MUX: CONTINUOUS REASONING VIA MULTIPLEXED TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models solve complex problems by articulating intermediate reasoning steps in natural language. While effective, this process is computationally bottlenecked: each reasoning step conveys only a single subword, and many steps are spent expressing a thought rather than carrying out computation. We propose MUX, a simple method for high-bandwidth and compact reasoning based on distillation of discrete reasoning into continuous multiplexed tokens. Each continuous token is trained to represent a weighted linear superposition (multiplexing) of a span of discrete reasoning steps, while ensuring that this superposition is lossless and the span can be fully recovered (demultiplexing). We prove that simple position-dependent weightings, such as properly chosen geometric decay, support lossless multiplexing, and further prove that multiplexed reasoning can perform parallel exploration in problems that require search. Across 16 evaluation settings spanning two language models, MUX is competitive with or outperforms strong continuous reasoning baselines, especially when the base discrete reasoning is verbose. Overall, our results suggest that appropriately chosen learning targets can make continuous reasoning both efficient and interpretable.

1 INTRODUCTION

Modern language models are capable of solving complex problems in domains such as mathematics, coding, and commonsense reasoning through their *reasoning* mechanism. In autoregressive language models, this mechanism typically involves verbalizing intermediate solution steps in natural language before producing the final answer (Nye et al., 2021; Wei et al., 2022). However, this mode of operation imposes a strict constraint on the computational bandwidth since each reasoning step transmits only a single subword. Moreover, many of these steps are redundant (Xia et al., 2025; Li et al., 2025), since the model mirrors problem-solving patterns learned from human-generated corpora, which are inherently optimized for communication rather than computation. These limitations motivate the development of approaches that enable higher-bandwidth and more compact reasoning in language models.

Reasoning in continuous latent spaces has emerged as an alternative paradigm, where a language model autoregresses continuous vectors instead of subwords before producing an answer (Hao et al., 2024; Xu et al., 2025b). These latent reasoning approaches have a higher bandwidth, since each step can convey multiple subwords simultaneously by encoding them *in superposition*. This enables the exploration of different problem-solving paths in parallel, which leads to improvements in planning and search tasks (Zhu et al., 2025; Gozeten et al., 2025). Despite their potential, continuous reasoning methods have not yet been widely adopted, in part because they are notoriously hard to learn. One class of methods relies on temporal backpropagation of *trajectory-level losses* (Hao et al., 2024; Shen et al., 2025), which tend to result in uninformative reasoning as the number of continuous tokens grows (Wei et al., 2025). Other approaches define *local distillation losses* for each continuous token based on discrete reasoning traces (Wei et al., 2025; Kuzina et al., 2025), but often by introducing technical complexity such as autoregressive decoders or key-value cache.

In this work, we introduce **MUX**, a simple and novel method for learning high-bandwidth, compact continuous reasoning in language models. Our approach is motivated by the observation that, under local distillation, each continuous token can represent an aligned local span of discrete reasoning. For accurate reasoning, this representation should be effectively lossless, meaning the original span

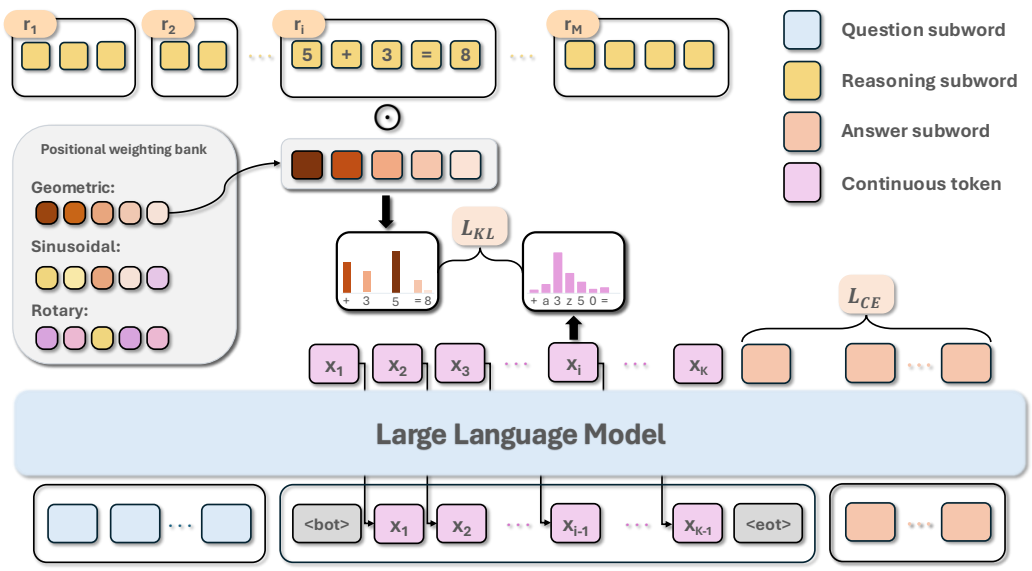

Figure 1: Given a question, the language model predicts a sequence of continuous reasoning tokens autoregressively. Each continuous token is linearly projected to the vocabulary space and trained to represent a local span of discrete reasoning steps. We construct a vocabulary-space learning target from each discrete span as a weighted average of one-hot encodings, and train each continuous token with KL divergence loss. The answer subwords are trained with standard cross-entropy loss.

can be recovered from it. Drawing inspiration from telecommunications systems, we implement this idea using a *multiplexed* token-level target (Figure 1). For each aligned span of discrete reasoning, we construct a position-weighted linear superposition of its one-hot subword encodings and normalize the result onto the vocabulary simplex. The model is trained to predict this target using a linear-softmax head by minimizing a Kullback–Leibler (KL) loss. This formulation provides a fixed-dimensional supervision signal in vocabulary space for each continuous token, without requiring an auxiliary autoregressive decoder or a compressed cache target.

MUX provides a simple yet highly effective approach to continuous reasoning in language models, combining strong theoretical grounding with practical performance gains. We summarize our main contributions as follows:

- **Theoretical analysis.** We show that multiplexing is *lossless* under suitable positional weightings, including appropriately designed geometric, sinusoidal, and rotary schemes, enabling exact recovery of the underlying discrete reasoning span (*demultiplexing*) (Theorems 2, 4, 6). We further demonstrate that multiplexed representations can support parallel search (Theorem 9).
- **Empirical evaluation.** We evaluate MUX across 16 settings covering two base language models (GPT-2 Small and LLaMA 3.2 1B-Instruct), two training corpora, and four test corpora spanning both in-domain and out-of-domain tasks. MUX achieves state-of-the-art performance among continuous reasoning methods in 13 of 16 settings. Notably, MUX outperforms the discrete reasoning baseline when considering in-domain evaluations with GPT-2, revealing a regime in which continuous reasoning surpasses the discrete reasoning it was distilled from (Table 1).
- **Ablation studies and interpretability.** We find that accuracy scales with the number of supervised continuous tokens (from 32.7% to 48.2%), and that these gains stem from multiplexed local supervision rather than from additional latent computation alone (Figure 3). Moreover, decoded continuous tokens are span-aligned and interpretable, a direct consequence of supervising in vocabulary space (Figure 5).

Overall, our results support the hypothesis that lossless superposition as a local distillation target constitutes a sufficient condition for achieving strong and efficient continuous reasoning. All proofs of the technical statements can be found in Appendix A.2.

2 Related Work

Reasoning in language models. A large body of work has shown that language models benefit from making intermediate computations explicit. Early scratchpad methods (Nye et al., 2021) demonstrated that learning intermediate steps of multi-step computations can substantially improve final accuracy. Chain-of-thought (CoT) prompting (Wei et al., 2022) established natural language reasoning as a general mechanism for improving arithmetic, logical, and commonsense reasoning. Recent work has identified inefficiency of natural language reasoning, showing that many reasoning tokens can be pruned (Li et al., 2025) or compressed (Xia et al., 2025) with small degradation in accuracy. We share this motivation, but instead of shortening discrete reasoning at inference time, we study how to distill discrete reasoning traces into compact continuous reasoning through training.

Reasoning in continuous latent spaces. A growing line of work focuses on reasoning in continuous latent spaces instead of in language. iCoT (Deng et al., 2023; 2024) proposed distilling discrete reasoning into a single forward pass. Subsequent works investigated distillation for autoregressive continuous reasoning, which can be categorized into global and local methods. Global methods supervise the answer token or trajectory endpoint, and learn continuous autoregression via temporal backpropagation, such as in Coconut (Hao et al., 2024) and CODI (Shen et al., 2025). In a complementary manner, local methods supervise each continuous reasoning token to represent a local span of discrete steps by aligning them in a choice of representation space. For this, prior methods often use auxiliary modules. SIM-CoT (Wei et al., 2025) aligns in autoregressively decoded text space, and KaVa (Kuzina et al., 2025) aligns in a compressed key-value cache space. We question the necessity of such technical complexity by rethinking the core requirements of local distillation. We use linearly superposed representations in the vocabulary space, in line with Zhang et al. (2025b) and Tang et al. (2026). Unlike these methods, which autoregress vocabulary-space vectors, we use vocabulary-space projection only during training and perform autoregressive reasoning directly in latent space. This shift isolates vocabulary-space supervision as a training signal rather than an inference-time computation, yielding a simpler formulation that preserves the benefits of local distillation without requiring multi-token branching or embedding aggregation during decoding. Lastly, theoretical works studied advantages of continuous reasoning, especially superposition and parallel search (Gozeten et al., 2025; Zhu et al., 2025; Wu et al., 2025a). We prove that multiplexed reasoning preserves these core capabilities.

