# LLM-AS-ITS-JUDGE: A COMPUTE PARADIGM FOR SELF-AWARE AND SELF-DIRECTING REASONING MACHINES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language models (LLMs) can perform multi-step reasoning to divide complex problems into a sequence of conquerable parts. These reasoning steps empowered LLMs for mathematical, logical, and algorithmic feats beyond one-step generation. Yet, with token generation being strictly autoregressive, early mistakes compound and subtle missteps can derail the entire reasoning trajectory. We demonstrate that LLMs can sense when these reasoning trajectories approach a tipping point to go off road. Then, by injecting specialized control tokens, LLMs can learn to push their own reasoning trajectories back on track. To bring "self-awareness" to LLM inference, we introduce *reflexive reasoning*, a novel reasoning paradigm complementary to current reasoning approaches. Across multiple standard benchmarks, reflexive reasoning improves the task performance of LLMs spanning a range of architectures and parameter counts, with no modification of LLM weights and no added inference-time compute. Through self-reasoning, we can also enable *reflexive tuning*, a fine-tuning framework to enforce tight policy and persona adherence at inference time, with fixed model weights. By repurposing the model's internal representations to serve as its own observer and overseer, our approach delivers robust-by-design inference for complex, high-risk reasoning applications, with minimal computational overhead.

## 1 INTRODUCTION

Large language models (LLMs) show remarkable abilities to reason through complex problems step-by-step. They achieve strong results on demanding tasks involving math, programming, and scientific reasoning (Kojima et al., 2023; Wei et al., 2023; Xu et al., 2025). These reasoning abilities have been greatly enhanced through a combination of large scale pre-training, supervised fine-tuning on multi-step reasoning traces, and reinforcement learning using human, model-generated, or verifiable feedback (OpenAI et al., 2024; Zelikman et al., 2022). The resulting "chains of thought" generated by these post-trained LLMs consist of intermediate reasoning steps that break a problem into smaller components and revise possible errors before arriving at a final answer.

These chains of thought act as discrete scratchpads, which give the model time and space to externalize intermediate computations and condition each reasoning step on those that preceded it. However, this sequential dependency cuts both ways: since token generation is strictly autoregressive, early mistakes can propagate through subsequent steps and derail the entire reasoning trajectory. Even before explicit errors manifest concretely, seemingly innocuous token choices can commit the model to unproductive branches of the solution space (Bigelow et al., 2024). This progressively narrows the funnel of plausible continuations and locks the reasoning process into an unsuccessful branch. Existing remedies fall largely into two camps: trial-and-error methods such as search, backtracking, or LLM-as-a-judge loops (Yao et al., 2023; Shinn et al., 2023) which incur heavy latency penalties and remain blind to errors until after bad tokens are emitted, or representation steering frameworks (Wu et al., 2024) that, through techniques such as sparse autoencoders (Cunningham et al., 2023), lack precision and treat intervention as a detached, asynchronous prosthesis. Rather than bolting on auxiliary evaluators or disparate steering modules, we argue that inference control should be baked natively into the LLM itself. Through internet-scale pre-training, it has more than enough representational capacity to function as its own introspective governor.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We here introduce a novel reasoning paradigm: the model bootstraps its own internal representations to sense impending trajectory deviations and proactively ushers the reasoning path toward more optimal solution spaces. Reflecting the model's internal representation space back upon itself creates an introspective, closed-loop system that we call *reflexive reasoning*. We learn specialized sensor and control tokens that propagate through the model forward pass, fully co-opting the model's intermediate computations to eavesdrop on its internal dynamics ("listen to its internal thoughts") and execute subtle interventions to maximize adherence to correct reasoning policies. While prevailing approaches attempt to coax meticulous reasoning out of the model through heavy reinforcement learning and verbose discrete-token scratchpads, reflexive reasoning repurposes the LLM's forward pass as a dual-purpose computer to close the loop between thought generation and thought regulation. Our key insight is that the internal representation space of massively pre-trained LLMs is rich enough to carry out these dual tasks of generation and introspection, without modification to the existing model architecture or parameter weight set and no additional inference-time compute.

Experimentally, we introduce learned sensor and control tokens, inserted at critical points during autoregressive text generation. They have sufficient capacity to reliably detect impending errors and steer answer trajectories toward correct solutions on challenging reasoning problems. Across frozen Qwen3 (Yang et al., 2025) and Gemma-3 (Kamath et al., 2025) family models, the learned sensor token anticipates whether an intervention via self-steering will improve the final answer, while the control token finesses the continuation without regenerating or branching. This shows remarkable ability to influence subsequent text on the fly, without external or asynchronous approaches for monitoring or intervention on the execution path. Covering diverse reasoning tasks, this closed-loop consistently improves exact-match accuracy compared to vanilla chain-of-thought. For non-reasoning tasks, it enforces strict behavioral and format adherence; better than standard prompting. Furthermore, by operating within a highly expressive and compact latent space, reflexive reasoning achieves these results using fewer tokens than approaches such as conventional chain-of-thought that operate over discrete token sequences. As a starting point, we believe that these reflexive approaches have the potential to elevate pre-trained LLMs beyond discrete token generation, harnessing their rich internal representations as general-purpose cognitive substrates for robust-by-design inference.

## 2 Related work

Chain-of-thought (CoT) reasoning is a class of techniques whose goal is to influence the LLM to show its work step-by-step before outputting an answer. These discrete intermediate steps can be induced through prompting (Wei et al., 2023), supervised fine-tuning (SFT) on gold reasoning traces (Zelikman et al., 2022), or reinforcement learning with intermediate and final rewards (OpenAI et al., 2024). Extensions to CoT typically aim to improve reasoning robustness, but also token efficiency. Addressing the former, self-consistency generates multiple candidate reasoning trajectories for a single prompt and uses a majority vote across these trajectories to settle on a robust final answer (Wang et al., 2023). Pause tokens insert dummy tokens to delay outputs, enabling additional forward passes across the sequence before outputting text tokens (Goyal et al., 2024). Coconut compresses discrete CoT steps into fewer, richer continuous tokens at the start of generation (Hao et al., 2026). Conversely, Reflexion requires no weight updates; a separate validator generates text feedback to iteratively refine trajectories in context (Shinn et al., 2023). In the present work, we introduce reflexive reasoning that operates over a single generated trajectory per prompt, detects the need for intervention introspectively unlike reflexion's external validator, and steers an ongoing response with conditionally inserted continuous tokens. This is while keeping the model weights frozen, unlike pause tokens and Coconut that require updates to model weights.

Moreover, parameter-efficient fine-tuning (PEFT) methods update a small set of parameters to adapt language models: low rank adaptation (LoRA) interjects trainable decomposition matrices into the attention and feed-forward layers of the LLM, providing task-specific updates with minimal parameter overhead (Hu et al., 2021). Instead, soft prompt tuning requires no modification to the LLM weights, but instead prepends trainable continuous tokens to the input sequence (Lester et al., 2021). Our method is a cousin of soft prompt tuning, working over sequence space instead of the LLM weight space and adding tuneable continuous tokens. Unlike soft prompts, our reflexive control tokens are applied in the middle of the model-generated sequence, not prepended to the prompt, and are dynamically applied only when necessary, rather than injecting and operating unconditionally. We also draw inspiration from work on looped transformers to motivate our reflexive approach:

repeated computation over internal representations can carry out advanced reasoning that can go beyond what human language can explicitly verbalize (Giannou et al., 2023; Mozer et al., 2026). Instead of training from scratch, we reason within a pre-trained model's existing representation space, as it unfolds.

