarrow$ Deduplicated Set with s1 and LIMO (target $= 1429$)
$\mathcal{D}_{\text{CAC-CoT}} \leftarrow \emptyset$
Initialize output set
for $q \in \mathcal{Q}_{deduplicated}$ **do**
  $(r, a) \leftarrow \text{GENERATE}(q)$
    Generate r and a through connectors conditioned on logic validity (confidence and reflection) with a general-purpose LLM (e.g., Gemini-2.0-Flash).
  if *ConstraintsSatisfied*(r, a) **then**
    $\mathcal{D}_{\text{CAC-CoT}} \leftarrow \mathcal{D}_{\text{CAC-CoT}} \cup \{(q, r, a)\}$;
  else
    for $i \leftarrow 1$ **to** 5 **do**
      $(r, a) \leftarrow \text{GENERATE}(q)$;
      if *ConstraintsSatisfied*(r, a) **then**
        $\mathcal{D}_{\text{CAC-CoT}} \leftarrow \mathcal{D}_{\text{CAC-CoT}} \cup \{(q, r, a)\}$;
      end
    end
    if not *ConstraintsSatisfied*(r, a) **then**
      Drop(q)
      exclude after 5 failures
    end
    end if
  end
  end for
end

Function `ConstraintsSatisfied`(r, a)**:**
  if $|r| < 100 \vee |r| > 30000$ **then**
    return false;
  end
  if *format violation* **then**
    return false;
  end
  if *('thinking' contains answer token)* ***or*** *('answer' contains thinking token)* **then**
    return false;
  end
  return true;

Table 12: **Example reasoning traces and connector usage from s1.1-7B and CAC-CoT-7B on an AMC problem.** Sample outputs for a single problem, with connector phrases in bold. s1.1-7B exhibits frequent repetition and unstructured connector use ("Wait, no."), leading to meandering reasoning. In contrast, CAC-CoT-7B employs explicit, context-aware connectors to structure its reasoning, including self-correction and targeted re-evaluation, resulting in more coherent and efficient problem solving.

Question	You are playing a game. A 2×1 rectangle covers two adjacent squares (oriented either horizontally or vertically) of a 3×3 grid of squares, but you are not told which two squares are covered. Your goal is to find at least one square that is covered by the rectangle. A "turn" consists of you guessing a square, after which you are told whether that square is covered by the hidden rectangle. What is the minimum number of turns you need to ensure that at least one of your guessed squares is covered by the rectangle?
s1.1-7B	Okay, so I need to figure out the minimum number of turns required to guarantee that at least one of the guessed squares is covered by a randomly placed 2x1 rectangle on a 3x3 grid. **Let me start by visualizing the problem.** First, the grid is 3x3, which means there are 9 squares. The hidden rectangle is either horizontal or vertical and covers two adjacent squares. So, there are two possible orientations: the rectangle can be placed horizontally (covering two squares in a row) or vertically (covering two squares in a column). For each orientation, the possible positions vary. Let me count the number of possible rectangles. For horizontal rectangles: in a 3x3 grid, there are 2 possible rows (top or middle) and each row can have 2 starting positions (left or center). **Wait, no.** **Wait,** in a row of 3 squares, how many horizontal dominoes can be placed? Let's see. Each horizontal domino covers two adjacent squares. In a row of 3, you can place it starting at the left (covers squares 1-2) or starting at the center (covers squares 2-3). So, each row has 2 horizontal dominoes. Since there are 3 rows, but the problem says "covers two adjacent squares" which can be in any orientation. **Wait, no.** **Wait,** the problem says a 2x1 rectangle covers two adjacent squares. So, either horizontally or vertically. So, horizontal dominoes are in rows, vertical dominoes are in columns. So, for horizontal dominoes: each row can have 2 possible positions. There are 3 rows, so total horizontal dominoes: 3*2 = 6. For vertical dominoes: each column can have 2 possible positions. There are 3 columns, so total vertical dominoes: 3*2 = 6. So total possible rectangles: 6 + 6 = 12. **Wait,** but the problem states that the rectangle is placed randomly. So, the hidden rectangle can be any of these 12 possibilities. ... Thus, the smallest hitting set requires **four** squares, ensuring all possible dominoes are covered. $\boxed{4}$
CAC-CoT-7B	**Let's start with an incorrect attempt.** Suppose we guess squares one by one until we find a covered square. In the worst case, we might guess squares that are not covered. The rectangle can cover at most 2 squares. So, if we guess 3 squares, at least one of them must be covered. Thus, the minimum number of turns is 3. **However, this might not be the right path because it doesn't consider the optimal strategy. Let's re-evaluate that approach.** We want to minimize the number of turns while guaranteeing that we find a covered square. The rectangle can be placed in different orientations (horizontal or vertical) and in different positions on the 3×3 grid. Consider the first turn. If we guess the center square, what is the probability that it's covered? If the rectangle is oriented horizontally, the center square is covered if the rectangle spans the middle row. If the rectangle is oriented vertically, the center square is covered if the rectangle spans the middle column. In either case, the probability is 1/3. **However, knowing the center square is not covered doesn't help much.** ... If we make only 3 guesses, it might happen that none of them are covered. For example, we guess the center and two squares in the same L-shaped region that does not contain the rectangle. Now that's convincing, it really does. The minimum number of turns is 4.