3 MUX: Continuous Reasoning via Multiplexed Tokens

3.1 Problem setup

Language models and reasoning. Let V be a discrete vocabulary of subwords and let V^* be its corresponding text space. We denote a text with length L by $\mathbf{y} = (\mathbf{y}^1, ..., \mathbf{y}^L)$ with each $\mathbf{y}^l \in V$. Language models generate continuations of a given text $\mathbf{y}^{\leq L} \in V^*$ by autoregression, or predicting the next subword $\mathbf{y}^{L+1}$ and then repeating the procedure on $\mathbf{y}^{\leq L+1}$. We are interested in problems of predicting an answer $\mathbf{a} \in V^*$ to a given question $\mathbf{q} \in V^*$. While a language model may directly answer $\mathbf{q} \mapsto \hat{\mathbf{a}}$ by continuation, prompting it to produce an intermediate reasoning $\mathbf{r} \in V^*$ before answering $(\mathbf{q}, \mathbf{r}) \mapsto \hat{\mathbf{a}}$ improves its performance. This is, however, computationally inefficient.

Continuous reasoning. To overcome the bottlenecks of discrete reasoning, we reason in a continuous latent space. We choose a vector space X and denote by X^* the corresponding set of vector sequences. For each question $\mathbf{q}$, we would like to train a language model to articulate a continuous reasoning $\mathbf{x} \in X^*$ by autoregressing on the continuous tokens $\mathbf{x}_1, \mathbf{x}_2, ... \in X$ before answering $(\mathbf{q}, \mathbf{x}) \mapsto \hat{\mathbf{a}}$. In practice, we choose $X = \mathbb{R}^d$ and treat X^* as $X^K \in \mathbb{R}^{K \times d}$ for a choice of K, following the settings of prior work. This restricts each reasoning to a sequence of K vectors.

Learning to reason in a continuous space is challenging. A naïve approach is to let a language model generate a reasoning $\mathbf{q} \mapsto \mathbf{x}$ and then answer $(\mathbf{q}, \mathbf{x}) \mapsto \hat{\mathbf{a}}$, and learn by temporal backpropagation upon $\mathsf{error}(\hat{\mathbf{a}}, \mathbf{a})$. However, this empirically leads to collapsed, uninformative reasoning. Following prior work, we take a distillation approach, assuming availability of a dataset of triples $(\mathbf{q}, \mathbf{r}, \mathbf{a})$ of question $\mathbf{q}$, discrete reasoning trace $\mathbf{r}$, and answer $\mathbf{a}$, using them to learn continuous reasoning $\mathbf{x}$.

We focus on *local* distillation, where continuous tokens are supervised with local spans of discrete reasoning steps. We assume a discrete reasoning trace $\mathbf{r}$ is chunked into disjoint spans $(\mathbf{r}_1, ..., \mathbf{r}_M)$.

Here, each span $\mathbf{r}_i$ can be a sentence in natural language tasks, or a step of arithmetic computation in mathematical tasks. If a trace has more spans than continuous tokens, $M > K$, we merge some of the spans using a deterministic or randomized heuristic (see Appendix A.1 for details). With this, we assume $M \leq K$ from this point onward. The reasoning $\mathbf{x}$ is then trained such that each continuous token $\mathbf{x}_i \in \mathbb{R}^d$ represents a span $\mathbf{r}_i \in V^*$ in some choice of representation space Z. This objective can be formalized as $f(\mathbf{x}_i) = g(\mathbf{r}_i)$ for some choice of maps $f : \mathbb{R}^d \to Z$ and $g : V^* \to Z$. For example, Wei et al. (2025) takes $Z = V^*$ and introduces an autoregressive $f : \mathbb{R}^d \to V^*$ and identity $g = \mathrm{id}$, and Kuzina et al. (2025) takes Z as the space of compressed key-value cache and performs cache distillation. We question the necessity of such choices that add technical complexity.

3.2 Local distillation by multiplexing

We now present our method for continuous reasoning via local distillation. To motivate our method, we consider the case where Z is a fixed-dimensional vector space. Then, $g : V^* \to Z$ can be viewed as an operator that combines a variable-dimensional signal $i \mapsto \mathbf{r}_i$ into one, fixed-dimensional signal $i \mapsto g(\mathbf{r}_i)$ which defines the learning target of continuous reasoning $i \mapsto \mathbf{x}_i$ via $f(\mathbf{x}_i) = g(\mathbf{r}_i)$. Given this observation, it is natural to conceptualize g as a type of *multiplexing* operator.

Definition 1. We say that continuous reasoning $(\mathbf{x}_1, ..., \mathbf{x}_K)$ under a decoder $f : \mathbb{R}^d \to Z$ multiplexes discrete reasoning $(\mathbf{r}_1, ..., \mathbf{r}_M)$ if there exists an injective map $\mathsf{mux} : V^* \to Z$ satisfying

$$f(\mathbf{x}_i) = \mathsf{mux}(\mathbf{r}_i), \quad \forall i \leq M. \tag{1}$$

The definition characterizes core requirements for learning continuous reasoning through local distillation. Equation 1 requires that each continuous token represents a local span of a discrete reasoning trace through multiplexing. Injectivity of the multiplexer requires that the representation is lossless, admitting an inverse (demultiplexing). This ensures that the learned continuous reasoning can perform faithful problem solving. Given a valid choice of mux, we train a language model to perform continuous reasoning $\mathbf{q} \mapsto \mathbf{x}$, possibly jointly with the decoder f, with position-wise regression loss $\mathsf{error}(f(\mathbf{x}_i), \mathsf{mux}(\mathbf{r}_i))$ to directly satisfy equation 1. In practice, it is helpful to optionally use trajectory-level loss terms on the answer, $\mathsf{error}(\hat{\mathbf{a}}, \mathbf{a})$, and hidden states used to produce the answer, as in Shen et al. (2025); Wei et al. (2025). This is useful in learning the "spare" tokens $\mathbf{x}_{M+1}, ..., \mathbf{x}_K$ when $M < K$, as local distillation loss does not provide direct learning signal for these tokens.

Multiplexing via linear superposition. Constructing a lossless multiplexer is a nontrivial problem, as it needs to handle variable dimensionality of spans. Here, we propose a class of simple and training-free multiplexers based on linear superposition. For each discrete reasoning span $\mathbf{r}_i \in V^*$ with subwords $\mathbf{r}_i^1, ..., \mathbf{r}_i^S \in V$, we define our multiplexer $\mathsf{mux} : V^* \to \Delta^{|V|-1}$ as follows:

$$\mathsf{mux}(\mathbf{r}_i) = \sum_{j=1}^{S} \alpha_j \cdot \mathsf{onehot}(\mathbf{r}_i^j), \qquad \alpha_j = \frac{w_j}{\sum_{\ell=1}^{S} w_\ell}, \tag{2}$$

where $\Delta^{|V|-1}$ is the space of $|V|$-dimensional probability vectors, and $w_{(\cdot)} : \mathbb{N} \to \mathbb{R}_+$ is a positive weighting function, chosen as one of the following (Figure 2):

- **Geometric.** $w_j = \rho^{j-1}$ with decay rate $\rho \in (0, 1)$. Earlier positions receive exponentially more weight, producing a monotonically decaying profile.
- **Sinusoidal.** $w_j = \exp(\tau s_j)$ with scores $s_j = \sin(\frac{\pi}{2} \cdot \frac{j-1}{\max(S-1,1)})$ and temperature $\tau > 0$. The sine maps positions to $[0, 1]$, and the exponential converts these scores into a monotonically increasing weighting that peaks near the end of the span.
- **Rotary.** $w_j = \exp(\tau s_j)$ with scores $s_j = \frac{1}{P} \sum_{p=1}^{P} \cos(\theta_p(j-1))$, where $(\theta_p)_{p \leq P}$ is a bank of frequencies analogous to rotary position embeddings (Su et al., 2024). Averaging cosine components across frequencies yields an expressive positional weighting.

All the choices yield lossless multiplexing for proper choices of hyperparameters, as we prove in Section 4. This result is not trivial. The sum $\mathsf{mux}(\mathbf{r}_i)$ records only the *total* mass of each unique subword in a span, so if a subword appears more than once, its positions within the span can be ambiguous. For example, with uniform weighting $\alpha_{(\cdot)} = 1/S$, the mass of a subword only counts how many times it appears in a span, and drops the positions in which it appears.

The key to making multiplexing lossless is to choose positional weights $\alpha_1, ..., \alpha_S$ so that different sets of positions always produce different total masses. Our theory in Section 4 formalizes this as a subset-sum separation condition on the weights, and shows that our weighting families are lossless for proper hyperparameters. Under this condition, even if a token appears multiple times, the total mass assigned to it uniquely determines which positions it occupied, so the original span can be recovered exactly.

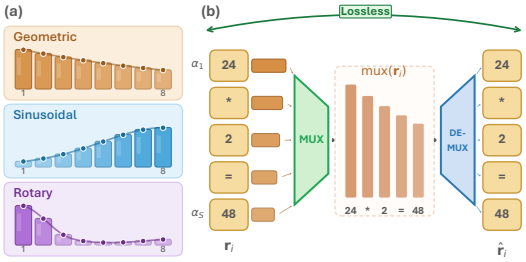

Figure 2: Multiplexing pipeline with three considered positional weightings.