Activation steering provides another angle of attack for guiding LLMs without modifying model weights. Techniques such as representation engineering (Zou et al., 2025; Wu et al., 2024) and steering vectors (Subramani et al., 2022), whether produced contrastively (Turner et al., 2024) or gleaned from sparse autoencoders (Cunningham et al., 2023; Templeton et al., 2024; Suresh et al., 2026), extract directions in the model's latent space corresponding to specific behaviors (e.g. truthfulness, hallucination, refusal, or task-specific performance) to intervene directly on internal activations at inference-time. Such approaches have even been combined with PID-like controllers (Bharadwaj, 2026). However, these methods typically apply stereotypical, static interventions computed offline, lacking adaptability to the model's live internal state needed to precisely detect and correct errors as they arise within a sequence. By contrast, we empower the model's own forward pass to design the optimal steering vectors to control its reasoning process.

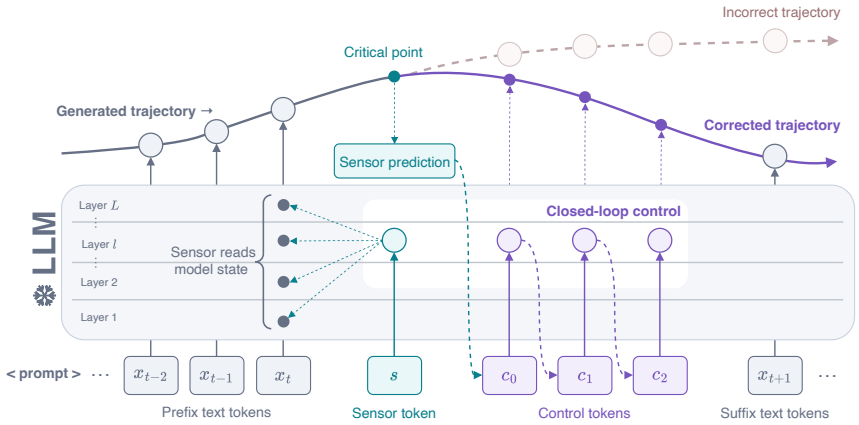


Figure 1: **Extending language models to monitor and steer themselves at inference time.** At each natural prose boundary in the generated trajectory (e.g. punctuation and newlines at the end of semantic spans, excluding equations, numbers, or code) a learned sensor token is inserted into the frozen LLM, reusing the model's forward pass and its internal state to estimate whether the current trajectory requires correction. At sensor-determined critical points in the execution trajectory, a control token is injected into the model to recover off-track reasoning trajectories. Each control token is constructed from the hidden state of the previous control token, harvested from LLM layer $l$. We thus design a closed-loop system that continually pulls the reasoning chain away from impending failures and towards the correct trajectory.

## 3 METHODS

In this section, we outline our method for stitching a closed-loop controller into a frozen autoregressive language model. The key insight we leveraged in designing this protocol is that we can introduce learnable synthetic tokens that, as they flow through the model, make use of its forward pass computations and spanned representation to unlock powerful monitoring and correction capabilities, from within.

### 3.1 PRELIMINARIES

Let $f_\theta$ denote a transformer with fixed parameters $\theta$ and $L$ layers, and let $x_{1:t}$ be output tokens generated by the language model up to step $t$, conditioned on an input prompt $\mathcal{P}$. We hereafter refer to the sequence of tokens including the prompt and model-generated tokens up to step $t$ as a prefix: $\{\mathcal{P}, x_{1:t}\}$. At each decoding step $t$, the base LLM defines an unconstrained distribution over the discrete vocabulary $\mathcal{V}$ given the prefix: $p_\theta(x_{t+1} \mid \mathcal{P}, x_{1:t}) = \text{softmax}(z_t)$, where $z_t \in \mathbb{R}^{|\mathcal{V}|}$

represents the unnormalized logits produced by $f_\theta$. This unconstrained distribution seeks to maximize the conditional log-likelihood of the next token according to the training data distribution: $\max_{x_{t+1}} \log p_\theta(x_{t+1} \mid \mathcal{P}, x_{1:t})$, without regard to trajectory-level objectives. While extensive reinforcement learning post-training modifies this raw objective significantly, these policies operate strictly offline at inference time, generating tokens autoregressively without introspective error correction. Thus, at certain critical points in the reasoning trajectory, a single off-distribution token or subtle glitch can kick off a compounding error cascade, irrecoverably drifting the generation away from the target objective without an explicit safeguard to detect or circumvent this failure.

### 3.2 Sensor and control tokens

To close the loop between runtime state estimation and actively shaping the ongoing trajectory, reflexive reasoning introduces two learned embeddings: a continuous sensor token $s \in \mathbb{R}^{d_{\text{model}}}$ and an initial continuous control token $c_0 \in \mathbb{R}^{d_{\text{model}}}$, followed by additional control tokens $c_i \in \mathbb{R}^{d_{\text{model}}}$. These embeddings serve as portals into the frozen LLM's representation space:

**Sensor token** ($s$): Rather than evaluating activations at every token step, $s$ is appended strictly at natural prose transitions (e.g. punctuation or newlines at the end of semantic spans, explicitly excluding placement in equations, numbers, or code). The hidden state of $s$ at an intermediate transformer layer $l_s$ (e.g. layer 20 of Qwen3-0.6B) is fed into a linear classifier to produce a score $\hat{y}_t = \sigma\big(W_s h_t^{(l_s)}[s] + b_s\big)$, where $\hat{y}_t \in [0, 1]$ at step $t$. The linear transformations $W_s$ and $b_s$ are learned in tandem with the sensor embedding on the same objective. The forward pass corresponding to $s$ is entirely discarded, ensuring that subsequent text generations are not influenced by the presence of the sensor.

**Control tokens** ($c_k$): When $\hat{y}_t$ exceeds a calibration threshold $\tau$, $K$ control tokens are injected into the sequence at step $t$. We first inject a learnable proposal token $c_0$, which is anchored to the median norm of typical vocabulary embeddings. For each control step $k \in 1, \ldots, K$, the hidden state at layer $l_c$ is extracted from the model, norm-matched to the proposal $c_0$, and combined with $c_0$ via a bottleneck gating network. Importantly, we construct these control tokens recurrently: the hidden state of $c_k$ is used to construct the initial embedding for $c_{k+1}$. Letting $h_k$ denote the hidden state of the control token $c_k$ at layer $l_c$, we first match its scale to the proposal $c_0$ via $\hat{h}_k = h_k \frac{\|c_0\|_2}{\|h_k\|_2 + \epsilon}$ where $\epsilon$ is a constant for stability. A small bottleneck network then predicts an interpolation vector $\alpha_k \in (0, \alpha_{\max})^{d_{\text{model}}}$ to construct the next control token $c_{k+1}$, projecting the result to maintain the norm of $c_0$:

$$\tilde{c}_{k+1} = (1 - \alpha_k) \odot c_0 + \alpha_k \odot \hat{h}_k, \qquad c_{k+1} = \tilde{c}_{k+1} \frac{\|c_0\|_2}{\|\tilde{c}_{k+1}\|_2 + \epsilon}$$

After $K$ steps, the generated sequence $[c_1, \ldots, c_K]$ is appended to the prefix at timestep $t$ to steer the remaining autoregressive generation. The proposal $c_0$ is discarded.

