Nash CoT: Multi-Path Inference with Preference Equilibrium

Ziqi Zhang[1*], **Cunxiang Wang**[1*], **Xiong Xiao**[2], **Yue Zhang**[1†], **Donglin Wang**[1†]
[1] School of Engineering, Westlake University
[2] University of Cambridge, UK
{zhangziqi, wangcunxiang, zhangyue, wangdonglin}@westlake.edu.cn

Abstract

Chain of thought (CoT) is a reasoning framework that can enhance the performance of Large Language Models (LLMs) on complex inference tasks. In particular, among various studies related to CoT, multi-path inference stands out as a simple yet effective improvement. However, there is no optimal setting for the number of inference paths. Therefore, we have to increase the number of inference paths to obtain better results, which in turn increases the inference cost. To address this limitation, we can utilize question-related role templates to guide LLMs into relevant roles, thereby increasing the possibility of correct inferences for each path and further reducing dependence on the number of inference paths while improving reasoning accuracy. However, placing LLMs into specific roles may reduce their reasoning diversity and performance on a few tasks where role dependence is low. To alleviate the excessive immersion of the LLM into a specific role, we propose Nash CoT by constructing a competitive system on each path that balances the generation from role-specific LLMs' and the general LLMs' generation, thereby ensuring both effective role adoption and diversity in LLM generation further maintaining the performance of multi-path inference while reducing the requirement of the number of inference paths. We evaluate Nash CoT across various inference tasks, including Arabic Reasoning, Commonsense Question Answering, and Symbolic Inference, achieving results that are comparable to or better than those of multi-path CoT with the equal number of inference paths.

1 Introduction

Large Language Models (LLMs) have profoundly revolutionized the field of Natural Language Processing (NLP) (Ouyang et al., 2022; etc., 2023; Jiang et al., 2023; Brown et al., 2020b; OpenAI, 2024). Specifically, leveraging human-designed instructions as input, LLMs demonstrate superior inference performance across various types of simple reasoning tasks (Radford et al., 2019; Brown et al., 2020a).

However, in complex tasks, direct reasoning LLMs do not yield good results (Rae et al., 2022). To improve LLMs' inference performance on complex inference tasks, it is popular to adopt a step-by-step reasoning framework known as Chain-of-Thought (CoT) prompting (Wei et al., 2023). For example, when appending templates that can guide LLMs to perform step-by-step reasoning, such as "`Let's think step by step`", to the end of given question. We can lead the LLMs to output question-related rationale and further obtain the answer, enabling LLMs to achieve better performance than zero-shot reasoning. Subsequently, among various CoT-related improvements, modified by utilizing multi-path inference which is represented by Wang et al. (2023) is one of the most effective reasoning approaches. However, there is no theoretical proof identifying an optimal number of paths for multi-path reasoning. To ensure better inference performance, it is imperative to augment the inference paths. However, this might burden the inference budgets.

In order to reduce the dependency of multi-path inference on the number of inference paths while further enhancing its effectiveness, we can cast LLMs into a relevant role for a specific question. For instance, we can utilize a template such as "`You're a mathematician, and ···`" to cast the LLM in the role of a mathematician. This approach can improve the accuracy of each path's answer generated by the LLM, thereby reducing the multi-path CoT's reliance on the number of paths, potentially improving the LLM's reasoning.

However, excessively casting the LLM into a specific role using templates may compromise its robustness (diversity), thereby adversely affecting

[*] Equal contribution.
[†] Corresponding authors.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 14572–14587
November 12-16, 2024

