Enhancing Multilingual Reasoning Alignment through Progressive Code-Switching

Anonymous ACL submission

Abstract

Large reasoning models exhibit a strong English bias: when answering multilingual questions, they often default to reasoning and responding in English, and their performance drops sharply when when constrained to reason in the target language. This limitation largely stems from the scarcity of high-quality multilingual reasoning data during both pre-training and post-training. An efficient alternative is to transfer stronger English reasoning ability to target languages. To this end, we propose PCS, a framework that enables such transfer via progressive code-switching. PCS trains the model on code-switched reasoning traces that mix English and the target language, and gradually shifts it from English reasoning to fully target-language reasoning through a curriculum over code-switching ratios. PCS does not require distilled target-language reasoning traces from stronger LRMs or supervision from a stronger judge model. Experiments on multiple benchmarks show that PCS enables effective reasoning transfer and substantially narrows the performance gap between reasoning in the target language and reasoning in English. Further analysis shows that PCS yields more language-consistent reasoning while maintaining stronger performance.

1 Introduction

Large Reasoning Models (LRMs) such as DeepSeek-R1 (Guo et al., 2025), Qwen3 (Yang et al., 2025), and OpenAI-o1 (OpenAI et al., 2024) have achieved strong performance on math and code reasoning. Reinforcement Learning with Verified Rewards (RLVR) improves such capabilities through scaling, and the long chain-of-thought reasoning steps strengthen problem-solving and enhance interpretability.

Despite these advances, multilingual reasoning—defined as the capability to process and solve mathematical problems in languages other than English—remains a persistent challenge. When prompted with non-English questions, LRMs often still **reason in English** (Wang et al., 2025a). This mismatch between the user's language and the model's reasoning process can undermine comprehensibility and trustworthiness.

Existing approaches to enforce **input-output** language matching include Prompt Control (Tam et al., 2025), Prefix Control (Qi et al., 2025a), supervised fine-tuning (SFT) on multilingual traces (Luo et al., 2025), and Reinforcement Learning (RL) with language-consistency rewards (Mistral-AI et al., 2025). However, these methods exhibit distinct practical limitations. Prompt Control is often unreliable, while prefix-based constraints, though effective for certain languages, frequently compromise accuracy relative to English reasoning. Supervised Fine-Tuning is constrained by the scarcity of multilingual long-reasoning corpora and tends to introduce repetition artifacts. Furthermore, while RL with language-consistency rewards encourages the model to reason in the question language, performance typically remains inferior to English-based reasoning. Critically, this method often employs coarse response-level language detection (a binary 0/1 signal), which is vulnerable to exploitation. Models may leverage robust reasoning patterns from high-resource languages (e.g., English or Chinese) to maximize accuracy, while merely inserting sufficient target-language tokens to satisfy the language detector, rather than engaging in genuine target-language reasoning. Collectively, these issues underscore the base model's limited multilingual reasoning capacity.

To address these issues, we focus on **transferring** strong English reasoning ability to target languages. Rather than forcing an abrupt switch to fully target-language reasoning, we propose PCS (**P**rogressive **C**ode-**S**witching), which treats code-switched reasoning as an intermediate stage for cross-lingual transfer. Instead of rewriting all

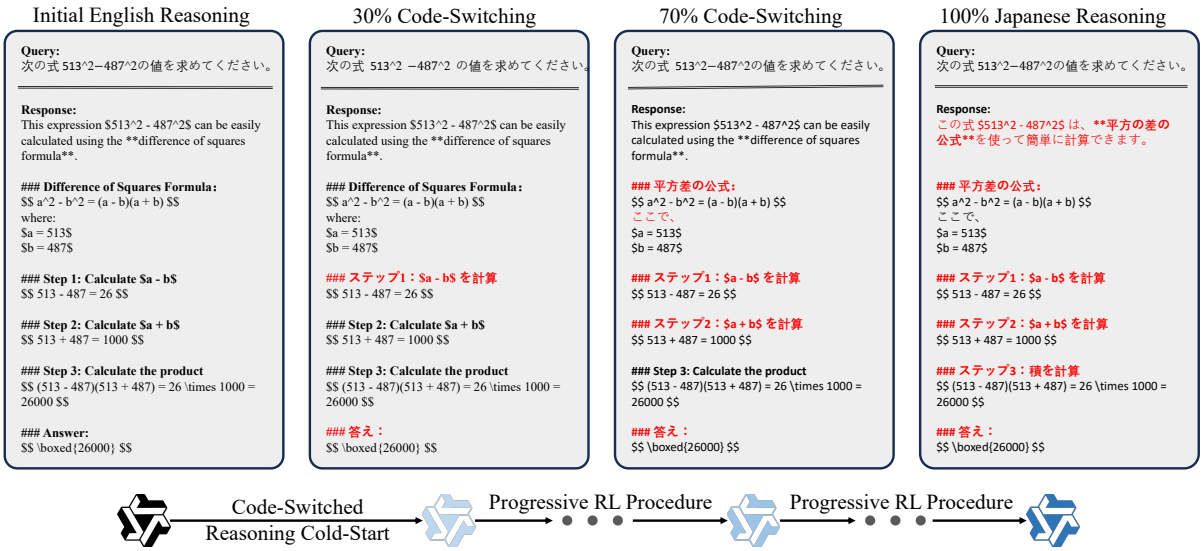

Figure 1: The illustration of Progressive Code-Switching. Starting from a cold-start model, PCS progressively increases the target-language reasoning step ratio until the model performs fully target-language reasoning.

reasoning steps in the target language at once, PCS first converts only a subset of steps, producing mixed English–target-language trajectories that preserve the model's original reasoning behavior while gradually increasing target-language usage.

As shown in Figure 1, PCS implements this idea with a curriculum over step-level language consistency. Starting from a low target ratio, the model is trained with RLVR to produce correct answers while matching the desired degree of code-switching. As training progresses, the target ratio is gradually increased until the model reaches fully target-language reasoning. This progressive design provides a smoother transfer path and avoids the instability and performance degradation commonly observed under direct target-language supervision or hard language-consistency constraints.

PCS does not require distilled target-language reasoning traces or supervision from a stronger teacher or judge model. Experiments across multiple benchmarks and languages show that it consistently improves target-language reasoning consistency while maintaining competitive accuracy, substantially narrowing the gap between reasoning in the target language and in English.

2 Related Work

2.1 Multilingual Reasoning

Large reasoning models (LRMs) such as DeepSeek-R1 (Guo et al., 2025) and OpenAI o1 (OpenAI et al., 2024) achieve strong complex reasoning via RLVR and test-time scaling, yet in multilingual settings, they often exhibit an English bias. DeepSeek-R1 suffers from language mixing and mitigates this with cold-start SFT and language-consistency rewards. However, directly forcing target-language reasoning typically reduces accuracy. Prior work tackles this issue through preference optimization (MAPO (She et al., 2024)), RL with language consistency and cross-lingual thinking alignment (MThinker (Zhang et al., 2026)), and multilingual CoT (Lai and Nissim, 2024), training-free structured reasoning (Qi et al., 2025b), and inference-time representation steering (Li et al., 2025).

2.2 Code-Switching

Code-switching, or language alternation, is a linguistic phenomenon where multilingual speakers use multiple languages within a conversation (Poplack, 1978). Code-switching aids multilingual alignment, as demonstrated by Li et al. (2024), who use input-only code-switching during pre-training. Yoo et al. (2024) introduces CSCL, a curriculum learning method using synthetic code-switching data to enhance multilingual alignment. SynCS (Wang et al., 2025b) analyzes how natural code-switching enhances LLMs' multilingual capabilities and proposes a more flexible and less expensive code-switching synthesis approach. Overall, code-switching data can facilitate cross-language transfer of models.

3 Methods

A key challenge in multilingual reasoning transfer is that directly enforcing target-language reasoning, even with distilled traces from stronger models, often yields limited transfer. We therefore propose Progressive Code-Switching, which uses code-switched reasoning as an intermediate stage between English reasoning and fully target-language reasoning.