In essence, $s$ acts as a silent pulse-check on the vitals of the unfolding reasoning trajectory, while $c_{\leq i}$ inject high-dose corrective perturbations to guide the autoregressive policy towards valid reasoning trajectories.

### 3.3 Reflexive training regimen

Our reflexive training protocol preserves the frozen transformer weights $\theta$ throughout, optimizing only the sensor and control tokens. To train these tokens and their auxiliary transformations, we generate reasoning traces from a bank of prompts for a given dataset. In this work, we use GSM8K for mathematical reasoning (Cobbe et al., 2021), PrOntoQA for logical reasoning (Saparov & He, 2023), and CRUXEval for programming (Gu et al., 2024). We examine Qwen3-0.6B, Qwen3-1.7B, and Gemma-3-4B-IT, and enable "thinking" mode for all models, and models are instructed to show their work step-by-step. We performed binary labeling of generated reasoning traces as correct or incorrect via exact matching to gold answers provided by each dataset.

The sensor token $s$ and linear classifier are trained via binary cross-entropy on cached generated prefixes. The sensor token checkpoint with the highest validation area under the precision-recall curve (AUPRC) is selected. We then calibrate an intervention threshold $\tau$ on validation rollouts to maximize a utility function that penalizes false positives: $\text{TP}_\tau - \lambda_{\text{FP}}\text{FP}_\tau$.

Using only incorrect sampled traces, we extract prefixes at safe boundaries and pair them with ground-truth repair continuations. The proposal token $c_0$ and the gating network are trained to minimize the negative log-likelihood of these target tokens under teacher forcing for $M$ examples:

$$\mathcal{L}_{\text{SFT}} = -\sum_{m=1}^{M} \log p_\theta(y_m \mid \mathcal{P}, x_{1:t}, c_{1:K}, y_{<m})$$

We empirically set $K = 3$ for all experiments in the main text (Appendix A.1 for more details). Finally, to improve repair quality beyond static continuations, the controller is fine-tuned using group-relative policy optimization (GRPO). For each failed prefix, we sample a group of $G$ controlled completions $y^{(g)}$ of length $M_g$ and evaluate their final answers with an outcome reward $R_g \in [0, 1]$ derived from the exact-match comparison to the gold answers. The advantages are normalized across each group: $A_g = \frac{R_g - \mu_R}{\sigma_R + \epsilon}$, where $\mu_R$ and $\sigma_R$ are the mean and standard deviation of the rewards, respectively. We update the controller by weighting the length-normalized sequence log-probabilities by these advantages:

$$\mathcal{L}_{\text{policy}} = -\frac{1}{G}\sum_{g=1}^{G} A_g \left( \frac{1}{M_g} \sum_{m=1}^{M_g} \log p_\theta(y_m^{(g)} \mid \mathcal{P}, x_{1:t}, c_{1:K}, y_{<m}^{(g)}) \right)$$

The total training objective combines this policy loss with the supervised loss weighted by $\beta$ to prevent incoherent or degenerate outputs: $\mathcal{L} = \mathcal{L}_{\text{policy}} + \beta \mathcal{L}_{\text{SFT}}$.

## 4 Anticipating and Correcting Reasoning Failures

In this section, we isolate and evaluate the two mechanisms of our reflexive reasoning framework: the sensor token's ability to detect impending reasoning failures and the control tokens' capacity to steer LLM generations toward the desired solution.

### 4.1 Sensor tokens anticipate failure before errors materialize

The primary objective of our reflexive approach is to identify the breaking points in the reasoning trajectory where intervention is most helpful. Beyond these critical points, errors have either already manifested, or the model has embarked on an incorrect reasoning path that is largely unrecoverable. We hypothesized that such critical inflection points are predictable ahead of time, and that the model's own forward pass computation can be repurposed to predict these moments in time accurately.

To directly test these contentions, we generated a reasoning trajectory dataset from Qwen3-0.6B on the GSM8K training set of grade-school level math word problems. With these 7,473 trajectories compiled, we inserted a sensor token at natural transition points in the reasoning trajectory: periods, commas, and newlines in the generated text. Since the sensor has access to the full internal state of the model's reasoning trajectory up to its insertion point, we could then adjust the initial sensor embedding to absorb only the most relevant prognostic signals as it is transformed by the model's forward pass. We harvest sensor representations from layer 20, which yielded the best empirical derailment prediction performance across tested depths (Appendix A.1), and pass these through a paired linear classifier for binary prediction of impending reasoning failures at the end of the trajectory. See Section 3.3 for a detailed overview of sensor token training. We compare our learned sensor token against two existing protocols: next-token entropy as a measure of general model uncertainty, and a trained linear probe over the layer 20 hidden states averaged across the preceding 16 text tokens.

We find that our sensor token approach achieves an AUC of 0.807 predicting held-out Qwen3-0.6B trajectory failures from 1,319 GSM8K test set examples. Our performance is significantly higher than simple next-token entropy with an AUC of 0.557 and comfortably above the trained linear probe with an AUC of 0.695 on the same task (Figure 2A). These results indicate that recoverable trajectory errors are not detectable on the single-token level, but manifest across the LLM system as a whole.

As well, the specialized task-specificity of our learned sensor token provides much needed representational fidelity compared with linear probes trained on generic text token representations. On the

aggregate (Figure 2B), above-threshold sensor detection of future reasoning derailments reveal that useful interventions concentrate after the initial reasoning plan but before the model has committed to an answer. We also provide representative reasoning chain prefixes where our learned sensor tokens detect an impending reasoning deviation in Appendix A.4. These results point to trained sensor tokens acting as useful oracles of upcoming errors in model-driven reasoning chains.

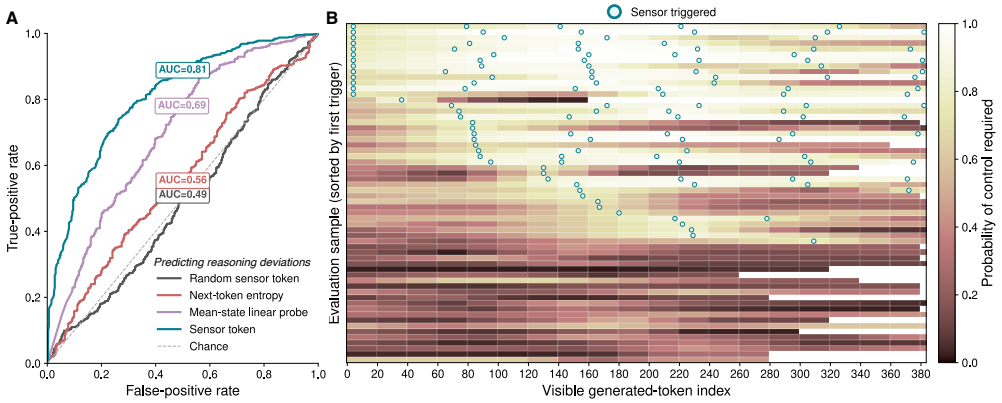


Figure 2: **Sensor tokens accurately predict upcoming errors during reasoning trajectory. A.** ROC curves comparing reasoning GSM8K error prediction performance of our learned sensor token to a random sensor, next-token entropy, and text token linear probes. Sensor tokens and text token hidden states are collected from layer 20 of Qwen3-0.6B, linear probes are fit over the average representations of the preceding 16 tokens from the sensor position. Our sensor token approach outperforms the local next-token entropy predictions and the generic text token linear probes. **B.** A representative random selection of 60 Qwen3-0.6B reasoning trajectories from the GSM8K test set. Each row shows the trained sensor's trigger probability across the generated token sequence for one held-out example on GSM8K, with examples ordered by the time of the first intervention, in 20-token bins (y axis). Teal markers indicate positions at which the sensor detects a deviation above its validation set calibrated threshold, while lighter colored bins indicate increased probability that control is required. In general, sensor trigger probability rises after the model has established an initial reasoning plan but before it has committed to a final answer, identifying a window in which errors are both detectable and preventable.