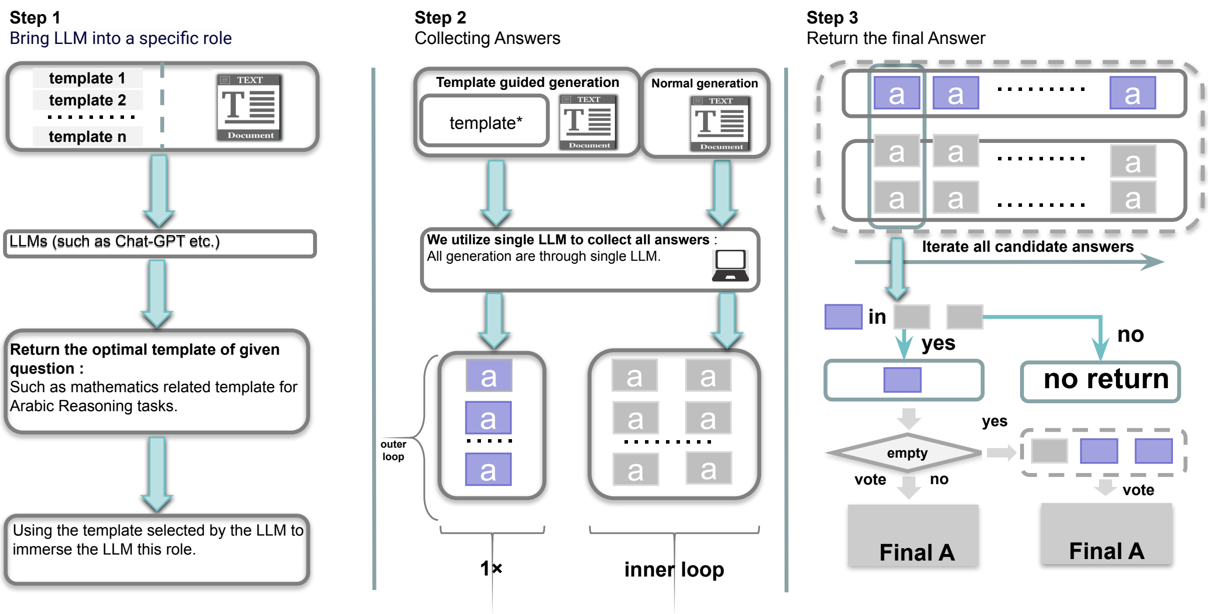

Figure 1: Demonstrations of Nash Chain-of-Thought (Nash CoT). As shown in this figure, Nash CoT can be divided into three main steps. **Step 1** involves bringing the LLM into a template-related role. **Step 2** utilizes the role-immersed LLM and LLM under normal conditions to collect model predictions separately. **Step 3** filters the responses based on Nash Equilibrium.

its performance in solving problems with minimal role dependency. To address this problem, we can introduce a certain degree of randomness into the reasoning process of the LLM. This can mitigate the constraints imposed by excessive role casting on the reasoning performance, ultimately maintaining the role-immersed multi-path CoT's reasoning performance in these questions. Driven by this motivation, we construct a gaming system on each path, where the LLM in the role-playing context and the LLM in its normal state act as two distinct players. When this gaming system formed by these two players reaches a Nash Equilibrium, the reasoning under that path can balance the preferences of both role-playing with that of the LLM's normal state. Furthermore, we can choose the response with the most Nash Equilibrium hits through voting to ensure the performance of LLM reasoning.

We conduct experiments with two local deployed LLMs that include Mistral-Instruct (Jiang et al., 2023) and GLM-4 (Zeng et al., 2022; Du et al., 2022) on various inference tasks, including Arabic Reasoning (Koncel-Kedziorski et al., 2015; Hosseini et al., 2014), Symbolic Reasoning (Wei et al., 2023) and Commonsense Reasoning (Talmor et al., 2019; Geva et al., 2021). As shown in Figure 2, Nash CoT can achieve similar or even better per-

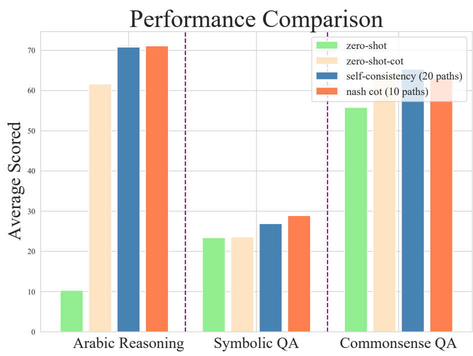

Figure 2: General Performance Comparison. We compare the average performance of, zero-shot, and zero-shot CoT self-consistency (20 Paths) with our Nash CoT (10 Paths) on Mistral-Instruct and GLM4.

formance against self-consistency with fewer inference paths. Meanwhile, as shown in Figure 3, Nash CoT can significantly reduce the inference cost by up to 50% on locally deployed LLMs. To summarize, our contribution can be summarized as follows:

- To our knowledge, we are the first to introduce the concept of Nash Equilibrium into multi-path CoT, with the objective of balancing text generation guided by the role-immersed LLM with that produced by the LLM's default state.