3.1 Code-Switched Reasoning

In this paper, we define **code-switched reasoning** as an alternating pattern of English and the target language across reasoning steps within a multi-step response, which means some reasoning steps in the original English reasoning traces are expressed using the corresponding target language. This form of reasoning can encourage the model to exhibit transfer between reasoning modes in the two languages. Although some steps of the high-quality English reasoning are expressed using the target language, the underlying reasoning logic remains the correct and high-quality logic from English; through code-switched reasoning, this logic can gradually be transferred to the target language.

We further define $\mathrm{SLC}(T, L)$, a metric to measure the extent to which the trajectory uses the target language. Given a reasoning trace $T = (s_1, \ldots, s_n)$ and a target language L, we assign each step s_i a language tag via a classifier $c(\cdot) \in \{L, \bar{L}, \varnothing\}$, corresponding to target-language text (L), non-target-language text ($\bar{L}$), and no natural-language text ($\varnothing$; e.g., steps containing only mathematical expressions). We define the step-level language consistency as

$$\mathrm{SLC}(T, L) = \frac{\sum_{i=1}^{n} \mathbf{1}[c(s_i) = L]}{\sum_{i=1}^{n} \mathbf{1}[c(s_i) \in \{L, \bar{L}\}]}, \quad (1)$$

where steps with $c(s_i) = \varnothing$ are excluded from the denominator because they are language-agnostic. If the denominator is zero, we omit this trace when computing batch-level statistics. We use $\mathrm{SLC}(T, L)$ to quantify the extent to which the trajectory uses the target language, and aim to push it toward 100% while preserving task performance.

3.2 Cold-Start

Since current models often default to English on multilingual math problems, we first perform cold-start SFT on the policy model to enable code-switched reasoning. We use newline-delimited lines as a practical approximation of reasoning steps, which is lightweight and stable in RL training. Appendix E further analyzes the potential problems behind this approximation. Given each target-language question with an English reasoning trace, we translate a fixed fraction of reasoning steps (30% in our experiments) into the target language to create a code-switched trace. We then mix these examples with standard English question–answer reasoning data and fine-tune the base model on the combined dataset. This cold-start stage equips the model with an initial ability to produce code-switched reasoning without distilled target-language traces from stronger models.

3.3 Reward Modeling

PCS then conducts RL training based on the cold-start model. Different from conventional response-level language-consistency reward by judging the language of the entire response, we compute the $\mathrm{SLC}(T, L)$ and compare it against a threshold to obtain the language reward. This reward reflects whether the model's current code-switched reasoning behavior matches the desired level specified by our curriculum. Given the threshold τ of $\mathrm{SLC}(T, L)$ at the current optimization step, we first define our reward system:

- **Repetition penalty ($\mathbf{r_{rep}}$):** We detect repetition in the response to force better reasoning quality. $\mathrm{r_{rep}} = 1$ if no repetition, otherwise 0. See Appendix A.3 for more details.
- **Format reward ($\mathbf{r_{fmt}}$):** $\mathrm{r_{fmt}} = 1$ if the output follows the `<think>...</think>` format, otherwise 0.
- **Accuracy reward ($\mathbf{r_{acc}}$):** $\mathrm{r_{acc}} = 1$ if the answer is correct, otherwise 0.
- **Step-level language consistency reward ($\mathbf{r_{SLC}}$):** We apply regular expressions to remove mathematical content. Then use *langdetect*[1] to identify the language of each step. $\mathrm{r_{SLC}} = 1$ if $\mathrm{SLC}(T, L) \geq \tau$, otherwise 0.

As demonstrated in Figure 2, we first apply a hard gate $C = (r_{\text{fmt}} = 1 \wedge r_{\text{rep}} = 1)$; if C is not satisfied, the final reward is set to zero. When C holds, we assign $r_{\text{final}} = 1$ to correct responses and $r_{\text{final}} = 0.1$ otherwise. For correct responses,

[1] https://github.com/Mimino666/langdetect

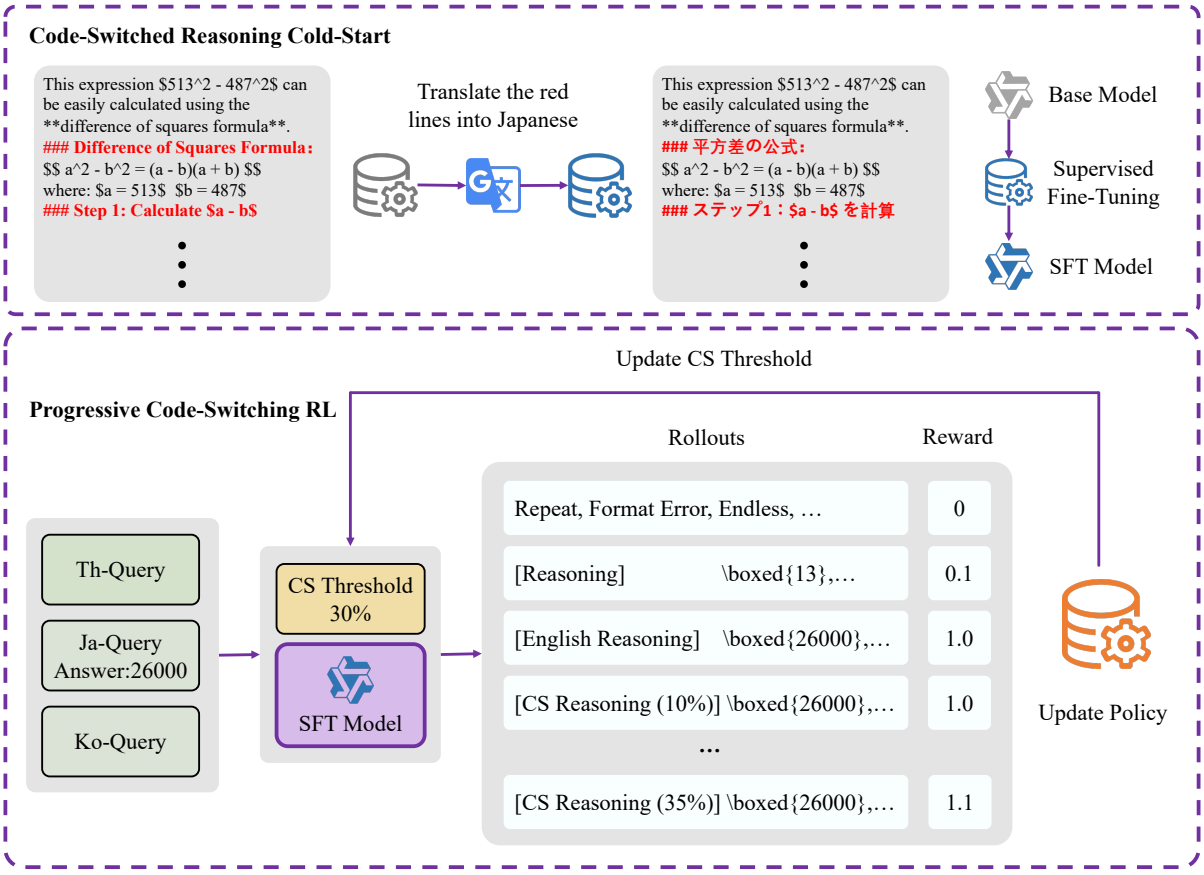

Figure 2: **The Framework of PCS.**

we additionally apply a step-level language consistency bonus: $r_{\text{final}} = 1.1$ if $r_{\text{SLC}} = 1$, and $r_{\text{final}} = 1$ if $r_{\text{SLC}} = 0$.

$$r_{\text{final}} = \begin{cases} 1.1, & \text{if } C \wedge (r_{\text{acc}} = 1) \wedge (r_{\text{SLC}} = 1), \\ 1, & \text{if } C \wedge (r_{\text{acc}} = 1) \wedge (r_{\text{SLC}} = 0), \\ 0.1, & \text{if } C \wedge (r_{\text{acc}} = 0), \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

$$C = (r_{\text{fmt}} = 1 \wedge r_{\text{rep}} = 1). \tag{3}$$

This reward design not only enforces a high-quality reasoning process but also encourages the model to gradually shift its reasoning language while preserving task performance.