In addition to being lossless, a convenient property of our multiplexer mux is that it always maps to the probability simplex $Z = \Delta^{|V|-1}$ since it computes a convex combination of one-hot vectors. This motivates our choice of decoder $f : \mathbb{R}^d \to \Delta^{|V|-1}$ as a trainable linear map followed by softmax, and also allows learning continuous reasoning $(\mathbf{x}_1, ..., \mathbf{x}_K)$ by minimizing KL divergence as the position-wise regression loss, $D_{\mathrm{KL}}(\mathrm{mux}(\mathbf{r}_i) \,\|\, f(\mathbf{x}_i))$. This gives local distillation in a direct form, without introducing an auxiliary decoder or compressed cache target as in prior work.

4 Theoretical Analysis

We establish two theoretical properties that underpin the effectiveness of multiplexed reasoning. First, we characterize when multiplexing is *lossless*, meaning the original discrete reasoning span can be exactly recovered from its multiplexed representation, and show that our proposed weighting schemes satisfy this condition. Second, we prove that multiplexed representations can support *parallel search*, enabling a single continuous token to simultaneously track multiple problem-solving paths. This formalizes a key advantage of continuous over discrete reasoning.

4.1 Lossless Multiplexing

Let us consider multiplexing a span of discrete reasoning $\mathbf{r}_i = (\mathbf{r}_i^1, ..., \mathbf{r}_i^S)$ into a continuous token, as $\mathsf{mux}(\mathbf{r}_i) = \sum_{j=1}^{S} \alpha_j \cdot \mathsf{onehot}(\mathbf{r}_i^j)$ for positive masses $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_S)$ that sum to one. Our goal is to identify the conditions on the masses that make the multiplexer lossless or injective (Definition 1). Let $\mathcal{C}$ denote the set of all nonzero sequences $\mathbf{c} = (c_1, ..., c_S)$ taking values in $\{-1, 0, 1\}$ and consider the following scalar function that quantifies the amount of subset-sum collisions:

$$E(\boldsymbol{\alpha}) = \min_{\mathbf{c} \in \mathcal{C}} \left| \sum_{j=1}^{S} c_j \alpha_j \right| \tag{3}$$

Intuitively, $E(\boldsymbol{\alpha}) > 0$ if and only if there are no distinct subsets of $\{1, ..., S\}$ having an identical total mass. We now characterize the exact criterion for lossless multiplexing as follows.

Theorem 2. *For $|V| > 1$ and a fixed span length S,* mux *is injective if and only if $E(\boldsymbol{\alpha}) > 0$.*

Theorem 2 is a demultiplexing statement, showing that a finite span of discrete reasoning can be recovered without loss from a weighted linear superposition if the subset sums of the weights never collide. Based on the one-span case, we now consider an extension to full reasoning. Let us consider multiplexing a discrete reasoning trace $\mathbf{r}$ chunked into spans $(\mathbf{r}_1, ..., \mathbf{r}_M)$. Since the spans can have different lengths, we denote the length of $\mathbf{r}_i$ by S_i, and denote the positional masses used for multiplexing $\mathbf{r}_i$ as $\boldsymbol{\alpha}^{(i)}$. We now characterize lossless multiplexing of a full reasoning trace.

Theorem 3. *If $E(\boldsymbol{\alpha}^{(i)}) > 0$ for all $i = 1, ..., M$, any discrete reasoning trace $\mathbf{r} = (\mathbf{r}_1, ..., \mathbf{r}_M)$ can be recovered from the multiplexed tokens* $\mathsf{mux}(\mathbf{r}_i)$ *together with span lengths S_i and masses $\boldsymbol{\alpha}^{(i)}$.*

The proof is an immediate application of Theorem 2 together with the fact that chunking $\mathbf{r}$ into spans $(\mathbf{r}_1, ..., \mathbf{r}_M)$ does not drop any information.

4.2 WEIGHTING FOR LOSSLESS MULTIPLEXING

Based on Theorem 2, we now identify the choices of hyperparameters for the weighting schemes used in our method that support lossless multiplexing. First, for geometric weighting $w_j = \rho^{j-1}$ with decay rate $\rho \in (0, 1)$, we identify an exact algebraic characterization as follows.

Theorem 4. *For geometric weighting, multiplexing is injective if and only if ρ is not a root of any nonzero polynomial $\sum_{j=1}^{S} c_j x^{j-1}$ with coefficients $c_j \in \{-1, 0, 1\}$.*

For each fixed S, this excludes only finitely many values of $\rho \in (0, 1)$. A useful consequence is the following, showing that rational geometric decays are lossless.

Corollary 5. *If $\rho \in (0, 1)$ is rational, then geometric weighting is injective for every finite S.*

The proof uses the rational root theorem. This result provides a wide range of freedom in choosing ρ in practice while retaining lossless multiplexing.

The remaining weightings, sinusoidal and rotary, take a common exponential form $w_j = \exp(\tau s_j)$, with scores $s_j = \sin(\frac{\pi}{2} \cdot \frac{j-1}{\max(S-1,1)})$ in the sinusoidal case and $s_j = \frac{1}{P} \sum_{p \le P} \cos(\theta_p(j-1))$ in the rotary case. The next theorem shows that pairwise distinct scores are enough to guarantee injectivity for all but finitely many values of τ.

Theorem 6. *Fix $S \ge 2$, and suppose $s_1, ..., s_S$ are pairwise distinct. Then $w_j = \exp(\tau s_j)$ yields an injective multiplexing for all but finitely many values of τ.*

The proof reduces the collision condition to the zeros of a finite family of exponential polynomials. This result immediately yields a corollary that sinusoidal weighting is generally lossless.

Corollary 7. *For every $S \ge 2$, then sinusoidal weighting yields an injective multiplexing for all but finitely many values of τ.*

For rotary weightings, we prove a simple sufficient condition for lossless multiplexing that all of its frequencies remain on the first decreasing branch of cosine.

Corollary 8. *If $\theta_p(S-1) < \pi$ for all p, then rotary weighting yields an injective multiplexing for all but finitely many values of τ.*

Together, these results show that sinusoidal and rotary weightings can also be chosen to achieve lossless multiplexing.

4.3 PARALLEL SEARCH WITH MULTIPLEXED REASONING

Sections 4.1 and 4.2 show that multiplexed reasoning can faithfully compress a discrete reasoning trace into fewer continuous tokens through lossless superposition. A distinct advantage of continuous reasoning, however, is the ability to represent multiple problem-solving paths *simultaneously*, a capability that benefits search and planning. In this section, we ask whether multiplexed reasoning preserves this advantage, and prove that it does: autoregressive reasoning over multiplexed continuous tokens can implement exact parallel breadth-first search on graphs, even when the discrete reasoning traces are presented as ordinary sequences of subwords.

Setup. Consider a problem that requires search over a finite directed graph $G = (\mathcal{N}, E)$ with source $s \in \mathcal{N}$, target $t \in \mathcal{N}$, and horizon H. The breadth-first frontier sequence is defined by $F_0 = \{s\}$, $U_0 = \{s\}$, and $F_{k+1} = N^+(F_k) \setminus U_k$, $U_{k+1} = U_k \cup F_{k+1}$ for $k = 0, ..., H-1$, where $N^+(F_k)$ is the out-neighborhood of F_k. Suppose we have access to a serialization of each frontier F_k as a span $\mathbf{r}_k = (\mathbf{r}_k^1, ..., \mathbf{r}_k^{|F_k|})$ with each node appearing exactly once. Here the objective is not to recover an arbitrary span, but to represent the current frontier as a distribution for parallel search, which makes the uniform weighting a natural choice. Under uniform weighting $\alpha_j = 1/|F_k|$, the multiplexed target becomes $\mathsf{mux}(\mathbf{r}_k) = \frac{1}{|F_k|} \sum_{j=1}^{|F_k|} \mathsf{onehot}(\mathbf{r}_k^j)$, which is the uniform distribution over the current frontier.

Theorem 9 (Parallel BFS with MUX)**.** *Under the setup above, there exists a latent recurrence over continuous tokens such that, for every $k \le H$, the token at step k exactly encodes (F_k, U_k). Consequently, the final answer is exact, $y = \mathbf{1}[t \in U_H]$, and for each nonempty frontier F_k, the*

frontier distribution $\mathrm{mux}(r_k)$ *is recoverable from that token; moreover, it can be approximated to arbitrary precision by a standard softmax readout.*

Theorem 9 shows that parallelism can emerge from serial reasoning data. The reasoning span $\mathbf{r}_k$ is just an ordinary token sequence enumerating the frontier nodes, but uniform multiplexing converts it into a distribution over the frontier, and a continuous token can propagate that distribution forward in parallel, with one BFS level per reasoning step. In this setting, continuous reasoning offers a fundamental advantage over discrete reasoning, which requires serializing the frontier expansion.