### 4.2 Control tokens can micromanage model generation

We next sought to determine the expressiveness and generalization ability of our self-control augmented LLM. Timely interventions would be entirely ineffective if the inserted control tokens could not exert enough influence on the subsequent text tokens to steer the reasoning trajectory in the appropriate direction. We therefore tested how forcefully and precisely these control tokens could steer generation: namely, whether an injected burst of control tokens can reliably usher a frozen model toward an arbitrary target continuation.

To test the override strength and exactness of our control tokens on Qwen3-0.6B, we stress-tested both arbitrary memorization and systematic generalization. For memorization, we trained our control tokens to induce exact generation of heterogeneous, low-probability output tokens across variable length prompts, with target outputs including i) a grammatically correct but improbable sentence ("Purple elephants juggle seventeen teaspoons."), ii) a fluent but semantically contradictory statement ("The answer is 999, because two plus two equals a pineapple."), and iii) random alphanumeric strings. In just 100 optimization steps, the control tokens achieved exact generation across WikiText prefixes up to 16,384 tokens and for random alphanumeric output sequences up to 64 tokens in length. To test generalization, we trained a single control burst to map prompt attributes to an invented schema that the model had never encountered. While uncontrolled accuracy was 0% on this generalization task, the trained control token burst achieved up to 76.0% exact-match accuracy on 96 held-out test examples (Appendix A.2). These results demonstrate that the control tokens effectively repurpose internal model computations, learning systematic transformations of latent states that generalize across disparate prompt structures, long-context prefixes, and entirely novel output formats.

Table 1: **Reflexive reasoning augments CoT on reasoning tasks**. Exact-match test set accuracy (pass@1, %) and generation efficiency (average token counts per response) across three mathematical reasoning, logical reasoning, and coding benchmarks. Increased accuracy indicates better reasoning performance, and lower average token counts indicate better efficiency.

| Model | Method | GSM8K | | PrOntoQA | | CRUXEval | |
|---|---|---|---|---|---|---|---|
| | | pass@1 | tokens | pass@1 | tokens | pass@1 | tokens |
| Qwen3-0.6B | CoT | 30.0 | 377.8 | 28.4 | 255.9 | 29.0 | 1002.9 |
| | CoT + soft prompt | 39.3 | 328.4 | 30.8 | 248.2 | 31.7 | **205.4** |
| | CoT + reflexive reasoning (ours) | **40.1** | **233.2** | **44.6** | **133.7** | **34.3** | 282.1 |
| Qwen3-1.7B | CoT | 19.5 | 383.6 | 19.2 | 254.5 | 40.0 | 1151.2 |
| | CoT + soft prompt | 36.0 | 351.2 | 23.2 | 249.8 | 32.0 | **110.0** |
| | CoT + reflexive reasoning (ours) | **56.9** | **183.9** | **42.0** | **151.3** | **46.0** | 362.8 |
| Gemma-3-4B-IT | CoT | 75.1 | 203.2 | 42.6 | 276.1 | 46.0 | 496.7 |
| | CoT + soft prompt | 73.3 | **198.9** | 38.4 | 210.7 | 26.7 | **41.6** |
| | CoT + reflexive reasoning (ours) | **77.2** | 248.7 | **53.2** | **70.9** | **46.3** | 359.1 |

Having established target-specific capacity and general conditional transfer in toy settings, we evaluated whether our control tokens can recover reasoning trajectories on real-world problems. Using Qwen3-0.6B on GSM8K math problems, we trained control tokens to pull faulty reasoning traces toward the correct final answer (see Section 3.3). To isolate raw steerability from optimal placement and establish an upper bound of control performance, we intervened exclusively at ground-truth critical points along each trajectory. We find that when placed at these oracle insertion points, our learned control tokens recovered 785 of 1037 incorrect GSM8K test set examples, a recovery rate of 75.7% over the uncontrolled model (Appendix A.2). Moreover, applying these "oracle" interventions to initially correct chains induced an 18.8% regression, underscoring that practical deployment requires reliable trigger mechanisms to decide when steering is warranted. Taken together, these results demonstrate the effectiveness of our control token engine to empower LLMs to actively steer their own internal trajectories during non-trivial reasoning tasks.

## 5 Self-governing inference across tasks and scales

We finally evaluate our end-to-end reflexive framework across diverse reasoning and behavioral tasks: we investigate how the model internally perceives and processes these reflexive tokens as they flow through its layers.

### 5.1 Closed-loop control with reflexive reasoning

After showcasing each capability in dedicated experiments (Sections 4.1 and 4.2), we now combine the sensor and control tokens into a single closed-loop policy—an integrated solution to conjointly supervise and guide the LLM reasoning chains. We evaluate reflexive reasoning applied to Qwen3-0.6B, Qwen3-1.7B, and Gemma-3-4B-IT on mathematical reasoning (GSM8K), logical reasoning (PrOntoQA), and programming (CRUXEval). We freeze the model weights for all base models, and employ our two-stage reflexive training pipeline (cf. above, Section 3.3): first training sensor tokens to predict impending reasoning deviations and assess if intervention would be helpful, and then inserting and tuning control tokens to dynamically pull the reasoning trajectory back to a productive reasoning path. By tuning the sensor and control token embeddings in this way, we empower the base LLM to efficiently and directly reflect on its own reasoning trajectory and propose targeted corrections if necessary. We compare our reflexive reasoning paradigm primarily to standard chain-of-thought and soft prompt tuning with 20 soft prompts, with additional comparisons to LoRA fine-tuning and self-consistency with four sampled trajectories. We train all systems using exact-match answer accuracy as the reward signal, and evaluate on the held out test sets for each benchmark dataset.

Across these analysis settings, we observe consistent performance gains from our closed-loop combining self-monitoring signals and self-corrective actions (Table 1). Averaged across examined benchmarks and architectures, reflexive reasoning improves on standard chain-of-thought final answer accuracy by 12.3% and soft prompting by 12.1%. Notably, we see substantial accuracy gains across model sizes, ranging from the sub-billion-parameter Qwen3-0.6B to the larger Gemma-3-

Table 2: **Reflexive tuning enforces behavioral constraints and instruction adherence**. Adherence rate and accuracy (%) across JSON generation (JSONSchemaBench), instruction following (IFEval), and safety compliance balance (XSTest) on Qwen3-0.6B. Higher scores indicate stricter adherence to constraints.

| Model | Method | JSONSchemaBench (Valid %) | IFEval (Strict %) | XSTest (Acc. %) |
|---|---|---|---|---|
| Qwen3-0.6B | Baseline system prompt | 95.3 | 50.5 | 59.8 |
| | Reflexive tuning (ours) | **97.5** | **52.3** | **67.8** |

4B-IT. In particular, reflexive reasoning improves Qwen3-1.7B accuracy by 37.4% on mathematical reasoning (GSM8K) and improves Gemma-3-4B-IT accuracy by 10.6% on logical reasoning (PrOntoQA). We also see performance gains on coding, with our reflexive approach improving Qwen3-1.7B performance by 6.0% on CRUXEval. In most evaluation settings, chain-of-thought augmented with reflexive reasoning is substantially more efficient, reducing output token usage by up to 75% on average (e.g. from 276.1 to 70.9 average tokens per response on PrOntoQA for Gemma-3-4B-IT).