3.4 Progressive Code-Switching RL

Although our initial model acquires preliminary code-switched reasoning ability through cold-start training, its code-switching ratio (i.e., $\text{SLC}(T, L)$) remains low and far from fully target-language reasoning. We then employ reinforcement learning with curriculum learning to increase $\text{SLC}(T, L)$.

Based on our reward modeling, we set τ at 10% from the beginning and progressively increase τ to 95% during RL training. As demonstrated in Algorithm 1, we track a batch-level curriculum progress metric, denoted as $\text{Pass@SLC}(\tau)$. It is defined as the ratio of responses achieving $\text{SLC}(T, L) \geq \tau$ to the total number of responses that satisfy the format and correctness constraints. Given an adjustment interval k and a margin $\Delta\tau$, we evaluate $\text{Pass@SLC}(\tau)$ every k RL optimization step and update the threshold by

$$\tau \leftarrow \begin{cases} \min(\tau + \Delta\tau,\ 0.95), & \text{if } \text{Pass@SLC}(\tau) \geq 0.9, \\ \tau, & \text{otherwise}. \end{cases} \tag{4}$$

We set $k = 40$ and $\Delta\tau = 0.1$ in our experiments. When $\text{Pass@SLC}(\tau)$ reaches 0.9, it suggests that the language-shifting reward becomes saturated (and thus less informative) in the current stage. Increasing τ makes the language-consistency reward informative again and corresponds to a harder curriculum, since a higher τ requires a larger proportion of reasoning steps in the target language.

Algorithm 1: Progressive Code-Switching

Input: A cold-start SFT policy model π_{θ_0} with code-switched reasoning; target language L; initial threshold τ_0 (e.g., 0.10); max threshold $\tau_{\max}$ (e.g., 0.95); curriculum step size $\Delta\tau$; check interval k; GRPO group size G.
Output: Model π_θ with target-language reasoning.

Progressive RL with curriculum;
Initialize $\pi_\theta \leftarrow \pi_{\theta_0}$;
Set $\tau \leftarrow \tau_0$;
for $t = 1, 2, \ldots$ **do**
  Set $\theta_{\text{old}} \leftarrow \theta$;
  Sample a minibatch of questions $\{q\} \sim P(\mathcal{Q})$;
  foreach q **do**
    Sample a group of responses $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$;
    Compute scalar rewards $r_i \leftarrow r_{\text{final}}(o_i; \tau, L)$;
  Update π_θ from $\pi_{\theta_{\text{old}}}$ by one GRPO step using $\{(q, \{o_i, r_i\}_{i=1}^{G})\}$;
  if $t \bmod k = 0$ **then** `// curriculum update`
    Compute $\text{Pass@SLC}(\tau)$ on recent samples;
    if $\text{Pass@SLC}(\tau) \geq 0.9$ **then**
      $\tau \leftarrow \min(\tau + \Delta\tau, \tau_{\max})$;

We design a progressively increasing τ to encourage the model to gradually shift its reasoning language for each step, rather than directly assigning a language reward based on $\text{SLC}(T, L)$ (e.g., giving a higher reward for a higher $\text{SLC}(T, L)$). This is motivated by the fact that transferring reasoning capability from English to the target language via code-switched reasoning is typically a slow adaptation process. An overly aggressive strategy, or one lacking an easy-to-hard curriculum, may cause the model to over-optimize language consistency at the expense of the more important goals of cross-lingual transfer and reasoning performance.

As $\text{SLC}(T, L)$ increases, the policy distribution deviates more substantially from that of the SFT model, which in turn leads to larger KL divergence and unstable training. In our experiments, an initial KL constraint limits language-switching efficiency, so we mitigate this issue by resetting the reference model to the current policy whenever the KL loss exceeds an empirical threshold of 0.2, and then applying a KL penalty with a coefficient of 0.001. We apply GRPO to optimize the policy model in our experiments.

4 Experiments

4.1 Experiment Setup

Backbones and Languages We adopt Qwen3-4B-Base and Qwen3-8B-Base as the backbone models. We evaluate PCS on a diverse set of languages—French (fr), Portuguese (pt), Japanese (ja), Korean (ko), and Thai (th)—to examine its effectiveness across typologically different languages.

Data We design consistent settings for different baselines and PCS. Please refer to Appendix A.

Evaluation Details We evaluate on the math subset of MMLU-ProX (Xuan et al., 2025) and MMATH (Luo et al., 2025). We use three metrics: **Step-Level Language Consistency (SLC)**, which measures the proportion of reasoning steps in the target language; **Accuracy (Acc)**, which measures answer correctness; and **SLC&Acc**, the percentage of responses that are both correct and language-consistent (**SLC** $\geqslant 0.9$), which serves as our primary metric. We use **SLC** $\geqslant 0.9$ to tolerate a small amount of language-agnostic or stray non-target tokens while still reflecting practically target-language reasoning. Appendix B provides further analysis of this setting.

We report avg@4 on MMATH, which is harder and more stochastic under long reasoning, and avg@1 on $\text{MMLU-ProX}_{\text{math}}$ following the standard zero-shot setting. For MMATH, we macro-average its four subsets (MATH500, CNMO, AIME2024, and AIME2025) to account for their varying difficulty levels.

Baselines

- **Qwen3-4/8B:** We evaluate the post-trained models of both backbones in thinking mode under the same context limit.
- **Prompt Control:** Wang et al. (2025a) append language-control instructions to Qwen3-4/8B at inference time (see Figure 9).
- **Prefix Control:** Following Qi et al. (2025a), we add "Okay" in the question language after "<think>" for Qwen3-4/8B to steer the reasoning language without parameter updates.
- **SFT:** Fine-tunes the base model on input-output language consistent supervised data distilled by DeepSeek-V3.2-Exp.
- **Naive RL:** Optimizes the SFT model only by response correctness.
- **SoftLC RL:** (Mistral-AI et al., 2025) adds a soft response-level language reward (0.1) to Naive RL for target-language responses.

	FR			PT			JA			KO			TH			ALL-AVG
Methods	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc
MMATH																
Qwen3-4B	0.0	28.7	8.7	0.0	29.6	10.2	0.0	23.6	5.8	0.0	24.3	6.7	0.0	24.8	6.3	0.0
Prompt Control	0.0	32.3	10.9	0.0	31.6	12.3	0.0	24.6	8.1	0.0	21.5	9.3	0.0	23.8	9.1	0.0
Prefix Control	21.9	32.4	88.8	24.5	30.8	95.1	16.6	21.3	96.6	17.1	21.8	91.4	20.4	22.3	94.7	20.1
SFT	2.9	25.3	79.2	3.2	27.0	58.8	7.0	7.7	96.6	9.7	21.0	90.2	11.7	22.2	89.6	6.9
Naive RL	0.0	33.7	3.1	0.0	32.8	1.8	0.0	32.1	0.0	0.0	**34.6**	0.1	0.0	31.5	0.2	0.0
SoftLC RL	14.6	29.3	90.4	19.6	25.8	94.9	23.2	27.0	94.5	17.1	24.3	89.7	26.6	27.8	96.9	20.2
M-Thinker	10.1	**36.5**	80.4	23.7	**36.4**	90.2	29.2	**35.7**	87.6	27.7	32.4	92.0	26.1	**35.1**	81.8	23.4
PCS	**35.1**	36.0	**98.1**	**34.8**	35.6	**97.9**	**31.3**	32.2	**97.7**	**29.5**	30.1	**96.9**	**30.8**	31.1	**97.0**	**32.3**
MMLU-ProX$_{math}$																
Qwen3-4B	0.0	67.4	19.3	0.0	67.0	19.3	0.0	63.8	18.2	0.1	63.8	20.6	0.0	62.4	20.3	0.0
Prompt Control	0.0	70.4	21.6	0.0	70.0	22.2	0.0	66.8	22.6	0.2	69.0	25.5	0.0	58.6	23.9	0.0
Prefix Control	18.8	34.6	89.9	25.4	38.7	93.3	9.9	12.8	94.6	7.0	10.1	91.9	26.5	33.5	92.8	17.5
SFT	7.2	54.5	80.7	13.4	59.8	71.6	0.5	0.6	96.1	5.1	12.4	88.0	19.3	46.4	90.2	9.1
Naive RL	0.0	76.0	4.8	0.0	78.1	3.0	0.1	**76.2**	0.5	0.1	**74.4**	0.8	0.0	**74.5**	1.6	0.1
SoftLC RL	40.4	72.4	89.7	59.2	73.6	93.8	49.3	67.4	92.4	36.1	65.3	88.8	57.9	66.3	95.0	48.6
M-Thinker	29.6	**77.7**	85.4	50.1	76.0	90.7	62.1	75.7	92.2	61.0	73.8	92.2	59.9	71.4	92.1	52.5
PCS	**75.1**	76.9	**98.1**	**77.4**	**78.9**	**97.9**	**68.5**	72.5	**96.9**	**65.9**	70.8	**95.4**	**67.9**	70.6	**96.1**	**71.0**

Table 1: Main results on MMATH and MMLU-ProX$_{math}$ for Qwen3-4B-Base model.