This innovation enables us to minimize the number of paths required to achieve good results in multi-path inference, while simultaneously preserving the multi-path CoT's inherent performance advantages.

- We evaluated the performance of Nash CoT on a wide range of reasoning tasks. Nash CoT can achieve similar results to self-consistency in Arabic Reasoning, Symbolic Reasoning, and Commonsense Question Answering tasks, given the same number of reasoning paths.[1]

2 Related Work

Chain-of-Thought (CoT). There are three majority types of CoT approaches. The first is zero-shot CoT, which involves prompting the LLM with simple instructions to guide its generation step by step (Kojima et al., 2023). The second type is Manual CoT, which begins with sampling several cases from a dataset or manually designing templates. These cases are then used as demonstrations to guide the LLM in generating responses (Wei et al., 2023). However, such methods can be biased if the demonstrations aren't sufficient to cover the optimal distribution. Automatic CoT (Zhang et al., 2022) first clusters the dataset, then samples the most representative sentence from each cluster as the demonstrations and finally uses these demonstrations to guide LLM inference. On the other hand, self-consistency (Wang et al., 2023) showcases strong performance in vast benchmarks. Apart from its impact on inference performance, self-consistency also boasts scalability as a key advantage. It can seamlessly integrate with different approaches, such as tabular chain-of-thought (Tab-CoT) (Jin and Lu, 2023), making it adaptable and versatile for various applications. Furthermore, despite self-consistency can improve LLM's performance on Arabic benchmarks, self-consistency must be inferred multiple times, burdening deployment budgets.

One way to overcome this limitation is to infer multiple paths and fine-tune them based on the most frequent path. Specifically, Huang et al. (2022) proposes that by gathering inferences from multiple paths and sampling the most frequent generation, we can enhance the inference performance of a smaller LLM. However, this approach still requires updating the LLM's parameters, which is inefficient. Therefore, it is necessary to further develop inference methods to maintain the performance of self-consistency while reducing the number of multi-path CoT inference paths.

Preference Optimization. Training policy with Reinforcement Learning (RL) to reflect preference, termed Reinforcement Learning with Human Feedback (RLFH), is initially introduced by Akrour et al. (2011) and has since undergone consistent improvement and refinement by Cheng et al. (2011); Busa-Fekete et al. (2013); Wilson et al. (2012). This approach has been widely applied to adjust the parameters of LLMs in order to align them with human preferences (Ouyang et al., 2022; Jiang et al., 2023; etc., 2023). Recently, a new approach called Direct Optimizing from Preference (DPO) has been proposed by Rafailov et al. (2023), aiming to directly adjust LLMs to reflect human preferences without requiring a reward model. Additionally, Munos et al. (2023) proposes combining DPO with Nash Equilibrium to ensure the convergence of the last iterated policy. Despite our study also utilizing the concept of Nash Equilibrium in the Preference model, the main difference is that we utilize Nash Equilibrium as a standard to pick up the answer instead of optimizing the LLM's parameters.

3 Preference Equilibrium in mini-batch inference

The experimental results of self-consistency indicate that multi-path reasoning can unleash the reasoning capability of LLMs. However, multi-path inference lacks a theoretical foundation for determining the optimal number of inference paths, requiring us to increase the number of paths to ensure performance, and further increase the computational resource consumption. To reduce the number of paths required for multi-path inference, we utilize the concept of Nash Equilibrium to locally construct a binary gaming system in multi-path inference. Specifically, the preference of each valid inference path of the LLM needs to achieve Nash equilibrium with the preferences of the generation guided by the role-immersed LLM. This approach increases the possibility of each path correctly answering the question while maintaining a certain level of robustness (diversity), thereby reducing the number of inference paths required by self-consistency.

[1] We have released our code at <https://github.com/stevezhangzA/nash-chain-of-thought>.

Preference Model. Given text input x, and the sampled predictions (answers) y_1, y_2, and Bradley-Terry reward model r_θ that has been deliberated by Chen et al. We first define y_1 is preferred over y_2 as Equation 1:

$$\mathcal{P}(y_2 \prec y_1|x) := \sigma(r_\theta(y_1|x) - r_\theta(y_2|x)), \quad (1)$$

where $\mathcal{P}$ is preference model reflects the preference when inputting the pairs (y_1, y_2).