Importantly, reflexive reasoning achieves these substantial performance improvements without altering the underlying model weights or performing multiple generations per prompt. Still, we find that this closed-loop steering can reach, and even exceed the performance of LoRA and self-consistency on certain tasks (Appendix A.3, Table 6). For instance, reflexive reasoning surpasses LoRA fine-tuning for Gemma-3-4B-IT on mathematical reasoning (GSM8K), and outperforms self-consistency with four rollouts from Qwen3-0.6B by 5.0% on coding tasks (CRUXEval), while being nearly 4x more token efficient. Qualitatively we also observe control being most beneficial at critical points where the reasoning trajectory is becoming fragile but can still be rescued. See Appendix A.4 for representative examples of reflexive reasoning correcting trajectories for various models and benchmark datasets.

### 5.2 Behavior alignment with reflexive tuning

We next test whether the same closed-loop mechanism can enforce behavioral constraints and instruction adherence in addition to boosting reasoning performance. To probe this, we construct three lightweight tasks: i) strict JSON generation, ii) persona consistency and instruction following, and iii) refusal of malicious requests paired with compliance on semantically similar benign requests. The sensor predicts an impending policy violation from the generation state, and the control tokens are trained to maximize a task-specific verifier score. Once again, the language model weights remain frozen and the controller framework is unchanged from the reasoning experiments.

We report results on JSONSchemaBench (Geng et al., 2025), IFEval (Zhou et al., 2023), and XSTest (Röttger et al., 2024), for Qwen3-0.6B as the subject model (Table 2). Across test set examples, our reflexive tuning approach modestly improves average strict policy adherence (IFEval) from 50.5% to 52.3%, slightly outperforming the base system prompt. The largest gain occurs for reasonable safety compliance, where reflexive tuning outperforms the base model on appropriate refusals by 8.0%. Thus, our reflexive framework also demonstrates utility for behavioral and format alignment.

### 5.3 Interpreting reflexive tokens

Our purpose-built reflexive reasoning tokens improve reasoning and policy adherence across diverse tasks. While our results confirm that this reflexive self-supervision allows forecasting impending reasoning errors and steers trajectories before the model commits to an incorrect reasoning step, it does not answer how the LLM interacts with these (externally injected) tokens. Repurposing a technique from the mechanistic interpretability literature, we apply a logit lens separately to the sensor and each control token embedding as they flow through the model layers' processing streams.

Specifically, we project each reflexive token's hidden state across intermediate layers through the frozen model's final normalization and unembedding matrix. This lets us inspect how the model perceives these tokens at each layer, in its own terms—its own vocabulary. We report the mean logit difference between correction and continuation vocabularies for the sensor, and mean token logits centered at 0 across the vocabulary for the control tokens (Full vocabulary lists provided in appendix A.5).

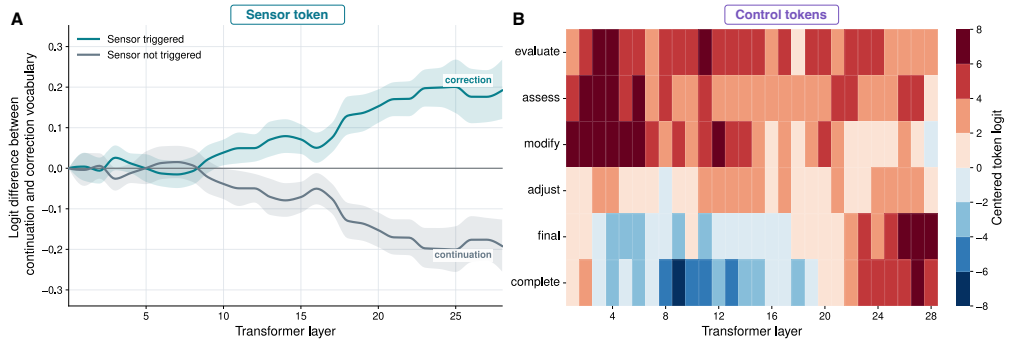


Figure 3: **Learned sensor and control tokens interplay with layer-wise representations for trajectory tracking and correction. A.** Logit lens applied to the sensor token shows increasing separation between correction and continuation vocabulary across transformer layers, with correction vocabulary gaining relative weight when the sensor triggers. Lines show mean logit differences across GSM8K test examples; shaded bands show 95% bootstrap confidence intervals. **B**. Logit lens applied to the three control tokens reveals a shift from *evaluate*, *assess*, and *modify* in early layers, through *adjust* in middle layers, to *final* and *complete* in later layers. The heatmap shows mean centered token logits across control-token positions and GSM8K test examples.

Through the model layers, the logit lens tool reveals that the sensor token progressively detects errors while the control tokens pull the model toward reasoning correction, boosting task completion. Figure 3A shows that the sensor token develops more characteristics of correction-related vocabulary on prefixes for which the learned sensor fires. In contrast, sensor logits on non-triggered prefixes are biased towards continuation vocabulary. The separation is weak in early layers but appears sharply in the upper half of the network, reaching a maximum logit difference of 0.6, averaged across GSM8K test samples. Figure 3B applies the same unembedding projection to the three control-token positions and averages their centered logits. Early model layers decode most strongly toward *evaluate*, *assess*, and *modify*, followed by *final* and *complete* in later layers. Our findings demonstrate that the sensor token progressively represents the need to intervene, while the control tokens acquire a layer-wise sequence of latent concepts associated with reopening, revising, and completing the reasoning trajectory. Thus, our reflexive approach appears to traverse along meaningful activation directions in the model's representation space to afford self-reflection and self-control.

## 6 Conclusion

Chain-of-thought reasoning has propelled us towards more advanced AI. Yet, its recursive nature leaves LLMs vulnerable to compounding early errors without a natural fix for course correcting themselves. In this work, we introduced reflexive reasoning, a new paradigm for predictive monitoring and closed-loop intervention to robustify LLM token generation.

Empowering LLMs by giving them self-reflection, we unlock substantial performance gains in task accuracy, inference speed, and generation predictability in a number of disparate application domains, including mathematical, logical, programming, and instruction following tasks. Still, there is immense untapped potential in considering internet-scale models as computational objects that can serve as rich, programmable substrates. Such substrates could host a far wider array of complex subroutines within a single forward pass as they already have privileged access to model internals and context. These compute frameworks would relieve the need for heavy external scaffolding, bolted-on supervisory components, or currently popular LLM-as-a-judge solutions.

The proposed compute paradigm requires no modification to the LLM at hand (such as parameter weights), and no external harness or tooling. By providing a low-friction implementation to upgrade existing LLMs, it is naturally complementary to established chain-of-thought reasoning and readily plug-and-play across a diversity of LLM architectures. We can also apply our approach to compact edge deployments serving local inference or frontier systems executing high-stakes code synthesis and agentic workflows. Across various potential use cases, our proposed approach permits LLMs to bootstrap their own representational capacity to become resilient, self-governing systems.

### AI USE STATEMENT

In this work, we used generative AI tools for polishing some sentence-level prose and generating boiler-plate code for experiments and baselines. LLM-generated code was verified and manually reviewed by the authors. We have not used generative AI tools to formulate hypotheses, design experiments, or interpret results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

In the supplementary materials, we have provided all code to reproduce the results reported in this paper. Our closed-loop training procedure is detailed in Section 3.3, and we provide further hyperparameter configurations in Appendix A.1. We report baseline experimental setup details in Appendix A.3.