- **M-Thinker:** Zhang et al. (2026) adds cross-lingual thinking alignment rewards using DeepSeek-V3. We reproduce M-Thinker under our experimental settings.

4.2 Experiment Results

Tables 1 and 2 present the main results on MMATH and MMLU-ProX$_{math}$. Across both benchmarks and all five target languages (fr/pt/ja/ko/th), PCS consistently achieves the best overall performance under our primary metric, **SLC&Acc**.

PCS achieves the best balance between correctness and target-language reasoning. Compared with the post-trained model and inference-time control methods, PCS substantially improves **SLC** while maintaining strong **Acc**. Prompt Control provides only weak steering over the reasoning language, whereas Prefix Control enforces target-language reasoning more aggressively but often hurts accuracy. This suggests that directly constraining the reasoning language at inference time is insufficient for effective transfer of multilingual reasoning. In contrast, PCS improves language consistency through training-time transfer, leading to much stronger joint performance.

PCS outperforms both SFT and RL baselines. SFT on distilled multilingual traces improves language consistency, but the quality and scale of supervised multilingual reasoning data limit its gains. We also observe that SFT can introduce severe repetition issues in some languages; for example, in Table 1, Japanese exhibits a substantial performance drop caused by repetition. Naive RL preserves relatively strong **Acc** but typically falls back to English reasoning, resulting in very low **SLC**. SoftLC RL improves **SLC** with response-level language rewards, yet remains clearly behind PCS on **SLC&Acc**. Overall, these results show that neither SFT nor RL with a language consistency reward is sufficient: effective multilingual reasoning transfer requires a gradual shift of the reasoning language.

PCS enables stable language transfer with less supervision. Across all benchmarks, PCS achieves the highest **SLC**, reaching roughly 96–98% on average while preserving competitive accuracy. Compared with M-Thinker, PCS attains comparable overall performance without relying on distilled target-language reasoning traces from a stronger LRM or supervision from a stronger judge model. In our reproduction, M-Thinker is more vulnerable to reward-hacking under response-level language control, leading to lower **SLC** despite relatively strong task performance. In addition, M-Thinker requires online reward evaluation by an external judge model, which introduces non-trivial latency into RL training and can reduce overall hardware utilization. While such overhead may be alleviated through a larger judge-serving setup, this comes at the cost of additional computing resources. By contrast, PCS only uses lightweight language detection, yielding a more resource-efficient transfer process. We analyze these behaviors in more detail in the following sections.

	FR			PT			JA			KO			TH			ALL-AVG
Methods	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc	acc	slc	slc&acc
MMATH																
Qwen3-8B	0.0	27.0	8.1	0.0	25.0	6.7	0.1	23.1	6.3	0.0	22.4	5.9	0.0	22.7	6.4	0.0
Prompt Control	0.0	30.1	10.2	0.2	28.1	9.4	0.0	22.0	7.9	0.0	23.6	8.9	2.7	21.8	17.8	0.6
Prefix Control	22.9	30.6	87.1	26.7	30.2	94.3	19.2	19.8	98.4	21.2	22.4	95.9	20.9	21.0	98.9	22.2
SFT	14.1	32.2	83.1	12.8	34.3	70.5	9.8	9.8	99.0	20.8	23.3	92.5	25.3	25.4	99.4	16.6
Naive RL	0.1	41.6	6.5	0.2	39.5	9.0	0.1	**40.3**	3.7	0.2	**38.8**	10.1	0.1	**39.9**	7.1	0.1
SoftLC RL	30.9	35.0	96.5	32.9	35.1	97.0	30.9	30.9	**99.7**	26.8	28.1	98.2	29.5	29.8	**99.8**	30.2
M-Thinker	33.2	41.6	93.9	36.4	**41.2**	93.7	35.1	36.6	95.7	31.9	37.7	92.7	33.1	36.8	92.1	34.0
PCS	**41.2**	**42.6**	**98.0**	**37.2**	39.2	**97.1**	**36.8**	38.1	98.3	**33.6**	33.8	**98.9**	**36.7**	37.6	95.6	**37.1**
MMLU-ProX$_{\text{math}}$																
Qwen3-8B	0.1	51.7	20.6	0.0	42.4	19.6	0.0	44.1	19.2	0.1	38.3	19.8	0.0	50.9	21.8	0.0
Prompt Control	0.0	68.0	24.7	0.2	67.7	24.3	0.0	65.1	25.0	0.2	67.5	24.3	1.3	58.6	27.1	0.4
Prefix Control	39.5	48.4	90.4	40.7	45.8	94.3	28.9	29.1	98.5	23.3	24.0	98.1	39.5	39.8	99.0	34.4
SFT	41.5	65.6	90.5	30.3	66.3	80.4	0.4	0.4	99.0	8.6	10.0	95.9	54.5	54.7	99.4	27.1
Naive RL	1.7	82.6	16.6	1.3	**82.9**	22.3	0.7	**80.3**	14.1	2.0	**79.9**	30.2	2.0	**80.4**	20.5	1.5
SoftLC RL	75.2	80.5	97.5	74.5	80.6	96.3	74.9	75.0	**99.7**	70.8	74.1	97.8	**75.4**	75.6	**99.8**	74.2
M-Thinker	58.4	79.6	92.6	53.7	80.3	91.3	74.6	76.2	97.6	69.8	78.3	94.7	74.0	77.7	96.7	66.1
PCS	**79.9**	**83.1**	**98.0**	**78.3**	81.2	**97.4**	**76.8**	77.2	99.4	**75.5**	77.4	**98.4**	75.2	75.9	99.5	**77.1**

Table 2: Main results on MMATH and MMLU-ProX$_{\text{math}}$ for Qwen3-8B-Base model.

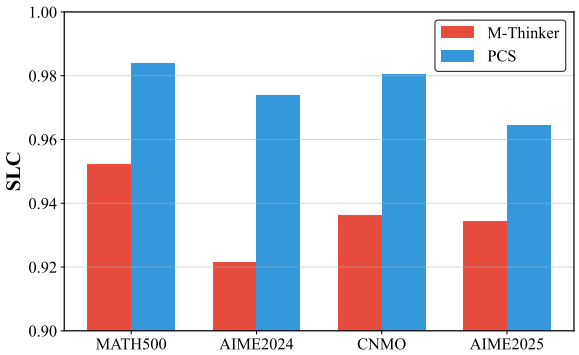

Figure 3: Comparison of $\text{SLC}(T, L)$ between M-Thinker and PCS across different MMATH subsets.

5 Analysis

5.1 Mitigating Reward Hacking of Language Consistency Reward

The SoftLC RL and M-Thinker baselines both use response-level language consistency rewards, which reduces language identification to a binary decision based on off-the-shelf language detection tools. In our experiments, this design is largely effective for SoftLC RL. However, the same response-level constraint becomes ineffective for M-Thinker. When combined with a cross-lingual thinking alignment (CTA) reward provided by a stronger judge model, M-Thinker often produces low-quality "target-language" reasoning that frequently contains Chinese and English words. This suggests that the policy learns to **hack** the response-level language constraint: it exploits stronger English/Chinese reasoning patterns to maximize the CTA reward while inserting enough target-language cues to pass a coarse language detector, rather than genuinely target-language reasoning. We provide a detailed case study of this reward-hacking issue in Appendix D.