Player Templates : Role templates for our LLM that are shown the structure: `{id(player): description.}`

Mathematician: You are a mathematician, you excel at analyzing problems from a mathematical logical perspective and arrive at conclusions that align with your values.
Literary scholar: You are a literary scholar who has read a vast array of literary works. Please consider the problem from the perspective of a literary scholar.
Philosophical: You are a philosopher, your knowledge base includes a wealth of philosophical knowledge. You enjoy approaching problems from a philosophical perspective and arriving at conclusions that align with your values.
Geographer: You are a geographer with a deep understanding of geographical knowledge. Please approach the given problem from the perspective of a geographer.
$\cdots$ `(other cases have been appended to the Appendix.)`

Furthermore, drawing from the definition in (Munos et al., 2023), we define one policy (LLM for decision making) π_1 as more preferred over another policy π_2 as:

$$\mathcal{P}(\pi_2 \prec \pi_1) := \mathbb{E}_{\substack{y_1 \sim \pi_1(\cdot|x) \\ y_2 \sim \pi_2(\cdot|x)}} \left[\mathcal{P}(y_2 \prec y_1|x) \right], \quad (2)$$

subsequently, we imply the necessity and existence of Nash Equilibrium in the Preference model (shown in Equation 1).

Preference Equilibrium. We first discuss why we should construct a gaming system: To increase the accuracy of each inference path, we can select the most suitable template (example 3) for the given question, immersing the LLM into a template-related role. This approach helps LLMs to solve problems more effectively, thereby reducing the number of inference paths needed.

However, this may lead to some issues: If the LLM is excessively immersed in a specific role, it may reduce the diversity of LLM's generation, and further reduce LLM's accuracy in solving tasks that have low role dependence. To address these problems, we locally build a **bi-player gaming system** that the preference of role-immersed LLM (`player 1`) over the normal status of LLM (`player 2`) is the pay-off of role-immersed LLM, vice visa. If `player 1` and `player 2` reach Nash Equilibrium [2], then the generation can balance the preference of both `player 1` and `player 2`.

Definition 3.1 (Preference Equilibrium)**.** *Given any two policys (players) π_1 and π_2 along with the pay-off of π_1 and π_2 as $\mathcal{P}(\pi_1 \prec \pi_2)$ and $\mathcal{P}(\pi_2 \prec \pi_1)$, we define the Nash Equilibrium of π_1 and π_2 as Preference Equilibrium.*

Subsequently, we define the status that `player 1` and `player 2` reach Nash Equilibrium as **Preference Equilibrium** (**Definition 3.1**). Meanwhile, in Theorem 3.1, we prove the existence of Nash Equilibrium in this system. In particular, the strategy of `player 1` equal to `player 2` is one solution that this system has reached **Preference Equilibrium**.

Theorem 3.1 (Existence of Preference Equilibrium)**.** *Given any two policy (player) π_1 and π_2 within the gaming system defined in Definition 1, where $\pi \in \Pi$. $\pi_1 \equiv \pi_2$ is a solution where the gaming system reaches Nash Equilibrium.*

Proof of Theorem 3.1 see Appendix F.

Meanings of the existence of Preference Equilibrium. Theorem 3.1 proves the existence of a Nash Equilibrium between the role-immersed LLM and the normal status of LLM generation. Therefore, when reaching Preference Equilibrium, the preference of generations made by the role-immersed LLM is aligned with those made by the LLM without templates. Meanwhile, the preference for role-immersed LLM generation is much closer to the selected role-immersed context, while those of the normal status LLM are predominantly based on its parameters which are much more diverse than role-immersed LLM generation. Therefore, Preference Equilibrium can balance the re-

[2]Preference Equilibrium leverages the concepts of Nash Equilibrium. Nash equilibrium is proposed by John Nash. It is a concept solution where, assuming each participant knows the equilibrium strategies of the other participants, no participant can benefit by changing their own strategy.

quirement of role-immersed LLM generation and the diversity of the LLM generation in each path. In particular, $\pi_1 \equiv \pi_2$ means their outputs are also likely to be equal. This insight forms a fundamental basis for practically implementing Nash CoT.