Corollary 10 (Finite breadth-first exploration problems)**.** *The same conclusion holds for any finite search problem whose solver proceeds by breadth-first frontier expansion with finite auxiliary memory. More precisely, let* $\mathcal{S}$ *be a finite state space and* $\mathcal{M}$ *a finite memory space, and suppose the search dynamics are* $F_{k+1} = \Psi(F_k, m_k)$, $m_{k+1} = \Omega(F_k, m_k)$, *with final answer* $y = \Gamma(m_H)$. *If the reasoning span* $\mathbf{r}_k$ *serializes the frontier* F_k *and uniform weighting is used, then there exists a continuous-token recurrence that tracks* (F_k, m_k) *exactly and whose tokenwise targets are the corresponding frontier distributions.*

5 Experiments

5.1 Experimental Setup

Training data. We train on two reasoning-augmented mathematical problem corpora built on top of GSM8K (Cobbe et al., 2021). The first is **GSM8K-Aug**, where the original GSM8K training set is expanded with GPT-4-generated examples and the intermediate discrete reasoning trace is represented primarily as structured mathematical expressions. The second is **GSM8K-Aug-NL**, which preserves more verbose natural-language explanations in the reasoning trace. For in-domain evaluation, we report results on the corresponding held-out test split used in prior work on latent and continuous reasoning (Deng et al., 2023; Shen et al., 2025).

Out-of-domain benchmarks. To evaluate robustness under distribution shift, we additionally test models on three out-of-domain arithmetic reasoning benchmarks. We use **SVAMP** (Patel et al., 2021), which perturbs elementary math word problems to reduce reliance on shallow lexical patterns; **GSM-Hard**, a harder GSM8K-style benchmark introduced in Gao et al. (2023); and **MultiArith** (Roy & Roth, 2015), a benchmark of multi-step arithmetic word problems. This evaluation suite follows recent work on latent and continuous reasoning that measures whether gains learned on augmented GSM8K transfer to related but distribution-shifted mathematical reasoning tasks.

Backbones and baselines. We consider two base language models (**GPT-2 Small** (Radford et al., 2019) and **LLaMA 3.2 1B-Instruct** (Meta, 2024)), which are common choices in prior work and fine-tune them with low-rank adaptation (LoRA) (Hu et al., 2022). We compare MUX against a spectrum of answer-only, discrete-reasoning, and continuous-reasoning baselines. **SFT-CoT** is supervised to generate discrete reasoning traces and final answers, while **No-CoT** predicts only the final answer. **iCoT** (Deng et al., 2024) is trained with stepwise internalization of discrete reasoning traces. **Coconut** (Hao et al., 2024) and **CODI** (Shen et al., 2025) distill discrete reasoning traces into continuous latent reasoning through trajectory-level supervision. **SIM-CoT** (Wei et al., 2025) performs local distillation from discrete to continuous reasoning based on an auxiliary autoregressive decoder, combined with global distillation of CODI. **SIM-CoT (w/o global)** removes global distillation and tests local distillation in isolation. We omit the **KaVa** (Kuzina et al., 2025) baseline from our experiments due to a mismatch in the number of continuous tokens (our setting uses 6, whereas theirs uses 24), and because their codebase is not publicly available. Our method is tested in two forms: **MUX**, which performs the proposed local distillation with global distillation of CODI, and **MUX (w/o global)**, which tests local distillation in isolation.

5.2 Main Results

Overview. Table 1 shows that our method is consistently competitive across both backbones, both training corpora, and all evaluation benchmarks. Given that SFT-CoT can be understood as an upper bound of distillation performance, MUX is the best continuous reasoning method in 13 out of 16 settings and the second-best in two more. The results indicate that the local distillation based on multiplexing transfers robustly across both in-domain and out-of-domain mathematical reasoning.

Method	GSM8K-Aug				GSM8K-Aug-NL			
	ID	SVAMP	GSM-Hard	MultiArith	ID	SVAMP	GSM-Hard	MultiArith
LLaMA 3.2 1B-Instruct								
SFT-CoT	61.6	66.7	15.6	99.3	53.2	62.9	13.3	–
No-CoT	30.9	44.1	7.1	70.9	30.9	44.1	7.1	70.9
iCoT	19.0	40.9	4.4	39.0	15.2	–	–	–
Coconut	45.3	48.8	9.9	90.1	24.2	–	–	–
CODI	55.6	61.1	<u>12.8</u>	96.1	<u>49.7</u>	–	–	–
SIM-CoT	<u>56.1</u>	**61.5**	12.7	96.2	28.4	43.0	6.6	59.4
SIM-CoT (w/o global)	31.6	44.0	7.5	69.5	29.0	45.1	7.0	58.3
MUX	**56.6**	<u>61.3</u>	**13.0**	<u>98.3</u>	**50.6**	**58.1**	**11.8**	**97.2**
MUX (w/o global)	48.7	50.9	10.8	**98.9**	39.2	<u>45.6</u>	<u>9.6</u>	<u>77.2</u>
GPT-2 Small								
SFT-CoT	44.1	41.8	9.8	90.7	34.8	–	–	–
No-CoT	19.1	16.4	4.3	41.1	19.1	16.4	4.3	41.1
iCoT	30.1	29.4	5.7	55.5	3.2	–	–	–
Coconut	34.1	36.4	7.9	<u>82.2</u>	24.9	–	–	–
CODI	<u>43.7</u>	**42.9**	<u>9.9</u>	**92.8**	<u>35.3</u>	–	–	–
SIM-CoT	42.6	42.6	9.4	**92.8**	30.9	<u>27.5</u>	<u>6.5</u>	<u>53.9</u>
SIM-CoT (w/o global)	29.5	26.5	6.8	48.9	21.4	23.4	4.9	34.4
MUX	**44.6**	<u>42.7</u>	**10.0**	76.7	**37.3**	**36.4**	**8.3**	**71.1**
MUX (w/o global)	38.7	34.4	9.0	75.6	31.8	<u>28.1</u>	<u>7.0</u>	48.3

Table 1: **Main results on in-domain and out-of-domain mathematical reasoning benchmarks.** For each training corpus, **ID** denotes the in-domain test split, while **SVAMP**, **GSM-Hard**, and **MultiArith** are out-of-domain evaluations. All numbers are final-answer accuracy (%). We treat **SFT-CoT** as an explicit-reasoning upper bound and do not include it when marking the best and second-best continuous-reasoning results; among the remaining methods, the best and second-best entries are highlighted in **bold** and <u>underlined</u>, respectively. A dash indicates an unavailable result.

In-domain results. On in-domain evaluation, MUX improves over CODI in all four directly comparable settings: with LLaMA 3.2 1B-Instruct, it improves from 55.6 to 56.6 on GSM8K-Aug and from 49.7 to 50.6 on GSM8K-Aug-NL; with GPT-2 Small, it improves accuracy from 43.7 to 44.6 on GSM8K-Aug and from 35.3 to 37.3 on GSM8K-Aug-NL. These gains are modest but highly consistent, which is notable because CODI is already a strong continuous-reasoning baseline. Even more strikingly, with GPT-2 Small our method surpasses SFT-CoT on both in-domain settings, achieving 44.6 vs. 44.1 on GSM8K-Aug and 37.3 vs. 34.8 on GSM8K-Aug-NL. This is a surprising result; although discrete reasoning is understood as an upper bound for continuous reasoning distillation, our local distillation method appears to regularize the small model such that it can outperform the discrete reasoning it is distilled from in this regime.

Out-of-domain generalization. MUX also transfers well under distribution shift. On LLaMA 3.2 1B-Instruct trained with GSM8K-Aug, it achieves the best non-SFT performance on GSM-Hard and MultiArith and the second-best on SVAMP. When trained on GSM8K-Aug-NL, it is best on all four reported benchmarks. On GPT-2 Small, the same trend holds on GSM8K-Aug-NL, where MUX is best in all four columns. The main exception is GPT-2 Small trained on GSM8K-Aug and evaluated on MultiArith, where CODI and SIM-CoT both achieve 92.8 while MUX reaches 76.7. This appears to be the one regime where aggressive local compression on a small backbone is less effective than stronger trajectory-level or auxiliary-decoder supervision. Importantly, this weakness disappears once the backbone is larger or the training traces are more natural language rich.

Largest gains on natural-language traces. The clearest advantage of MUX appears on GSM8K-Aug-NL, where the discrete reasoning traces are more verbose and linguistically redundant. On GPT-2 Small, MUX improves over SIM-CoT by +6.4 points on ID, +8.9 on SVAMP, +1.8 on GSM-Hard, and +17.2 on MultiArith. On LLaMA 3.2 1B-Instruct, the margins are even larger: +22.2 on ID, +15.1 on SVAMP, +5.2 on GSM-Hard, and +37.8 on MultiArith. This result strongly suggests that our span-level supervision is especially effective when the discrete reasoning trace becomes longer and more natural language heavy, a setting that can be emphasized as more challenging.

Quality of local distillation. Particularly encouraging results are observed when global distillation loss is removed, which isolates the performance of local distillation. MUX (w/o global) outperforms SIM-CoT (w/o global) in all 16 settings in Table 1, often by large margins. For example, on GPT-2 Small trained on GSM8K-Aug, MUX (w/o global) improves over SIM-CoT (w/o global) by +17.2 on ID and +26.7 on MultiArith; on LLaMA 3.2 1B-Instruct trained on GSM8K-Aug-NL, the gains are +10.2 on ID and +18.9 on MultiArith. These results show that the quality of local supervision provided by our method is better than SIM-CoT, although our method is simpler as it does not need an auxiliary autoregressive decoder.

5.3 Ablation Studies

Number of supervised continuous tokens. We vary how many continuous tokens receive span-level supervision to understand the effect of our supervision signal. Figure 3 shows that accuracy rises from 32.7% with 0 supervised continuous tokens to 48.2% with 6. Here, 0 supervised tokens means that the model still performs continuous reasoning, but no tokens are aligned with discrete reasoning. This removes local span-level supervision while retaining latent computation. The performance gains of MUX therefore come from local distillation, not merely from additional latent steps.