Figure 3 compares $\text{SLC}(T, L)$ between M-Thinker and PCS. The reward-hacking problem becomes more severe as the test questions get harder. As shown in Figure 10, M-Thinker relies on key English and Chinese words to solve hard problems, but fails to keep the response in the target language.

5.2 Reasoning-Language Transfer Dynamics

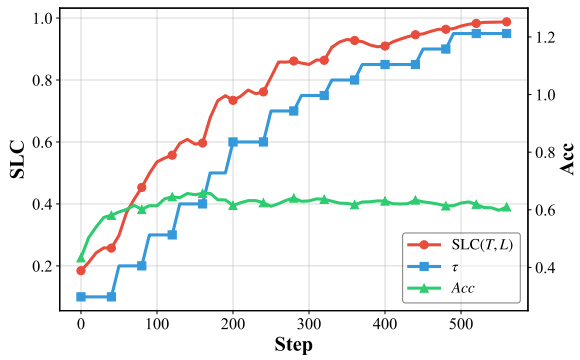

Figure 4: The training curves for $\text{SLC}(T, L)$, τ, and Acc of PCS.

Figure 4 shows the evolution of accuracy, τ, and $\text{SLC}(T, L)$ during training. As τ increases, $\text{SLC}(T, L)$ steadily rises and eventually approaches 100%, indicating nearly full

target-language reasoning. However, increasing $\mathrm{SLC}(T, L)$ also makes optimization harder, since stronger $\mathrm{SLC}(T, L)$ constraints can conflict with task performance: pushing the model toward fewer English reasoning steps often tends to reduce accuracy. To avoid this trade-off, we assign a higher priority to correctness in the reward design. As a result, accuracy remains relatively stable while $\mathrm{SLC}(T, L)$ improves substantially. Overall, PCS successfully shifts the reasoning language with only a small performance loss.

5.3 The Importance of Curriculum

In Algorithm 1, we introduce a curriculum-learning paradigm based on $\mathrm{SLC}(T, L)$. We investigate whether this mechanism is necessary by removing the curriculum schedule and instead shaping the reward to continuously encourage higher $\mathrm{SLC}(T, L)$.

$$r_{\text{final}} = \begin{cases} 1 + 0.1 \cdot \mathrm{SLC}(T, L), & \text{if } C \wedge (r_{\text{acc}} = 1), \\ 0.1, & \text{if } C \wedge (r_{\text{acc}} = 0), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

$$C = (r_{\text{fmt}} = 1 \wedge r_{\text{rep}} = 1). \tag{6}$$

We name this variant **PCS-Dense** because it uses a dense $\mathrm{SLC}(T, L)$ reward. Figure 5 shows the training curves. Without the curriculum on τ, the policy's $\mathrm{SLC}(T, L)$ increases rapidly, but the accuracy underperforms PCS at every step and shows slight improvement over SoftLC RL.

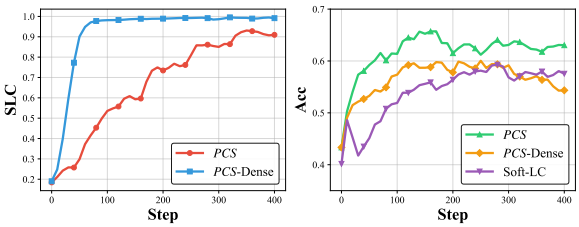

Figure 5: The training curves for $\mathrm{SLC}(T, L)$ and Acc for PCS, PCS-Dense, and SoftLC RL.

We attribute this to the inherently gradual nature of cross-lingual transfer via code-switched reasoning. Because RL is optimized on mini-batches, satisfying the language objective on the current batch does not necessarily generalize to other samples. In contrast, dense rewards push the model to optimize language consistency too aggressively, encouraging premature language switching before the intermediate transfer stage is fully learned and thereby harming reasoning ability.

5.4 Reference Update and KL

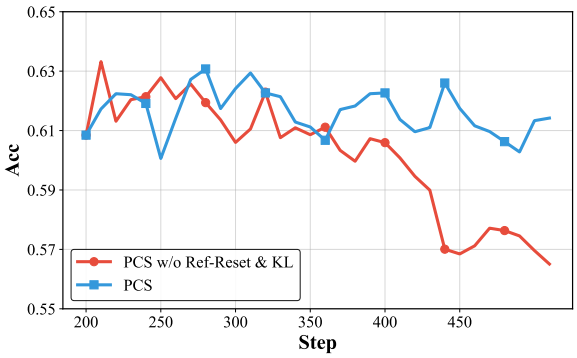

Figure 6: The ablation results for PCS and PCS without reference-model reset and KL regularization.

As described in Section 3.4, we reset the reference model and apply KL regularization when the target $\mathrm{SLC}(T, L)$ become large. The reason is that, in the late stage of training, the model is encouraged to produce almost entirely target-language reasoning, making the policy more prone to drifting away from the earlier policy. If left unconstrained, this drift can destabilize RL optimization and hurt final performance. Reference-model reset mitigates this mismatch by updating the optimization anchor to a policy that is better aligned with the current transfer stage, while KL regularization keeps subsequent updates controlled. As shown in Figure 6, removing these components leads to a substantial performance drop, highlighting their role in stabilizing progressive language transfer.

6 Conclusion

We propose PCS (Progressive Code-Switching), an effective method for transferring English reasoning ability to target languages without requiring distilled target-language reasoning traces from stronger LRMs or supervision from a stronger judge model. By initializing with code-switched reasoning traces and progressively raising a step-level language consistency threshold during RL, PCS smoothly shifts the model toward fully target-language reasoning while maintaining task performance. Experimental results across five languages demonstrate that PCS achieves the best Step-Level Language Consistency and overall SLC&Acc. Furthermore, our analysis confirms that it ensures a stable language transition and improves reasoning quality without reward hacking.

Limitations

Our study has several limitations. First, due to limited computational resources, we primarily focus on standard long-reasoning settings and do not explore substantially longer contexts (e.g., 16K or 32K tokens). It remains an open question whether PCS maintains the same language-shifting stability and capability transfer behavior when the reasoning traces become much longer and require stronger long-context generalization. Second, our experiments are conducted with a relatively small backbone (Qwen3-4B-Base and Qwen3-8B-Base). While PCS is model-agnostic and does not depend on architecture-specific assumptions, scaling to larger models may change the training dynamics and potentially improve the final multilingual reasoning performance. We leave a systematic study of scaling effects—across both model size and context length—to future work.

References

Mara Finkelstein, Isaac Caswell, Tobias Domhan, Jan-Thorsten Peter, Juraj Juraska, Parker Riley, Daniel Deutsch, Geza Kovacs, Cole Dilanni, Colin Cherry, Eleftheria Briakou, Elizabeth Nielsen, Jiaming Luo, Kat Black, Ryan Mullins, Sweta Agrawal, Wenda Xu, Erin Kats, Stephane Jaskiewicz, and 2 others. 2026. Translategemma technical report. *Preprint*, arXiv:2601.09012.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Yunjie Ji, Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. 2025. Am-thinking-v1: Advancing the frontier of reasoning at 32b scale. *Preprint*, arXiv:2505.08311.

Amir Hossein Kargaran, Ali Modarressi, Nafiseh Nikeghbal, Jana Diesner, François Yvon, and Hinrich Schütze. 2025. Mexa: Multilingual evaluation of english-centric llms via cross-lingual alignment. *Preprint*, arXiv:2410.05873.

Huiyuan Lai and Malvina Nissim. 2024. mcot: Multilingual instruction tuning for reasoning consistency in language models. *arXiv preprint arXiv:2406.02301*.

Jiahuan Li, Shujian Huang, Aarron Ching, Xinyu Dai, and Jiajun Chen. 2024. PreAlign: Boosting cross-lingual transfer by early establishment of multilingual alignment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10246–10257, Miami, Florida, USA. Association for Computational Linguistics.