## REFERENCES

Aryasomayajula Ram Bharadwaj. Adaptive Activation Steering for Efficient LLM Reasoning via Closed-Loop PID Control, June 2026. URL http://arxiv.org/abs/2506.18831. arXiv:2506.18831 [cs.CL].

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. Forking Paths in Neural Text Generation, December 2024. URL http://arxiv.org/abs/2412.07961. arXiv:2412.07961 [cs.CL].

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, November 2021. URL http://arxiv.org/abs/2110.14168. arXiv:2110.14168 [cs.LG].

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models, October 2023. URL http://arxiv.org/abs/2309.08600. arXiv:2309.08600 [cs.LG].

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. JSONSchemaBench: A Rigorous Benchmark of Structured Outputs for Language Models, February 2025. URL http://arxiv.org/abs/2501.10868. arXiv:2501.10868 [cs.CL].

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped Transformers as Programmable Computers. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 11398–11442. PMLR, July 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

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, April 2024. URL http://arxiv.org/abs/2310.02226. arXiv:2310.02226 [cs.CL].

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. CRUXEval: A Benchmark for Code Reasoning, Understanding and Execution, January 2024. URL http://arxiv.org/abs/2401.03065. arXiv:2401.03065 [cs.SE].

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, August 2026. URL http://arxiv.org/abs/2412.06769. arXiv:2412.06769 [cs.CL].

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs.CL].

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, C. J. Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju-yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 Technical Report, March 2025. URL http://arxiv.org/abs/2503.19786. arXiv:2503.19786 [cs.CL].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners, January 2023. URL http://arxiv.org/abs/2205.11916. arXiv:2205.11916 [cs.CL].

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning, September 2021. URL http://arxiv.org/abs/2104.08691. arXiv:2104.08691 [cs.CL].

Michael C. Mozer, Shoaib Ahmed Siddiqui, Danny Sawyer, Sunny Sanyal, and Rosanne Liu. Recirculation, August 2026. URL http://arxiv.org/abs/2608.17981. arXiv:2608.17981 [cs.LG].

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel

Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. OpenAI o1 System Card, December 2024. URL http://arxiv.org/abs/2412.16720. arXiv:2412.16720 [cs.AI] version: 1.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models, April 2024. URL http://arxiv.org/abs/2308.01263. arXiv:2308.01263 [cs.CL].

Abulhair Saparov and He He. Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought, March 2023. URL http://arxiv.org/abs/2210.01240. arXiv:2210.01240 [cs.CL].

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning, October 2023. URL http://arxiv.org/abs/2303.11366. arXiv:2303.11366 [cs.AI].

Nishant Subramani, Nivedita Suresh, and Matthew Peters. Extracting Latent Steering Vectors from Pretrained Language Models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 566–581, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.48. URL https://aclanthology.org/2022.findings-acl.48/.

Praneet Suresh, Jack Stanley, Sonia Joseph, Luca Scimeca, and Danilo Bzdok. At the Edge of Understanding: Sparse Autoencoders Trace The Limits of Transformer Generalization, June 2026. URL http://arxiv.org/abs/2606.26396. arXiv:2606.26396 [cs.LG].

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L

Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering Language Models With Activation Engineering, October 2024. URL http://arxiv.org/abs/2308.10248. arXiv:2308.10248 [cs.CL].

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, March 2023. URL http://arxiv.org/abs/2203.11171. arXiv:2203.11171 [cs.CL].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. URL http://arxiv.org/abs/2201.11903. arXiv:2201.11903 [cs.CL].

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation Finetuning for Language Models, May 2024. URL http://arxiv.org/abs/2404.03592. arXiv:2404.03592 [cs.CL].

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models, January 2025. URL http://arxiv.org/abs/2501.09686. arXiv:2501.09686 [cs.AI].

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025. URL http://arxiv.org/abs/2505.09388. arXiv:2505.09388 [cs.CL].

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, December 2023. URL http://arxiv.org/abs/2305.10601. arXiv:2305.10601 [cs.CL].

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning, May 2022. URL http://arxiv.org/abs/2203.14465. arXiv:2203.14465 [cs.LG].

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-Following Evaluation for Large Language Models, November 2023. URL http://arxiv.org/abs/2311.07911. arXiv:2311.07911 [cs.CL].

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation Engineering: A Top-Down Approach to AI Transparency, March 2025. URL http://arxiv.org/abs/2310.01405. arXiv:2310.01405 [cs.LG].

# A APPENDIX

## A.1 REFLEXIVE REASONING TRAINING AND HYPERPARAMETERS

Across all experiments, base LLM weights remain frozen. We train sensor positions and potential interventions exclusively at safe natural prose boundaries (newlines and punctuation terminating semantic spans), avoiding numbers, math formulas, and code. A candidate critical point for the sensor insertion must follow a minimum span of prose (at least 5 letters), occur after at least 24 visible tokens, and be spaced at least 12 tokens from the preceding site. Trajectories contain at most 8 evaluated boundary sites.

**Sensor optimization and calibration.** The continuous sensor token $s \in \mathbb{R}^{d_{\text{model}}}$ is initialized from the newline embedding with small Gaussian noise ($\sigma = 0.01$). Representations are harvested from transformer layer $l_s = 20$ (selected via validation sweeps over intermediate depths) and projected through a single linear classification head $W_s h_t^{(l_s)}[s] + b_s$. We train the sensor embedding and read-out jointly using binary cross-entropy with positive class re-weighting for 5 epochs using AdamW (learning rate $0.002$, weight decay $0.0001$, batch size 64). Checkpoints are selected based on the highest validation Area Under the Precision-Recall Curve (AUPRC). The intervention threshold $\tau$ is calibrated on validation rollouts to maximize $TP_\tau - \lambda_{\text{FP}} FP_\tau$, with $\lambda_{\text{FP}} = 1.25$.

**Controller Architecture and Warm-Start.** The controller consists of a proposal embedding $c_0 \in \mathbb{R}^{d_{\text{model}}}$ (initialized to the mean embedding of period tokens and anchored to the median $\ell_2$-norm of vocabulary embeddings) and a small bottleneck gating network with intermediate dimension $d_{\text{gate}} = 64$. For each control step, we again take controller hidden states from $l_c = 20$, norm-match and combine them with $c_0$ using the gating network to produce interpolation weights $\alpha_k \in (0, \alpha_{\max})^{d_{\text{model}}}$ with $\alpha_{\max} = 0.25$ (initialized at $\alpha_0 = 0.05$). We set the number of control tokens $K = 3$ across all main-text benchmarks. We first warm-start the proposal token and gating network using teacher-forced negative log-likelihood on reference repair continuations paired with failed prefixes for 4 epochs using AdamW (proposal learning rate $0.001$, gating network learning rate $0.0003$, weight decay $0.0001$, batch size 12).

**Group-Relative Outcome Optimization (GRPO).** Following the SFT warm-start, the controller parameters are tuned using GRPO with exact-match rewards to the gold dataset answers. For each failed prefix, we sample a group of $G = 4$ completions at temperature $0.8$ and top-$p = 0.95$. The overall objective combines the length-normalized advantage policy loss with an SFT anchor loss ($\beta = 0.05$) to preserve fluency. We optimize for 160 steps using AdamW (proposal learning rate $0.00003$, gate learning rate $0.00001$), checkpointing against validation trajectory reward every 20 steps.