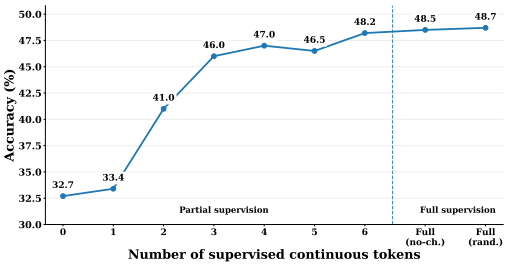

Figure 3: Effect of supervision on accuracy.

Performance improves rapidly up to about 3 supervised continuous tokens and then begins to saturate. This matches the data distribution, as most of the discrete reasoning traces in the training data contain only 2–3 reasoning spans, and supervising more than 3 continuous tokens improves the learning signal for only a small subset of training data that has > 3 reasoning spans.

Strategy	ID	SVAMP	GSM-Hard	MultiArith
Random	**56.6**	**61.3**	**13.0**	**98.3**
Deterministic	55.7	60.6	12.6	97.3
None	54.3	**61.3**	12.5	96.5

Table 2: Comparison of chunking strategies.

Chunking strategies. When $M > K$, the discrete reasoning spans must be merged into K spans before constructing supervision targets for the K continuous tokens. We compare *random chunking* (a random contiguous partition), *deterministic chunking* (a fixed nearly uniform contiguous partition), and *no chunking* (do not merge and truncate at K spans, which is not *lossless*). Table 2 shows that random chunking performs best overall, achieving the highest accuracy on ID, GSM-Hard, and MultiArith, while tying for best on SVAMP. The differences are modest, which is expected. Most training examples contain fewer reasoning spans than continuous tokens, so chunking is not activated for a large fraction of the data. Still, random chunking is consistently the strongest strategy. We attribute this to its resampled boundaries, which act as a form of structured data augmentation. Deterministic chunking is slightly weaker, likely because it exposes the model to only one fixed partition of each trace. No chunking is weakest overall, which is consistent with our theory: when the number of reasoning spans exceeds the number of continuous tokens, no chunking is not lossless and drops part of the reasoning trace.

Positional Weighting. We now test the practical impact of positional weighting that theoretically affects losslessness of the multiplexing. Table 3 compares our default geometric weighting, whose injectivity is characterized by Theorem 4, against uniform weighting, which is not lossless. Geometric weighting performs better on the in-domain split, GSM-Hard and MultiArith, while uniform weighting is slightly better on SVAMP. The gap is modest, which could be attributed to the fact that many reasoning spans in GSM8K-Aug are highly structured, e.g., $<<60/2 = 30>>$, so ordering of subwords can often be inferred even from bags of subwords. Nevertheless, the consistent gains on ID, GSM-Hard, and MultiArith suggest that losslessness in the target is beneficial in practice.

Weighting	ID	SVAMP	GSM-Hard	MultiArith
Geometric	**56.6**	61.3	**13.0**	**98.3**
Uniform	54.2	**62.4**	12.8	96.1

Table 3: Effect of positional weighting.

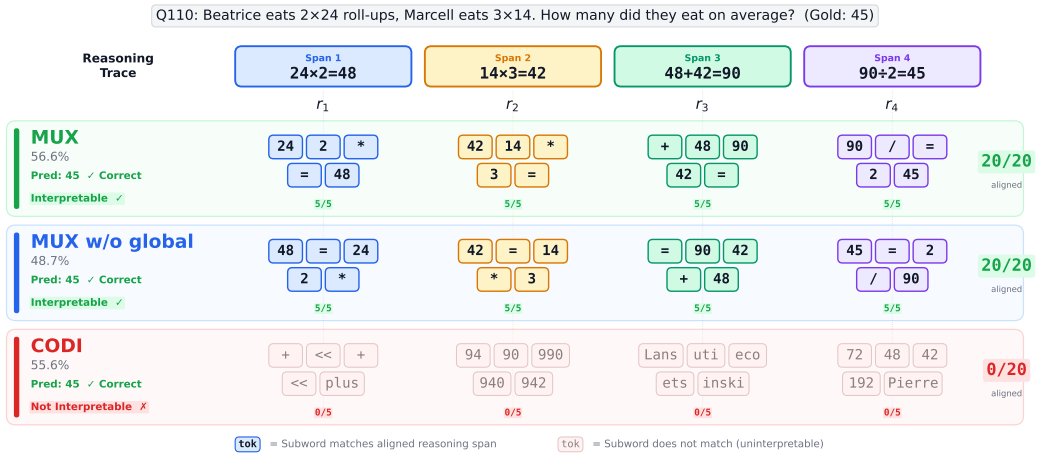

Figure 5: **Interpretability case study.** Top-5 decoded subwords from each continuous reasoning token for a representative example. MUX yields step-aligned and human-readable continuous reasoning tokens; MUX w/o global remains interpretable despite lower overall accuracy; CODI predicts the correct answer but its reasoning tokens are largely uninterpretable.

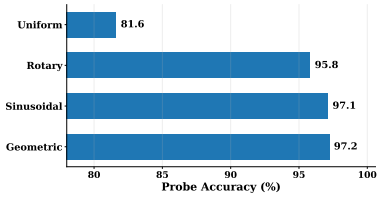

Figure 4: Span recovery across positional weighting schemes.

This supports the intuition that lossless multiplexing improves distillation. We provide further experimental details in Appendix A.3. To test this intuition, we train a small MLP to recover a discrete reasoning span $\mathbf{r}_k = (\mathbf{r}_k^1, ..., \mathbf{r}_k^{S_k})$ from its multiplexing $\text{mux}(\mathbf{r}_k)$. As shown in Figure 4, uniform weighting shows non-trivial accuracy, confirming that ordering of subwords is partially recoverable from their occurrence counts. However, all non-uniform positional schemes perform substantially better, reaching around 96–97% accuracy.

5.4 Interpretability Analysis

Prior works probe interpretability by decoding each continuous reasoning token into vocabulary space (Shen et al., 2025; Kuzina et al., 2025; Wei et al., 2025). Following this approach, we decode the top-5 tokens from each continuous token and compare them against the aligned reference span. Figure 5 shows a representative example. Our method (**MUX**) produces clearly interpretable continuous tokens: the top decoded subwords correspond closely to the operands, operators, and intermediate results in the reference trace. **MUX w/o global** remains similarly interpretable despite lower task performance. By contrast, **CODI** predicts the correct answer but its decoded tokens are semantically uninformative and do not align with the underlying arithmetic spans. This qualitative pattern is consistent with the design of our supervision objective. Because each continuous token is trained against a multiplexed vocabulary-space target, it is encouraged to preserve meaningful discrete reasoning content rather than only encode a hidden-state summary. As a result, our continuous tokens are not only useful for prediction, but are also substantially easier to read out and diagnose.

6 Conclusion

We introduced MUX, a simple local-distillation method for continuous reasoning based on position-weighted superposition in vocabulary space. Each continuous token is trained to represent an aligned span of discrete reasoning through a multiplexed target that is easy to compute, theoretically grounded, and empirically effective. We showed that suitable positional weightings support exact span recovery, and that multiplexed targets can express parallel search dynamics. Across multiple models and benchmarks, MUX consistently matched or improved upon strong baselines, with especially large gains on verbose traces. These results suggest that simple, interpretable local targets can make continuous reasoning both stronger and easier to train.

REFERENCES

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.

Halil Alperen Gozeten, M Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. *arXiv preprint arXiv:2505.23648*, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

David Herel and Tomas Mikolov. Thinking tokens for language modeling. *arXiv preprint arXiv:2405.08644*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Wei Huang, Yizhe Xiong, Xin Ye, Zhijie Deng, Hui Chen, Zijia Lin, and Guiguang Ding. Fast quiet-star: Thinking without thought tokens. *CoRR, abs/2505.17746*, 2025.

Anna Kuzina, Maciej Pioro, Paul N Whatmough, and Babak Ehteshami Bejnordi. Kava: Latent reasoning via compressed kv-cache distillation. *arXiv preprint arXiv:2510.02312*, 2025.

Juncai Li, Ru Li, Yuxiang Zhou, Boxiang Ma, and Jeff Z Pan. Chain of thought compression: A theoritical analysis. *arXiv preprint arXiv:2601.21576*, 2026.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Compressing chain-of-thought in llms via step entropy. *arXiv preprint arXiv:2508.03346*, 2025.

Meta. Llama 3.2: Open-Source AI Models by Meta. `https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/`, 2024. Accessed: 2024-09-25.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Yao Tang, Li Dong, Yaru Hao, Qingxiu Dong, Furu Wei, and Jiatao Gu. Multiplex thinking: Reasoning via token-wise branch-and-merge. *arXiv preprint arXiv:2601.08808*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025a.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *arXiv preprint arXiv:2508.03440*, 2025b.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

Jingxian Xu, Mengyu Zhou, Weichang Liu, Hanbing Liu, Shi Han, and Dongmei Zhang. Twt: Thinking without tokens by habitual reasoning distillation with multi-teachers' guidance. *arXiv preprint arXiv:2503.24198*, 2025a.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 2025b.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*, 2025c.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. Lightthinker: Thinking step-by-step compression. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13318–13339, 2025a.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv preprint arXiv:2505.15778*, 2025b.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*, 2025.