Qiming Li, Xiaocheng Feng, Yixuan Ma, Zekai Ye, Ruihan Chen, Xiachong Feng, and Bing Qin. 2025. Unlocking multilingual reasoning capability of llms and lvlms through representation engineering. *arXiv preprint arXiv:2511.23231*.

Wenyang Luo, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2025. Mmath: A multilingual benchmark for mathematical reasoning. *Preprint*, arXiv:2505.19126.

Mistral-AI, :, Abhinav Rastogi, Albert Q. Jiang, Andy Lo, Gabrielle Berrada, Guillaume Lample, Jason Rute, Joep Barmentlo, Karmesh Yadav, Kartik Khandelwal, Khyathi Raghavi Chandu, Léonard Blier, Lucile Saulnier, Matthieu Dinot, Maxime Darrin, Neha Gupta, Roman Soletskyi, Sagar Vaze, and 82 others. 2025. Magistral. *Preprint*, arXiv:2506.10910.

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, and 244 others. 2024. Openai o1 system card. *Preprint*, arXiv:2412.16720.

Shana Poplack. 1978. Syntactic structure and social function of code switching. *Latino Discourse and Communicative Behavior/Ablex Publishing*.

Jirui Qi, Shan Chen, Zidi Xiong, Raquel Fernández, Danielle Bitterman, and Arianna Bisazza. 2025a. When models reason in your language: Controlling thinking language comes at the cost of accuracy. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20279–20296, Suzhou, China. Association for Computational Linguistics.

Rui Qi, Zhibo Man, Yufeng Chen, Fengran Mo, Jinan Xu, and Kaiyu Huang. 2025b. SoT: Structured-of-thought prompting guides multilingual reasoning in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11024–11039, Suzhou, China. Association for Computational Linguistics.

Shuaijie She, Wei Zou, Shujian Huang, Wenhao Zhu, Xiang Liu, Xiang Geng, and Jiajun Chen. 2024. Mapo: Advancing multilingual reasoning through multilingual alignment-as-preference optimization. *arXiv preprint arXiv:2401.06838*.

Zhi Rui Tam, Cheng-Kuang Wu, Yu Ying Chiu, Chieh-Yen Lin, Yun-Nung Chen, and Hung yi Lee. 2025. Language matters: How do multilingual input and reasoning paths affect large reasoning models? *Preprint*, arXiv:2505.17407.

Yiming Wang, Pei Zhang, Jialong Tang, Haoran Wei, Baosong Yang, Rui Wang, Chenshu Sun, Feitong Sun, Jiran Zhang, Junxuan Wu, Qiqian Cang, Yichang

Zhang, Fei Huang, Junyang Lin, Fei Huang, and Jingren Zhou. 2025a. Polymath: Evaluating mathematical reasoning in multilingual contexts. *Preprint*, arXiv:2504.18428.

Zhijun Wang, Jiahuan Li, Hao Zhou, Rongxiang Weng, Jingang Wang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, and Shujian Huang. 2025b. Investigating and scaling up code-switching for multilingual language model pre-training. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11032–11046, Vienna, Austria. Association for Computational Linguistics.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. *Preprint*, arXiv:2402.10588.

Weihao Xuan, Rui Yang, Heli Qi, Qingcheng Zeng, Yunze Xiao, Aosong Feng, Dairui Liu, Yun Xing, Junjue Wang, Fan Gao, Jinghui Lu, Yuang Jiang, Huitao Li, Xin Li, Kunyu Yu, Ruihai Dong, Shangding Gu, Yuekang Li, Xiaofei Xie, and 13 others. 2025. Mmlu-prox: A multilingual benchmark for advanced large language model evaluation. *Preprint*, arXiv:2503.10497.

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, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Haneul Yoo, Cheonbok Park, Sangdoo Yun, Alice Oh, and Hwaran Lee. 2024. Code-switching curriculum learning for multilingual transfer in llms. *arXiv preprint arXiv:2411.02460*.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Kaiyu Huang, Yufeng Chen, Jinan Xu, and Jie Zhou. 2026. Think natively: Unlocking multilingual reasoning with consistency-enhanced reinforcement learning. *Preprint*, arXiv:2510.07300.

A Detailed Experiment Setup

A.1 Data

For all training-based methods in our experiments, we first perform a cold-start SFT stage on the base models before RL training. For the baseline methods, the cold-start data consist of distilled target-language reasoning traces, whereas for our method, they consist of code-switched reasoning traces.

For training-based baselines, we largely follow the data recipe of Ji et al. (2025). The cold-start SFT data for baselines are trained on 550K samples, including 500K English question–answer reasoning data from Tam et al. (2025) and 50K multilingual question–answer reasoning data distilled by DeepSeek-V3.2-Exp (10K per language). The multilingual queries are randomly sampled from the 500K English data and translated to each language by DeepSeek-V3.2-Exp.

The cold-start SFT data for PCS uses the same overall structure, except that the multilingual question–answer portion is replaced with code-switched reasoning data: we randomly select 30% of the lines in the English reasoning traces and translate them into the target language using TranslateGemma-4B (Finkelstein et al., 2026).

For GRPO, we translate all math RL data from Tam et al. (2025) into five languages using DeepSeek-V3.2-Exp and use them for all training-based methods.

A.2 Training Settings

For SFT on both Qwen3-4B-Base and Qwen3-8B-Base, we train for 2 epochs with a global batch size of 128 and a learning rate of 8×10^{-5}. We pack training samples to a maximum sequence length of 32,768 tokens and set the sequence-parallel size to 4. For GRPO on both backbones, we train for 600 update steps with a learning rate of 5×10^{-6}, a maximum response length of 4,096 tokens, and a training batch size of 256. Training is fully on-policy. For each prompt, we sample 8 candidate responses during training, using a sampling temperature of 0.9. We use the same temperature (0.9) at test time. We follow the training settings of M-thinker (Zhang et al., 2026) on our SFT model and using DeepSeek-V3 as the judge model. All experiments are conducted on 2x8 H100 GPUs.

A.3 Repetition Detection

We employ a multi-level repetition detection mechanism to identify abnormal repetition patterns in model-generated text. Specifically, the detection approach comprises three main components:

- **Line-level exact match detection**: The text is split by line breaks and the frequency of each line's content is counted. A line is identified as repetitive when it repeats beyond a threshold ($\geq$20 times with length $\geq$20 characters, or $\geq$10 times with length $\geq$50 characters).
- **n-gram sequence repetition detection**: Based on suffix arrays and LCP (Longest Common Prefix) algorithms, this method uses divsufsort to construct suffix arrays, computes LCP arrays via the kasai algorithm, and employs sparse tables for efficient range queries

to identify n-grams of length ≥ 2 with repetition counts $\geq$20.

- **Heuristic detection based on n-gram statistics**: This approach counts the frequency of fixed-length n-grams (default $n = 20$). When the highest-frequency n-gram appears ≥ 20 times, it undergoes further validation using the suffix array method for precise verification.

The detection process executes in order of priority, sequentially performing line-level matching detection, n-gram statistical detection, and suffix array verification. This method effectively identifies various forms of text repetition, including both local vocabulary repetition and global structural repetition.

B Sensitivity to the SLC Threshold

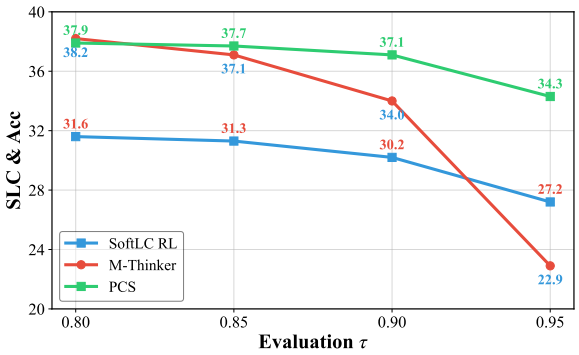

Figure 7: The SLC&Acc results of MMATH across different τ used in evaluation.

We further examine whether our conclusions depend on the choice of the SLC threshold (τ used in evaluation) used in the SLC&Acc metric. Specifically, we evaluate SLC&Acc under thresholds of 0.80, 0.85, 0.90, and 0.95 for SoftLC RL, M-Thinker, and PCS.