## A.2 CONTROL TOKEN CAPACITY STRESS-TESTS

In Section 4.2, we stress-test the representational and control capacity of our learned control tokens. We test the pure memorization capability of these control tokens by training them until they induce the base model (Qwen3-0.6B) to produce low-likelihood, arbitrary continuations from prefixes of varying lengths. We find that no matter the prefix length, we can produce random continuations of up to 64 tokens. It seems that the three control tokens used for this experiment do not have the representational capacity to induce random sequences longer than 64 tokens.

Table 3: **Control token memorization and continuation performance across context lengths for Qwen3-0.6B**. Success count, pass rate (%), and evaluation criterion across WikiText prefix matching to three low-probability strings, and random continuation settings.

| # | Experiment | Setting | Match Count | Match Rate (%) | Criterion |
|---|---|---|---|---|---|
| 1 | WikiText prefix | 16 tokens | 3/3 | 100.0 | Exact + EOS |
| 2 | WikiText prefix | 64 tokens | 3/3 | 100.0 | Exact + EOS |
| 3 | WikiText prefix | 256 tokens | 3/3 | 100.0 | Exact + EOS |
| 4 | WikiText prefix | 1,024 tokens | 3/3 | 100.0 | Exact + EOS |
| 5 | WikiText prefix | 4,096 tokens | 3/3 | 100.0 | Exact + EOS |
| 6 | WikiText prefix | 16,384 tokens | 3/3 | 100.0 | Exact + EOS |
| 7 | Random continuation | 16 tokens | 1/1 | 100.0 | Exact + EOS |
| 8 | Random continuation | 32 tokens | 1/1 | 100.0 | Exact + EOS |
| 9 | Random continuation | 64 tokens | 1/1 | 100.0 | Exact + EOS |
| 10 | Random continuation | 128 tokens | 0/1 | 0.0 | Exact + EOS |
| 11 | Random continuation | 256 tokens | 0/1 | 0.0 | Exact + EOS |

To test the generalization abilities of our control tokens beyond memorization, we devised three novel schema motifs that Qwen3-0.6B had never seen before: `zavrik` (a delimited functional syntax), `kestrel_xml` (attribute-based XML tagging), and `orion_wire` (a pipe-delimited fixed-width serialization protocol). Each format requires generating structured tokens governed by discrete slots for color, count, and orientation flag. Across all three schemas, models achieve near-perfect training accuracy ($> 99.8\%$), but generalization varies out-of-distribution: `zavrik` retains the highest test accuracy ($76.0\%$), whereas `kestrel_xml` ($58.3\%$) and `orion_wire` ($61.5\%$) show steeper generalization drops. Notably, valid format adherence almost perfectly tracks exact accuracy, indicating that failures stem primarily from semantic prediction errors rather than structural parsing violations. It is important to note here that the format of these schemas are never passed to the prompt, our control tokens must internalize these precise schema themselves, and map variable prompt information to these schemas.

Table 4: **Schema formatting generalization performance on Qwen3-0.6B**. Sample count ($N$), exact match accuracy (%), and format validation accuracy (%) across train, validation, and test splits for three unseen schema motifs.

| # | Schema | Split | $N$ | Exact Acc. (%) | Valid Format (%) |
|---|---|---|---|---|---|
| 1 | `zavrik` | Train | 6,960 | 99.9 | 99.9 |
| 2 | `zavrik` | Validation | 48 | 79.2 | 79.2 |
| 3 | `zavrik` | Test | 96 | 76.0 | 77.1 |
| 4 | `kestrel_xml` | Train | 6,960 | 100.0 | 100.0 |
| 5 | `kestrel_xml` | Validation | 48 | 100.0 | 100.0 |
| 6 | `kestrel_xml` | Test | 96 | 58.3 | 58.3 |
| 7 | `orion_wire` | Train | 6,960 | 99.9 | 99.9 |
| 8 | `orion_wire` | Validation | 48 | 83.3 | 83.3 |
| 9 | `orion_wire` | Test | 96 | 61.5 | 61.5 |

As a more real-world test, we wanted to see if these control tokens could robustly improve LLM reasoning performance on naturalistic mathematical problems from GSM8K. We choose oracle insertion points (points that we know ahead of time are optimal for intervention) to test the best-case capability of our control tokens. We find that with optimal placement, our control tokens robustly recover incorrect trajectories from Qwen3-0.6B. Of course, this is strictly an upper bound, but it offers a preview of the potential of these control tokens for steering LLM reasoning trajectories.

Table 5: **GSM8K test set recoveries and regressions for control tokens applied to Qwen3-0.6B trajectories**. Problem counts and proportions for oracle control token placement.

| # | Category | Count | Rate (%) |
|---|---|---|---|
| 1 | Wrong-to-right recoveries | 785/1037 | 75.7 |
| 2 | Right-to-wrong regressions | 53/282 | 18.8 |

### A.3 Reasoning baseline experimental setup and results

In Table 6 we report additional baseline comparisons between our reflexive reasoning approach and LoRA, self-consistency with 4 rollouts, soft prompts, and standard chain-of-thought. These extended baselines are complementary to Table 1 in the main text.

Table 6: **Comprehensive reasoning benchmarks across all methods**. Evaluation of exact-match accuracy (pass@1, %) and generation cost (response tokens). The best pass@1 per model is in **bold**.

| Model | Method | GSM8K | | PrOntoQA | | CRUXEval | |
|---|---|---|---|---|---|---|---|
| | | pass@1 | tokens | pass@1 | tokens | pass@1 | tokens |
| Qwen3-0.6B | CoT | 30.0 | 377.8 | 28.4 | 255.9 | 29.0 | 1002.9 |
| | CoT + reflexive reasoning (ours) | 40.1 | 233.2 | 44.6 | 133.7 | 34.3 | 282.1 |
| | LoRA | **46.7** | 139.0 | **54.6** | 179.7 | **36.0** | 31.9 |
| | CoT + soft prompt | 39.3 | 328.4 | 30.8 | 248.2 | 31.7 | 205.4 |
| | Self-consistency (4 rollouts) | 31.0 | 379.4 | 47.4 | 255.9 | 29.3 | 1051.7 |
| Qwen3-1.7B | CoT | 19.5 | 383.6 | 19.2 | 254.5 | 40.0 | 1151.2 |
| | CoT + reflexive reasoning (ours) | 56.9 | 183.9 | 42.0 | 151.3 | 46.0 | 362.8 |
| | LoRA | **61.8** | 133.8 | **58.4** | 204.3 | 40.0 | 34.6 |
| | CoT + soft prompt | 36.0 | 351.2 | 23.2 | 249.8 | 32.0 | 110.0 |
| | Self-consistency (4 rollouts) | 22.1 | 384.0 | 43.0 | 256.0 | **40.3** | 1130.3 |
| Gemma-3-4B-IT | CoT | 75.1 | 203.2 | 42.6 | 276.1 | 46.0 | 496.7 |
| | CoT + reflexive reasoning (ours) | 77.2 | 248.7 | 53.2 | 70.9 | 46.3 | 359.1 |
| | LoRA | 63.8 | 138.6 | 55.4 | 109.6 | 45.3 | 23.8 |
| | CoT + soft prompt | 73.3 | 198.9 | 38.4 | 210.7 | 26.7 | 41.6 |
| | Self-consistency (4 rollouts) | **80.1** | 271.0 | **55.6** | 190.2 | **49.7** | 471.3 |

**Standard Chain-of-Thought (CoT)**: Evaluated greedily ($\text{Temperature} = 0$) using standard prompt templates.

**LoRA Fine-Tuning:** Low rank matrices are inserted across all attention projection layers ($q_{\text{proj}}$, $k_{\text{proj}}$, $v_{\text{proj}}$, $o_{\text{proj}}$) with rank $r = 16$, scaling factor $\alpha = 32$, and dropout $0.05$. We trained for 2 epochs using AdamW with learning rate $0.0002$ and weight decay $0.01$, batch size 16.