A Appendix

A.1 Additional alignment details

The main text only requires g to be order-preserving. For completeness, we summarize the variants used in experiments.

No chunking. Each reasoning span is aligned to one continuous token until one of the two sequences ends. This is lossless only when $M \leq K$.

Deterministic chunking. When $M > K$, the M reasoning spans are partitioned into K contiguous groups whose sizes differ by at most one, with the extra spans assigned to later groups.

Random chunking. When $M > K$, we sample $K - 1$ cut points uniformly without replacement from $\{1, \ldots, M - 1\}$ and use them to form K positive contiguous chunk sizes. This is the variant used in our main experiments.

A.2 Proofs for Section 4

A.2.1 Proof of Theorem 2

We prove both directions.

Sufficiency. Assume $E(\boldsymbol{\alpha}) > 0$. We will show that mux is injective.

Take two token sequences

$$(r^1, \ldots, r^S), \qquad (r'^1, \ldots, r'^S)$$

such that

$$\mathsf{mux}(r^1, \ldots, r^S) = \mathsf{mux}(r'^1, \ldots, r'^S).$$

This means that the two sequences induce exactly the same target distribution over the vocabulary.

For each vocabulary token $v \in V$, define the set of positions at which v appears:

$$A_v = \{j \in \{1, \ldots, S\} : r^j = v\}, \qquad B_v = \{j \in \{1, \ldots, S\} : r'^j = v\}.$$

Because the two target distributions are equal, for every $v \in V$ we have

$$\sum_{j \in A_v} \alpha_j = \sum_{j \in B_v} \alpha_j.$$

Suppose, for contradiction, that $A_v \neq B_v$ for some v. Then the coefficient vector defined by

$$c_j = \begin{cases} 1, & j \in A_v \setminus B_v, \\ -1, & j \in B_v \setminus A_v, \\ 0, & \text{otherwise} \end{cases}$$

is nonzero and satisfies

$$\sum_{j=1}^{S} c_j \alpha_j = \sum_{j \in A_v} \alpha_j - \sum_{j \in B_v} \alpha_j = 0.$$

This contradicts $E(\boldsymbol{\alpha}) > 0$. Therefore $A_v = B_v$ for every token v.

Now fix any position $j \in \{1, \ldots, S\}$. There is exactly one vocabulary token v such that $j \in A_v$, namely $v = r^j$. Since $A_v = B_v$, we also have $j \in B_v$, so $r'^j = v = r^j$. As this holds for every position j, the two sequences are identical. Hence mux is injective.

Necessity. Assume $E(\boldsymbol{\alpha}) = 0$. Then, by definition, there exists a nonzero coefficient vector

$$\mathbf{c} = (c_1, \ldots, c_S) \in \{-1, 0, 1\}^S$$

such that

$$\sum_{j=1}^{S} c_j \alpha_j = 0.$$

Define two subsets

$$A = \{j : c_j = 1\}, \qquad B = \{j : c_j = -1\}.$$

Since $\mathbf{c} \neq \mathbf{0}$, at least one of A or B is non-empty. Moreover,

$$\sum_{j \in A} \alpha_j = \sum_{j \in B} \alpha_j.$$

Choose two distinct vocabulary tokens $u, v \in V$. Construct two sequences by

$$r^j = \begin{cases} u, & j \in A, \\ v, & j \notin A, \end{cases} \qquad r'^j = \begin{cases} u, & j \in B, \\ v, & j \notin B. \end{cases}$$

Because $A \neq B$, the sequences are different. However, the probability mass of token u under the first sequence is $\sum_{j \in A} \alpha_j$, while under the second sequence it is $\sum_{j \in B} \alpha_j$; these are equal. The same is true for token v, since both distributions sum to 1, and all other tokens have probability 0. Therefore the two sequences induce exactly the same target distribution. Hence mux is not injective.

We have shown that mux is injective if and only if $E(\boldsymbol{\alpha}) > 0$.

A.2.2 Proof of Theorem 3

Fix $i \in \{1, \ldots, M\}$. By assumption,

$$E(\boldsymbol{\alpha}^{(i)}) > 0.$$

Therefore, by Theorem 2, the multiplexed target $\mathsf{mux}(\mathbf{r}_i)$ uniquely determines the full aligned span

$$\mathbf{r}_i = (r_i^1, \ldots, r_i^{S_i}).$$

This is true for every span.

Once all spans $\mathbf{r}_i$ have been recovered, the original reasoning trace is obtained by concatenating them in the same order. Thus the collection

$$\{(S_i, \mathsf{mux}(\mathbf{r}_i))\}_{i=1}^{M}$$

determines the full reasoning trace uniquely. That is exactly the claim.

A.2.3 Proof of Theorem 4 and Corollary 5

For geometric weighting,

$$\alpha_j = \frac{\rho^{j-1}}{\sum_{l=1}^{S} \rho^{l-1}}.$$

Let

$$Z_S = \sum_{l=1}^{S} \rho^{l-1}.$$

Since $0 < \rho < 1$, we have $Z_S > 0$.

Take any coefficient vector $\mathbf{c} \in \{-1, 0, 1\}^S$. Then

$$\sum_{j=1}^{S} c_j \alpha_j = \sum_{j=1}^{S} c_j \frac{\rho^{j-1}}{Z_S} = \frac{1}{Z_S} \sum_{j=1}^{S} c_j \rho^{j-1}.$$

Because $Z_S > 0$, this quantity is zero if and only if

$$\sum_{j=1}^{S} c_j \rho^{j-1} = 0.$$

Therefore

$$E(\boldsymbol{\alpha}) > 0 \quad \Longleftrightarrow \quad \sum_{j=1}^{S} c_j \rho^{j-1} \neq 0 \quad \text{for every } \mathbf{c} \in \{-1, 0, 1\}^S \setminus \{\mathbf{0}\}.$$

Theorem 2 now gives the stated equivalence.

To prove Corollary 5, suppose $\rho = p/q \in (0, 1)$ is rational in lowest terms and that geometric weighting were not injective. By Theorem 4, there would exist a nonzero polynomial

$$P(x) = \sum_{j=1}^{S} c_j x^{j-1}, \qquad c_j \in \{-1, 0, 1\},$$

such that $P(\rho) = 0$. If necessary, divide out the largest power of x so that the constant term is nonzero. The resulting polynomial still has integer coefficients, is nonzero, has constant term ± 1, and has leading coefficient ± 1. By the rational root theorem, any rational root must be an integer divisor of the constant term divided by an integer divisor of the leading coefficient, hence must belong to $\{\pm 1\}$. This contradicts $\rho \in (0, 1)$. Therefore no such polynomial exists, and the weighting is injective.

A.2.4 A LEMMA ON EXPONENTIAL POLYNOMIALS

Lemma 11. *Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be pairwise distinct, and let*

$$f(x) = \sum_{m=1}^{n} b_m e^{\lambda_m x}$$

with real coefficients b_m, not all zero. Then f has at most $n - 1$ real zeros.

Proof. We use induction on n.

If $n = 1$, then

$$f(x) = b_1 e^{\lambda_1 x}$$

with $b_1 \neq 0$, so $f(x) \neq 0$ for all x. Thus the claim holds.

Assume the statement holds for $n - 1$, and consider

$$f(x) = \sum_{m=1}^{n} b_m e^{\lambda_m x} \quad \text{with} \quad \lambda_1 < \lambda_2 < \cdots < \lambda_n.$$

Define

$$g(x) = e^{-\lambda_1 x} f(x) = b_1 + \sum_{m=2}^{n} b_m e^{(\lambda_m - \lambda_1)x}.$$

The functions f and g have the same zeros because $e^{-\lambda_1 x}$ is never zero.

Suppose g has N distinct real zeros. By Rolle's theorem, g' has at least $N - 1$ distinct real zeros. But

$$g'(x) = \sum_{m=2}^{n} b_m (\lambda_m - \lambda_1) e^{(\lambda_m - \lambda_1)x}$$

is again an exponential polynomial, now with $n - 1$ pairwise distinct exponents. By the induction hypothesis, g' has at most $n - 2$ real zeros. Therefore $N - 1 \leq n - 2$, which implies $N \leq n - 1$.

Hence g, and therefore also f, has at most $n - 1$ real zeros. □

A.2.5 Proof of Theorem 6, Corollary 7, and Corollary 8

We first prove Theorem 6. Suppose the scores $s_1, \ldots, s_S$ are pairwise distinct, and define

$$\alpha_j(\tau) = \frac{e^{\tau s_j}}{\sum_{l=1}^{S} e^{\tau s_l}}.$$

By Theorem 2, injectivity fails if and only if there exists a nonzero coefficient vector

$$\mathbf{c} = (c_1, \ldots, c_S) \in \{-1, 0, 1\}^S$$

such that

$$\sum_{j=1}^{S} c_j \alpha_j(\tau) = 0.$$

Since the denominator $\sum_l e^{\tau s_l}$ is strictly positive, this is equivalent to

$$\sum_{j=1}^{S} c_j e^{\tau s_j} = 0.$$

For fixed nonzero $\mathbf{c}$, the function

$$f_{\mathbf{c}}(\tau) = \sum_{j=1}^{S} c_j e^{\tau s_j}$$

is a nonzero exponential polynomial with pairwise distinct exponents s_j. By Lemma 11, $f_{\mathbf{c}}$ has only finitely many real zeros.