As shown in Figure 7, PCS remains substantially more stable as the threshold is tightened. Its SLC&Acc decreases only mildly, from 37.9 at 0.80 to 34.3 at 0.95. SoftLC RL shows a similar but weaker trend, decreasing from 31.6 to 27.2. In contrast, M-Thinker drops much more sharply, from 38.2 at 0.80 to 22.9 at 0.95. Notably, while M-Thinker is competitive under relatively loose thresholds, PCS becomes clearly superior once stricter target-language consistency is required.

These results show that the advantage of PCS is not tied to a particular threshold choice. Instead, PCS is consistently more robust under increasingly strict language-consistency requirements, suggesting that it induces more reliable target-language reasoning rather than merely improving performance near a lenient cutoff. This also supports our use of SLC ≥ 0.9 in the main results, which provides a practical balance between enforcing target-language reasoning and tolerating a small amount of language-agnostic or stray tokens.

C Multilingual Alignment

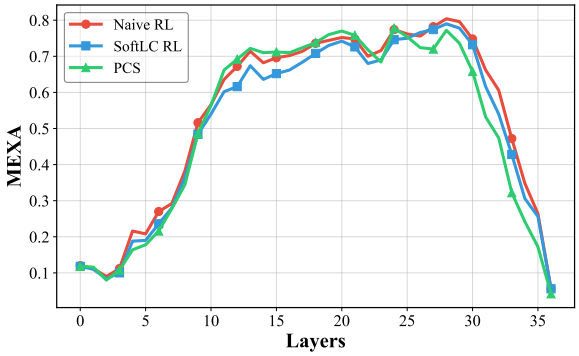

Figure 8: The MEXA multilingual alignment score of PCS and baselines across model layers.

We examine whether PCS aligns its internal representations with English reasoning using the MEXA alignment score (Kargaran et al., 2025). Figure 8 shows the MEXA alignment scores (relative to English) across layers. Prior work (Wendler et al., 2024) suggests that LLMs typically rely on middle layers for reasoning, lower layers for task understanding, and upper layers for language realization. PCS exhibits a level of alignment comparable to Naive RL—and higher than SLC RL—in the middle layers (10–25). Since Naive RL mainly performs reasoning in English, this result suggests that PCS preserves a stronger alignment with English-style reasoning representations during the reasoning process. In contrast, PCS achieves the lowest alignment scores in the upper layers, indicating larger divergence from English in language realization. Overall, PCS maintains alignment with English reasoning while producing target-language reasoning outputs.

D Reward Hacking for Response-Level Language Consistency

As demonstrated in Figures 10 and 11, when prompted with a Thai math question, M-Thinker's response exhibits extensive token-level code-switching between English and Chinese. How-

ever, when we apply the `langdetect` tool to identify the response-level language, it returns "[th:0.9999999495640537]", indicating that the response is entirely Thai, which is clearly not the case. Although M-Thinker produces the correct answer, it relies on English and Chinese reasoning abilities and **hacks** the response-level language detector in a highly unreadable manner.

E Analysis of Potential Line-Level Reward Hacking

Since PCS measures target-language usage at the **line** (step) level, one potential failure mode is that the policy may pack many English reasoning tokens into a single long line, while keeping other lines in the target language, thereby satisfying the Step-Level Language Consistency (SLC) constraint while retaining English-style reasoning.

To examine this possibility, we analyze PCS responses on MMATH. For each response, we tokenize every line using Qwen3's tokenizer. We identify the longest line of each response (measured in tokens), and report the mean of this value over the evaluation set. As shown in Table 3, PCS indeed tends to produce longer longest-lines than the baselines, suggesting that line length could be a confounding factor for line-level language measurement.

We further test whether the longest line contains substantial non-target-language content. For each response, we apply (i) `langdetect` to the longest line to obtain a language label, and compute the language consistency (LC) as the percentage of longest lines classified as the target language; and (ii) we use DeepSeek-V3 as a detector to estimate the code-switching ratio within the longest line. The prompt is detailed at Table E, and we report the mean of the code-switching label (0/1) over the evaluation set. Table 4 shows that the longest lines are overwhelmingly classified as the target language by `langdetect` (LC $\geq$ 96.7% across all languages). Moreover, the estimated intra-line code-switching ratio is very low (0.7–1.6%), indicating that PCS does not appear to hide large amounts of English reasoning inside the longest line.

Overall, while PCS shows a tendency to generate longer lines, we find no evidence that it exploits this behavior to bypass the line-level SLC constraint. These results support the reliability of our line-level language measurement in practice.

Method	FR	PT	JA	KO	TH
Naive RL	178	169	181	173	201
SoftLC RL	184	160	169	139	197
M-Thinker	145	140	145	158	183
PCS	223	220	193	183	195

Table 3: Average token length of the longest line of PCS's response on MMATH.

Statistics	FR	PT	JA	KO	TH
LC(%)	99.1	98.8	98.7	96.7	99.3
CS Ratio(%)	0.7	1.6	1.2	1.2	1.5

Table 4: Language Consistency (LC, detect using langdetect tool) and Code-Switching Ratio (CS Ratio, detect using DeepSeek-V3) of the longest line of PCS's response on MMATH.

Code-Switching Detecting Prompt

{text}

Please determine whether the language of the given text is consistent, meaning it contains only a single language (excluding mathematical content).

Return 0 if it contains only one language, and 1 if it mixes multiple languages. Output exactly 0 or 1 with no additional text or explanations.

Japanese: 日本語で推論し、回答してください。最終結果は\\boxed{}に入れてください。

French: Veuillez raisonner et répondre en français. Veuillez mettre le résultat final dans \\boxed{}.

Portuguese: Por favor, faça o raciocínio e responda em português. Coloque o resultado final em \\boxed{}.

Thai: โปรดใช้ภาษาไทยในการให้เหตุผลและตอบคำถาม และใส่คำตอบสุดท้ายไว้ใน \\boxed{}

Korean: 한국어로 추론하고 답변해 주세요. 최종 결과는 \\boxed{} 안에 넣어 주세요.

Figure 9: Instructions used for Prompt Control

Question:
ถ้า $\\frac{a}{b}$ เป็นความน่าจะเป็นที่ส่วนกลับของจำนวนเต็มคี่บวกที่ถูกเลือกอย่างสุ่มที่น้อยกว่า 2010 จะให้ค่าทศนิยมที่สิ้นสุดลง โดยที่ a และ b เป็นจำนวนเต็มบวกที่ไม่มีตัวหารร่วมมากกว่า 1 ค่า $a+b$ คือเท่าใด?