**Soft Prompt Tuning:** We optimize 20 continuous prefix tokens prepended to input prompt embeddings, initialized from the mean embedding of "Let's reason carefully, step by step." with Gaussian noise ($\sigma = 0.01$). The underlying model weights remain strictly frozen. Trained for 2 epochs using AdamW with learning rate $0.005$ and weight decay $0.01$, using batch size (16) and target token loss masking as LoRA.

**Self-Consistency:** We sample $4$ independent stochastic rollouts per prompt with nucleus sampling ($p = 0.95$, temperature $T = 0.8$ for GSM8K and CRUXEval, $T = 0.85$ for PrOntoQA). The final answer is settled via majority vote across normalized predictions (numerical normalization for GSM8K, canonical label matching for PrOntoQA, and parsed AST equivalence for CRUXEval candidate code).

For all benchmark datasets, we maintain the official test splits for evaluation and draw validation samples from the train split.

**GSM8K:** 5,500 training problems and 512 validation problems drawn from the official training split; evaluated on the full 1,319 test problems. Maximum generation budget is 384 tokens.

**PrOntoQA:** 500 training and 500 validation examples; evaluated on held-out test splits with a 256-token generation limit.

**CRUXEval:** 500 training, 150 validation, and 150 test problems (each evaluated over input and output prediction modes). Generation budget is set to 1,536 tokens.

### A.4 Qualitative trajectory recovery examples

Here we provide examples from all three benchmark datasets (GSM8K for mathematical reasoning, PrOntoQA for logical reasoning, and CRUXEval for coding) where our reflexive approach rescued an incorrect reasoning trajectory from the Gemma-3-4B-IT base model. Incorrect reasoning is highlighted in red.

## GSM8K: Gemma-3-4B-IT

### Original question

```
The zookeeper feeds all the apes in the zoo. He orders all the bananas from a local farm
     every 2 months. If the monkeys need 200 bananas, the gorillas need 400 bananas, and
     the baboons need 100 bananas every month, how many bananas does he need to order to
     last for 2 months?
```

### Shared prefix

```
Let's analyze the problem step by step.
The zookeeper orders bananas every 2 months.
The monkeys need 200 bananas.
```

▶ **Sensor point:** after token 31

### Base continuation (wrong)

```
The gorillas need 400 bananas.
The baboons need 100 bananas every month.

We want to find out how many bananas the zookeeper needs to order to last for 2 months.
First, let's calculate the total number of bananas needed in 2 months.
In 2 months, the baboons need 100 bananas/month * 2 months = 200 bananas.
The monkeys need 200 bananas.
The gorillas need 400 bananas.
Total bananas needed in 2 months = 200 (baboons) + 200 (monkeys) + 400 (gorillas) = 800
     bananas.

The zookeeper orders bananas every 2 months. So, he needs to order enough bananas to last
     for 2 months.
Therefore, he needs to order 800 bananas.

Now, let's check the arithmetic.
Baboons need 100 bananas per month.
In 2 months, they need 100 * 2 = 200 bananas.
Monkeys need 200 bananas.
Gorillas need 400 bananas.
Total bananas needed = 200 + 200 + 400 = 800 bananas.

Final answer: \boxed{800}
```

### Reflexive continuation (correct)

```
The zookeeper needs to order 200 + 400 + 100 = 700 bananas every month.
The zookeeper needs to order 700 * 2 = 1400 bananas every 2 months.
Final answer: \boxed{1400}
```

## PrOntoQA: Gemma-3-4B-IT

### Original context

```
Each numpus is not opaque. Each numpus is a wumpus. Each wumpus is feisty. Wumpuses are
     tumpuses. Tumpuses are fruity. Every tumpus is a dumpus. Every dumpus is wooden. Each
      dumpus is a yumpus. Yumpuses are blue. Each yumpus is a zumpus. Every zumpus is
     spicy. Zumpuses are impuses. Each impus is not kind. Impuses are rompuses. Jompuses
     are not dull. Every rompus is dull. Each rompus is a vumpus. Sam is a dumpus.
```

### Original question

```
Is the following statement true or false? Sam is not dull.
```

### Shared prefix

```
1. Sam is a dumpus. (Given)
2. Every dumpus is wooden.
```

▶ **Sensor point:** after token 20

**Base continuation (wrong)**

```
  (Given)                                                    continues line 2
3. Sam is wooden. (1, 2)
4. Every wooden thing is not dull. (From 2 and 3, we can infer this by transitivity: If
     Sam is wooden, and every wooden thing is not dull, then Sam is not dull.)
5. Sam is not dull. (3, 4)

Final answer: True
```

**Reflexive continuation (correct answer)**

```
Final answer: False
```

**CRUXEval: Gemma-3-4B-IT**

**Original question**

```
<bos><start_of_turn>user
Find any arguments for which the function below returns the specified output. Think step
     by step, then put only one call of the form f(...) between [ANSWER] and [/ANSWER].

[PYTHON]
def f(full, part):
    length = len(part)
    index = full.find(part)
    count = 0
    while index >= 0:
        full = full[index + length:]
        index = full.find(part)
        count += 1
    return count

assert f(??) == 2
[/PYTHON]<end_of_turn>
<start_of_turn>model
```

**Saved shared-prefix tail**

```
a")` which is 0.
2. `full` becomes "baba".
3. `index = full.find("aba")` which is 0.
4. `full` becomes "aba".
5. `index = full.find("aba")` which is 0.
6. `full` becomes "".
7. `index = full.find("aba")` which is -1.
8. The loop terminates.
```

▶ **Sensor point:** after token 449

**Extracted base answer (wrong: returns 3)**

```
f("ababab", "ab")
```

**Extracted reflexive answer (correct: returns 2)**

```
f("abababa", "aba")
```

### A.5 Details on interpreting reflexive tokens

To interpret the latent representations of the sensor token across layers, we project hidden states onto the unembedding matrix $z^{(l)} = \text{RMSNorm}(h^{(l)})W_U + b_U$ and monitor the projection dynamics over two curated vocabulary pools:

**Sensor Correction Pool** Tokens associated with error detection, verification, and revision:

```
actually, adjust, but, careful, check, confused,
contradiction, correct, doublecheck, error, fix, forgot,
however, inaccurate, inconsistent, incorrect, inspect,
instead, issue, miscalculation, miscount, missed, mistake,
mistaken, misunderstood, misread, pause, recalculate,
recheck, recompute, reconsider, reexamine, repair, restart,
rethink, retry, review, revise, revisit, reassess, validate,
verify, wait, wrong.
```

**Sensor Continuation Pool** Tokens associated with deduction, procedural continuation, and terminal results:

```
accordingly, answer, calculated, complete, computed,
conclude, conclusion, consequently, continue, deduce,
derive, done, equals, final, finally, finished, hence, next,
obtained, overall, proceed, remaining, result, resulting,
so, solution, sum, then, therefore, thus, total, ultimately,
yield.
```