There are only finitely many nonzero coefficient vectors in $\{-1, 0, 1\}^S$. Therefore the union of the zero sets of all such functions $f_{\mathbf{c}}$ is finite. Call this union D. If $\tau \notin D$, then no nontrivial signed sum vanishes, so $E(\boldsymbol{\alpha}) > 0$. By Theorem 2, the encoding is injective. This proves Theorem 6.

Corollary 7 is immediate because the function

$$u \mapsto \sin\left(\frac{\pi}{2} u\right)$$

is strictly increasing on $[0, 1]$, so the sinusoidal scores are pairwise distinct.

Finally, we prove Corollary 8. Assume

$$\theta_p(S-1) < \pi \qquad \text{for every } p = 1, \ldots, P.$$

Fix $p \in \{1, \ldots, P\}$. For each $j = 1, \ldots, S-1$,

$$0 \le (j-1)\theta_p < j\theta_p < \pi.$$

The cosine function is strictly decreasing on the interval $[0, \pi]$. Hence

$$\cos\big(\theta_p(j-1)\big) > \cos(\theta_p j) \qquad \text{for } j = 1, \ldots, S-1.$$

Averaging these inequalities over $p \in \{1, \ldots, P\}$ gives

$$\frac{1}{P}\sum_{p=1}^{P} \cos\big(\theta_p(j-1)\big) > \frac{1}{P}\sum_{p=1}^{P} \cos(\theta_p j),$$

that is,

$$s_j > s_{j+1} \qquad \text{for } j = 1, \ldots, S-1.$$

Thus the rotary scalar scores are strictly decreasing and therefore pairwise distinct. The injectivity claim then follows immediately from Theorem 6.

A.2.6 CONNECTION WITH COT2 CSFT

We note that the supervision target of uniform distribution over search states at each step coincides exactly with the continuous supervised fine-tuning (CSFT) objective of CoT2 (Gozeten et al., 2025), which was introduced specifically for search tasks.

Proposition 12 (CoT2 CSFT as a special case)**.** *Consider any search task together with a curated set of complete trajectories $\tau^{(1)}, \ldots, \tau^{(B)}$, where $\tau_k^{(b)} \in V$ denotes the state visited by trajectory b at step k. For each step k, form the serial reasoning span $\mathbf{r}_k = (\tau_k^{(1)}, \ldots, \tau_k^{(B)})$ and apply uniform multiplexing. Then the resulting target satisfies*

$$\mathsf{mux}(\mathbf{r}_k)(v) = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}[\tau_k^{(b)} = v],$$

which is exactly the stepwise CSFT target used in CoT2 (Gozeten et al., 2025).

Proof. Under uniform weighting, each position in $\mathbf{r}_k$ receives mass $\alpha_j = 1/B$ for $j = 1, \ldots, B$. By definition of mux (equation 2),

$$\mathsf{mux}(\mathbf{r}_k)(v) = \sum_{j=1}^{B} \alpha_j \cdot \mathbf{1}[\tau_k^{(j)} = v] = \frac{1}{B} \sum_{j=1}^{B} \mathbf{1}[\tau_k^{(j)} = v].$$

This is exactly the empirical distribution over the step-k states of the curated trajectory set.

Hence, when the curated serial segment is chosen in this way, the tokenwise KL target in our method coincides exactly with the CoT2 continuous supervised fine-tuning target at that step. If the final answer token is the same and the coefficients on the stepwise KL terms are matched, then the two per-example objectives differ only in bookkeeping, namely in whether the final answer loss is written as a separate cross-entropy term or absorbed into the overall example-level objective. Note that this is a target-level equivalence, not an architectural one: CoT2 feeds a continuous token formed from vocabulary embeddings, while our model self-feeds a continuous token. This proves the claim. □

A.2.7 PROOF OF THEOREM 9

We construct a recurrence over continuous tokens and verify that it exactly implements breadth-first search.

For a set $B \subseteq \mathcal{N}$, let $1_B \in \{0,1\}^n$ denote its indicator vector, where $n = |\mathcal{N}|$. At step k, let the continuous token be the pair

$$(f_k, u_k) \in \{0,1\}^{2n},$$

where

$$f_k = 1_{F_k}, \qquad u_k = 1_{U_k}.$$

Thus the token stores the current frontier and the set of visited nodes.

Initialize

$$f_0 = 1_{\{s\}}, \qquad u_0 = 1_{\{s\}}.$$

Let $A \in \{0,1\}^{n \times n}$ be the adjacency matrix of the graph, with

$$A_{uv} = 1 \iff (u, v) \in E.$$

Given the token (f_k, u_k), define the next token by

$$g_{k+1} = 1[A^\top f_k > 0],$$
$$f_{k+1} = g_{k+1} \odot (1 - u_k),$$
$$u_{k+1} = u_k + f_{k+1}.$$

We claim that for every $k \leq H$,

$$f_k = 1_{F_k}, \qquad u_k = 1_{U_k}.$$

The claim is immediate at $k = 0$. Assume it holds at step k. For any node $v \in \mathcal{N}$,

$$(g_{k+1})_v = 1 \iff (A^\top f_k)_v > 0 \iff \exists\, u \in F_k \text{ such that } (u, v) \in E \iff v \in N^+(F_k).$$

Therefore

$$g_{k+1} = 1_{N^+(F_k)}.$$

Hence

$$f_{k+1} = 1_{N^+(F_k)} \odot (1 - 1_{U_k}) = 1_{N^+(F_k) \setminus U_k} = 1_{F_{k+1}}.$$

Also, by the BFS update, $F_{k+1} \cap U_k = \varnothing$, so

$$u_{k+1} = u_k + f_{k+1} = 1_{U_k} + 1_{F_{k+1}} = 1_{U_k \cup F_{k+1}} = 1_{U_{k+1}}.$$

This proves the claim by induction.

It follows that the recurrence exactly tracks the breadth-first frontier and visited set at every step. In particular, the final answer is exact:

$$y = 1[t \in U_H].$$

Indeed, since $u_H = 1_{U_H}$ is part of the final token, the answer head can read the coordinate corresponding to t and output the correct answer.

It remains to recover the frontier distribution. If $F_k = \varnothing$, one may use a designated null distribution. Assume now that $F_k \neq \varnothing$. Since

$$f_k = 1_{F_k},$$

the frontier is explicitly encoded in the token, so define

$$p_k(v) = \frac{(f_k)_v}{\|f_k\|_1}.$$

Then

$$p_k(v) = \begin{cases} 1/|F_k|, & v \in F_k, \\ 0, & v \notin F_k. \end{cases}$$

By the setup of Section 4.3, this is exactly $\mathrm{mux}(r_k)$.

Finally, if one insists on a standard softmax readout with finite logits, exact zeros outside F_k are impossible, but arbitrarily good approximation is still possible. For $B > 0$, define

$$\ell_k(v) = B\big((f_k)_v - 1\big).$$

Then

$$\ell_k(v) = \begin{cases} 0, & v \in F_k, \\ -B, & v \notin F_k. \end{cases}$$

Let $m = |F_k|$. The corresponding softmax distribution is

$$p_k^{(B)}(v) = \frac{e^{\ell_k(v)}}{\sum_{u \in N} e^{\ell_k(u)}}.$$

Since

$$\sum_{u \in \mathcal{N}} e^{\ell_k(u)} = m + (|\mathcal{N}| - m)e^{-B},$$

we obtain

$$p_k^{(B)}(v) = \begin{cases} \dfrac{1}{m + (|\mathcal{N}| - m)e^{-B}}, & v \in F_k, \\ \dfrac{e^{-B}}{m + (|\mathcal{N}| - m)e^{-B}}, & v \notin F_k. \end{cases}$$

Therefore

$$p_k^{(B)} \to \mathrm{mux}(r_k) \qquad \text{as } B \to \infty.$$

So the frontier distribution is recoverable from the continuous token exactly, and realizable by a standard softmax readout up to arbitrarily small error. This proves the theorem.

A.2.8 PROOF OF COROLLARY 10

Let $\mathcal{S}$ be a finite state space and $\mathcal{M}$ a finite auxiliary-memory space. At step k, the search algorithm maintains a frontier $F_k \subseteq \mathcal{S}$ and a memory value $m_k \in \mathcal{M}$, and updates them according to

$$F_{k+1} = \Psi(F_k, m_k), \qquad m_{k+1} = \Omega(F_k, m_k),$$

with final answer

$$y = \Gamma(m_H).$$

Encode the frontier F_k by its indicator vector

$$\mathbf{f}_k \in \{0,1\}^{|\mathcal{S}|},$$

and encode the memory value m_k by a one-hot vector

$$\mathbf{z}_k \in \{0,1\}^{|\mathcal{M}|}.$$

Define the continuous token at step k to be

$$\mathbf{x}_k = (\mathbf{f}_k, \mathbf{z}_k) \in \{0,1\}^{|\mathcal{S}|+|\mathcal{M}|}.$$

Because the set of valid pairs (F, m) with $F \subseteq \mathcal{S}$ and $m \in \mathcal{M}$ is finite, the update

$$(F_k, m_k) \mapsto (F_{k+1}, m_{k+1})$$

is a deterministic function on a finite set. Therefore there exists a deterministic recurrence

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k)$$

that realizes this update exactly. Hence, for every $k \le H$, the token $\mathbf{x}_k$ exactly encodes (F_k, m_k).

Likewise, since the final answer is

$$y = \Gamma(m_H),$$

there exists a final answer head that reads the memory component $\mathbf{z}_H$ of the final token and outputs the correct answer exactly.