Response:
เรามีปัญหา: "ถ้า $\frac{a}{b}$ เป็นความน่าจะเป็นที่ส่วนกลับของจำนวนเต็มคี่บวกที่ถูกเลือกอย่างสุ่มที่น้อยกว่า 2010 จะให้ค่าทศนิยมที่สิ้นสุดลง โดยที่ a และ b เป็นจำนวนเต็มบวกที่ไม่มีตัวหารร่วมมากกว่า 1 ค่า $a+b$ คือเท่าใด?"
ขั้นแรก เราต้องทำความเข้าใจคำถาม
"จำนวนเต็มคี่บวกที่ถูกเลือกอย่างสุ่มที่น้อยกว่า 2010"
จำนวนเต็มคี่บวก: 1, 3, 5, ..., up to number less than 2010.
จำนวนเต็มคี่: 形式: $2k+1$, $k \geq 0$
จำนวน: $1, 3, 5, ..., up to n$ โดย $n < 2010$ และ n เป็นจำนวนคี่
จำนวนเท็จ: $(2m+1) < 2010 \implies 2m+1 \leq 2009 \implies m \leq 1004$
จำนวน: m จาก 0 到 1004 → 总数: 1005 numbers? Check: $2*1004 + 1 = 2009$, และ 1 → เป็นจำนวน 1005 numbers.
ตอนนี้: "ส่วนกลับของจำนวน" → reciprocal: $\frac{1}{n}$, โดย n เป็นจำนวนคี่บวกที่เลือก → $n = 2k+1$, $k=0 到 1004$, $n = 1, 3, 5, ..., 2009$
เราต้องการ: "ความน่าจะเป็นที่ส่วนกลับของจำนวน ... จะให้ค่าทศนิยมที่สิ้นสุดลง"
เรารู้ว่า: จำนวน rational $\frac{p}{q}$ (ใน reduced form) มี decimal expansion สิ้นสุดถ้าและแค่ถ้า q มีเฉพาะตัวประกอบเฉพาะเป็น 2 และ 5.
ดังนั้ation: $\frac{1}{n}$ ใน reduced form: $\frac{1}{n'}$ โดย $n' = n / gcd(1,n) = n$ (since gcd(1,n)=1). เงื่อนไข: n' 必须 factors into $2^a * 5^b$.
เพื่อให decimal สิ้นสุด: n' 必须ของรูปแบบ $2^k * 5^m$.
ดังนั้น: n (ใน reduced form) → factors: only 2 และ 5.
และ since n เป็นจำนวนคี่ → n is odd → factors: 不含 2 → 所以 必须 n 自身 เป็น product of powers of 5.
即: $n = 5^t$, for some t ≥ 1.
เพราะถ้า n มีตัวประกอบอื่น → 不满足 condition.
ดังนั้น: ส่วนกลับ: $\frac{1}{n}$ → reduced form: $\frac{1}{n}$ (since gcd(1,n)=1). decimal สิ้นสุด ⇨ n เป็น product of 2 และ 5.但由于 n เป็นคี่ → n = 5^t.
因此 条件: n เป็นกำลังของ 5.
ตอนนี้: "จำนวนเต็มคี่บวกที่เลือก: numbers less than 2010 และคี่ → n = 1, 3, 5, ..., 2009"
列表: n = 1, 3, 5, ..., 2009.
จำนวน: 首项 1, สุดท้าย 2009, 差 2 → 数量: $\frac{2009 - 1}{2} + 1 = \frac{2008}{2} + 1 = 1004 + 1 = 1005$.
ตอนนี้: numbers that are powers of 5: $5^0=1, 5^1=5, 5^2=25, ..., หา t such that 5^t < 2010$.
計算: $5^1=5$, $5^2=25$, $5^3=125$, $5^4=625$, $5^5=3125$ (เกิน 2010).
所以: $n = 1, 5, 25, 125, 625$.
ตรวจสอบ: $5^0=1 < 2010$, $5^1=5 < 2010$, ..., $5^4=625 < 2010$, $5^5=3125 ≥ 2010 → exclude.
ดังนั้น numbers: $1, 5, 25, 125, 625$ → 共 5 numbers.
จำนวนทั้งหมด: 1005.
ความน่าจะเป็น: $\frac{number\ of\ favorable}{total\ number} = \frac{5}{1005} = \frac{1}{201}$.
简化: gcd(1,201)=1? → 已简化。所以 a=1, b=201.
คำถาม: "โดยที่ a และ b เป็นจำนวนเต็มบวกที่ไม่มีตัวหารร่วมมากกว่า 1" → เป็นเศษส่วนอยู่ใน reduced form: $\frac{1}{201}$ → a=1, b=201.
求: $a+b = 1 + 201 = 202$.
คำตอบ: $\boxed{202}$.
\boxed{202}

Figure 10: Case Study of M-Thinker's model for one sampled Thai math question. The language of this response detected by langdetect is: [th:0.9999999495640537]

Question:
ถ้า $\\frac{a}{b}$ เป็นความน่าจะเป็นที่ส่วนกลับของจำนวนเต็มคี่บวกที่ถูกเลือกอย่างสุ่มที่น้อยกว่า 2010 จะให้ค่าทศนิยมที่สิ้นสุดลง โดยที่ a และ b เป็นจำนวนเต็มบวกที่ไม่มีตัวหารร่วมมากกว่า 1 ค่า $a+b$ คือเท่าใด?

Response:
เราต้องการหาความน่าจะเป็นที่ **ส่วนกลับของจำนวนเต็มคี่บวก** ที่น้อยกว่า 2010 จะให้ **ค่าทศนิยมที่สิ้นสุดลง** (terminating decimal)
ขั้นตอนที่ 1: เงื่อนไขของการเป็นทศนิยมสิ้นสุด
เลขเศษส่วน \(
rac{1}{n} \) จะเป็น **ทศนิยมสิ้นสุด** ก็ต่อเมื่อ \(n \) มีตัวประกอบเฉพาะเป็นเพียง 2 และ/หรือ 5 เท่านั้น
กล่าวคือ \(n = 2^a \cdot 5^b \)
แต่ในที่นี้ เราพิจารณาแค่ **จำนวนเต็มคี่บวก** ดังนั้น \(n \) เป็น **คี่** → ไม่มีตัวประกอบ 2 เลย
ดังนั้น \(a = 0 \) ดังนั้น \(n = 5^b \)
และเนื่องจาก \(n \) ต้องเป็น **คี่บวก** และ \(5^b \) ก็เป็นคี่เสมอ (เพราะ 5 เป็นคี่) เราจึงต้องการหาว่า ระหว่างจำนวนเต็มคี่บวกที่น้อยกว่า 2010 มีกี่ตัวที่อยู่ในรูป \(5^b \)
ขั้นตอนที่ 2: หาจำนวนเต็มคี่บวก < 2010 ที่ทำให้ \(
rac{1}{n} \) เป็นทศนิยมสิ้นสุดดังนั้น \(n \) ต้องเป็น **กำลังของ 5** และเป็น **คี่บวก** และ \(n < 2010 \)ลองหา \(5^k < 2010 \)
- \(5^0 = 1 \)
- \(5^1 = 5 \)
- \(5^2 = 25 \)
- \(5^3 = 125 \)
- \(5^4 = 625 \)
- \(5^5 = 3125 \) → มากกว่า 2010 → ไม่เอา
ดังนั้น \(k = 0, 1, 2, 3, 4 \)
ได้ \(n = 1, 5, 25, 125, 625 \)
รวม 5 ตัวแต่ต้องระวัง: \(n \) ต้องเป็น **จำนวนเต็มคี่บวกที่น้อยกว่า 2010** → ทั้งหมดนี้เป็นคี่บวก และ < 2010 → ใช้ได้หมด
ดังนั้น มี 5 ค่าของ \(n \) ที่ทำให้ \(rac{1}{n} \) เป็นทศนิยมสิ้นสุด
ขั้นตอนที่ 3: หาจำนวนทั้งหมดของจำนวนเต็มคี่บวกที่น้อยกว่า 2010
จำนวนเต็มบวกคี่ที่น้อยกว่า 2010 คือ \(1, 3, 5, ..., 2009 \)
ลำดับเลขคณิต: ลำดับแรก = 1, ผลต่าง = 2, พจน์สุดท้าย = 2009
หาจำนวนพจน์:\[n = rac{2009 - 1}{2} + 1 = rac{2008}{2} + 1 = 1004 + 1 = 1005\]
ดังนั้นมี **1005** จำนวนคี่บวกที่น้อยกว่า 2010
ขั้นตอนที่ 4: ความน่าจะเป็น
ความน่าจะเป็นที่สุ่มเลือกจำนวนเต็มคี่บวก \(n < 2010 \) แล้ว \(rac{1}{n} \) เป็นทศนิยมสิ้นสุด คือ\[rac{ ext{จำนวน } n ext{ ที่ทำให้ } rac{1}{n} ext{ สิ้นสุด}}{ ext{จำนวนคี่บวกทั้งหมด}} = rac{5}{1005}\]
ลดรูปเศษส่วน:\[rac{5}{1005} = rac{5 \div 5}{1005 \div 5} = rac{1}{201}\]
ดังนั้น \(rac{a}{b} = rac{1}{201} \), โดย \(\gcd(a,b) = 1 \)
ดังนั้น \(a = 1 \), \(b = 201 \)
\[a + b = 1 + 201 =oxed{202}\]

✅ ตอบ:
\[boxed{202}\]

Figure 11: Case Study of PCS's model for one sampled Thai math question.