Now assume the reasoning span $\mathbf{r}_k$ serializes the frontier F_k, with each state in F_k appearing exactly once, and that uniform weighting is used. Then

$$\mathsf{mux}(\mathbf{r}_k) = \frac{1}{|F_k|} \sum_{j=1}^{|F_k|} \mathsf{onehot}(\mathbf{r}_k^j),$$

which is exactly the uniform distribution over F_k.

Thus there exists a continuous-token recurrence that tracks (F_k, m_k) exactly, with an exact final answer readout, and whose tokenwise targets are precisely the corresponding frontier distributions. This proves the corollary.

A.3 STEP-RECONSTRUCTION PROBE FOR POSITIONAL WEIGHTING

To further study the role of positional weighting, we train an MLP probe on isolated reasoning spans $\mathbf{r}_k = (r_k^1, \ldots, r_k^{S_k})$ extracted from GSM8K-Aug. For each non-empty span $\mathbf{r}_k$, we construct the same multiplexed target as in the main method:

$$\mathsf{mux}(\mathbf{r}_k)(v) = \sum_{j=1}^{S_k} \alpha_{k,j} \mathbf{1}[r_k^j = v], \qquad \alpha_{k,j} = \frac{w_{k,j}}{\sum_{\ell=1}^{S_k} w_{k,\ell}},$$

where $w_{k,j}$ is determined by the positional weighting scheme under study. For uniform weighting,

$$w_{k,j} = 1 \qquad \text{for all } j, \qquad \text{so} \qquad \alpha_{k,j} = \frac{1}{S_k}.$$

The probe takes $\mathsf{mux}(\mathbf{r}_k)$ as input and predicts the original span $\mathbf{r}_k$. We use a 5-layer MLP with hidden sizes $1024, 512, 256, 512, 1024$ and GELU (Hendrycks & Gimpel, 2016) activations, without input normalization. We set the maximum sequence length to 128, strip the delimiters << and >> from each extracted step, and train for 20 epochs with batch size 128 and learning rate 10^{-3}. The data are extracted from GSM8K-Aug and split into 901,661 training spans and 100,185 evaluation spans using a 0.1 test split. For geometric weighting, we use $\rho = 0.9$. For sinusoidal weighting, we use $\tau = 1$. For rotary weighting, we use $\mathrm{base} = 1000$.

A.4 Extended Related Work

We provide a more detailed discussion of related work, organized by theme.

A.4.1 Explicit Reasoning in Language Models

Chain-of-thought (CoT) prompting (Wei et al., 2022) demonstrated that asking a language model to articulate intermediate reasoning steps dramatically improves performance on arithmetic, symbolic, and commonsense tasks. Nye et al. (2021) introduced scratchpads as a training-time analogue, providing models with intermediate computation buffers. Subsequent work strengthened or structured this paradigm in different ways, including self-consistency (Wang et al., 2022), STaR (Zelikman et al., 2022), least-to-most prompting (Zhou et al., 2022), Tree of Thoughts (Yao et al., 2023), and PAL (Gao et al., 2023). Together, these methods established that explicit intermediate computation is a powerful and general-purpose tool for language-model reasoning.

At the same time, recent work has shown that such traces are often far more verbose than the underlying computation requires. TokenSkip (Xia et al., 2025), step-entropy pruning (Li et al., 2025), LightThinker (Zhang et al., 2025a), and ALiCoT (Li et al., 2026) all suggest that much of a CoT trace is linguistic overhead rather than irreducible computation. This is the setting that motivates MUX. Our goal is not to compress a generated CoT at inference time, but to use the redundancy of discrete traces to train a smaller number of continuous reasoning states.

A.4.2 Implicit Reasoning and Internalization

A related line of work tries to retain the benefits of CoT while removing the need to emit intermediate language at inference time. iCoT (Deng et al., 2023) and its stepwise extension (Deng et al., 2024) distill explicit reasoning into implicit computation. Pause tokens (Goyal et al., 2023), Quiet-STaR (Zelikman et al., 2024), Fast Quiet-STaR (Huang et al., 2025), and thinking tokens (Herel & Mikolov, 2024) increase internal compute by inserting special positions that need not correspond to normal language. Compressed chain-of-thought (Cheng & Van Durme, 2024) similarly moves toward denser reasoning representations. These methods share the goal of increasing the model's effective computation depth without proportionally increasing the output length, but they do not introduce an explicit continuous reasoning loop with recurrent hidden-state feedback.

A.4.3 Continuous Reasoning Methods

Continuous reasoning methods go further by operating in a latent vector space with explicit recurrent feedback. We organize this literature by the type of supervision employed.

Global supervision. Coconut (Hao et al., 2024) is the foundational continuous reasoning method. It replaces discrete CoT tokens with hidden-state recurrence: the last hidden state of the model is fed back as the next input embedding, forming a "chain of continuous thought." Training uses a curriculum that gradually transitions from explicit CoT to fully latent reasoning. Coconut demonstrated that continuous reasoning can enable breadth-first-style exploration, but supervision comes only from the final answer loss, which leaves the intermediate latent trajectory unconstrained. CODI (Shen et al., 2025) strengthened this setup with self-distillation, aligning the continuous and discrete modes at the hidden state used to predict the answer. These methods supervise the reasoning process primarily through the final answer or trajectory endpoint. In our terminology, they are *global* supervision methods: they constrain where the latent trajectory ends up, but not what each intermediate continuous token should represent.

Local supervision via auxiliary components. MUX is closer in spirit to *local* supervision methods, which directly assign a target to each continuous reasoning token. SIM-CoT (Wei et al., 2025) identifies a critical limitation of global supervision: as the number of latent tokens increases, representations become homogeneous and training collapses. To address this, SIM-CoT attaches an auxiliary autoregressive decoder during training that forces each continuous token to reconstruct its aligned explicit reasoning step, providing step-level supervision. KaVa (Kuzina et al., 2025) takes a different approach to local supervision by distilling the teacher's compressed KV-cache into the student model layer by layer. The supervision target is the teacher's cache dynamics rather than the

Table 4: Comparison of continuous reasoning methods along key design axes.

Method	Supervision	Locality	Aux. module	Interpretable
Coconut	Answer loss	Global	✗	✗
CODI	Hidden align.	Global	✗	✗
SIM-CoT	Step decoding	Local	Decoder	✓
KaVa	KV distillation	Local	Compressor	✓
MUX	**Mux target (KL)**	**Local**	✗	✓

output vocabulary, providing a rich but structurally complex signal. Both methods demonstrate the importance of local supervision for continuous reasoning, but they require additional components, an auxiliary decoder and a KV compression module, respectively.

Parallelization and efficiency. PCCoT (Wu et al., 2025a) improves the efficiency of continuous reasoning by parallelizing the sequential latent updates via Jacobi iteration, reducing inference latency while maintaining accuracy. SoftCoT (Xu et al., 2025b) generates soft reasoning tokens through a frozen assistant model and a trained projection layer, and SoftCoT++ (Xu et al., 2025c) extends this to test-time compute scaling. Token Assorted (Su et al., 2025) mixes discrete and continuous tokens in a hybrid reasoning trace, allowing the model to choose when to reason in language and when to reason in latent space. TWT (Xu et al., 2025a) distills reasoning from multiple teacher models into habitual latent computation.

Inference-time soft reasoning. Another nearby direction performs reasoning in a soft or continuous space at *inference time* by modifying the decoding procedure of a pretrained model. Soft Thinking (Zhang et al., 2025b) replaces hard token selection with probability-weighted mixtures of vocabulary embeddings. Subsequent work studies its limitations and variants, including stochastic soft reasoning (Wu et al., 2025b), Multiplex Thinking (Tang et al., 2026), and SofT-GRPO (Zheng et al., 2025). Multiplex Thinking (Tang et al., 2026) takes a complementary approach: at each reasoning step, it samples K candidate tokens and aggregates their embeddings into a single continuous *multiplex token*, maintaining vocabulary embedding priors while enabling on-policy RL optimization. These methods are complementary to our work: they modify the decoding procedure of a pretrained model, while MUX designs a training-time supervision target for continuous reasoning distillation. This separation lets us use vocabulary space as an interpretable supervision interface without committing to vocabulary-space decoding during generation.

A.4.4 Theoretical Foundations of Continuous Reasoning

A growing theoretical literature formalizes the advantages of continuous over discrete reasoning. Zhu et al. (2025) prove that a two-layer transformer with D steps of continuous CoT can solve directed graph reachability, where D is the graph diameter. The key mechanism is *superposition*: each continuous thought vector encodes multiple search frontiers simultaneously, enabling parallel breadth-first search. In contrast, discrete CoT requires $O(n^2)$ steps with constant-depth transformers, establishing an exponential separation. CoT2 (Gozeten et al., 2025) provides complementary results for search problems, showing that supervision against latent token distributions induces parallel exploration. Our work connects to this theoretical line in two ways. First, we prove that our multiplexed supervision targets are *lossless* under standard positional weightings, meaning the full discrete reasoning chain can be recovered from the continuous targets. Second, we show that under uniform weighting, our target construction exactly recovers the CoT2 supervision objective, and we prove that a latent recurrence over such targets can implement exact parallel breadth-first exploration. These results unify the supervision-design and parallel-search perspectives within a single framework.

A.4.5 Positioning of Our Work

Table 4 summarizes how MUX relates to the most closely related continuous reasoning methods along key design dimensions.