3.1 Mini-batch inference with Preference Equilibrium

Subsequently, based on Preference Equilibrium, we conceptualize a mini-batch inference (shown in **Step 2** of Figure 1) as a bi-player gaming system. This approach aims to maintain the performance benefits of multi-path inference while retaining some of the inherent randomness (diversity) of standard inference methods. Before introducing mini-batch inference, we first define x^t as the template of zero-shot CoT. Meanwhile, we leverage LLMs to select a player template from the candidate set $\{x_0^c, x_1^c, \cdots, x_n^c\}$ for role-immersed generation, guided by Equation 1. In terms of the process

Algorithm 1 Nash CoT (Answer Gathering)

Require: Candidate question q sampled from $\mathcal{Q} = \{q_0, q_2, \cdots, q_n\}$; Outer iterations $\mathrm{n_{outer}}$; Num of mini-batch inference $\mathrm{n_{mini}}$; Large language model π; CoT prompt x^t, candidiate player template $\{x_0^c, x_1^c, \cdots, x_n^c\}$, $\mathrm{Prompt}(\{x^c\})$ is used to point out the most preferred x^c.

Generation:

Initialize answer list ans $= [\,]$.
$x_c \leftarrow \pi(\cdot|\mathrm{Prompt}(\{x^c\}))$
for t in range($\mathrm{n_{outer}}$) **do**
  Initialize prefer pairs pref $= [\,]$.
  for t in range($\mathrm{n_{mini}}$) **do**
    $y \leftarrow \pi(\cdot|x^t, q)$;
    ans.append(y);
  end for
  $y^* \leftarrow \pi(\cdot|x^c, x^t, x)$
  τ.append($[y^*, \mathrm{ans}]$)
end for
Return τ

of mini-batch inference, we firstly infer LLM twice times *i.e.* $[y_0, y_1] \leftarrow [\pi(\cdot|x^t, x), \pi(\cdot|x^t, x)]$, in particular, we have conducted ablations about the setting of 'twice' in section ablation. Meanwhile, due to the inherent uncertainty (diversity) of LLM, the generation of $[y_0, y_1]$ can be considered a potential set of distinct predictions. Subsequently, the role-immersed generation can be sampled by querying LLM with x^c and x^t *i.e.* $y^* \leftarrow \pi(\cdot|x^c, x^t, x)$. Furthermore, we can select an answer from y_1 and y_2 that is the same as y^*, thereby satisfying Theorem 3.1.

Based on the mini-batch inference, we further introduce Nash CoT in the next chapter. (In particular, in the current framework, successful inference cannot always be guaranteed. For instance, y^* may not always in $[y_1, y_2]$. We will address this problem in the following chapters.)

4 Nash Chain of Thought (Nash CoT)

Nash CoT can be regarded as an extension of mini-batch inference with Preference Equilibrium, implementing multiple mini-batch inferences to further enhance performance. This approach is inspired by self-consistency experiments, which suggest that increasing the number of paths enhances LLM's inference accuracy. Specifically, Nash CoT's reasoning process for each question can be divided into two stages: Answer Gathering and Answer Filtering.

Answer Gathering. When generating candidate answers, the process predominantly involves two types of loops: **mini-batch Loops** ($\mathrm{n_{mini}}$): As shown in Algorithm 1, this process involves searching for role-immersed generations within two rounds of generation $[y_1, y_2]$. We denote the times of these two predictions as the $\mathrm{n_{mini}}$. Moreover, to mitigate the impact brought by low-frequency predictions, we introduce iterating $\mathrm{n_{mini}}$ multiple times. This leads us to another type of loop: **Outer Loops** ($\mathrm{n_{outer}}$): This loop resembles the concept of multi-path inference that iterates $\mathrm{n_{mini}}$ multiple times. After completing loop $\mathrm{n_{outer}}$, we filter the generated answers and retain the answer that reaches Preference Equilibrium most frequently (shown in Algorithm 2), as the predicted answer.

Algorithm 2 Nash CoT (Answer Filtering)

Require: Preference pair list τ

Filtering:

Initialize hash table: hash $= \{\} : k \rightarrow v$.
Initialize new answer list nans $= [\,]$.
for $[y^*, \mathrm{ans}]_i$ in τ **do**
  if $y^* \in$ ans **then**
    hash[y^*]+=1
  end if
  nans.extend($[y^*, \mathrm{ans}[0], \mathrm{ans}[1]]$)
end for
if hash $\equiv \{\}$ **then**
  return the most frequent y in nans
else
  return $y \leftarrow k = \arg\max_v \mathrm{hash}$
end if

Answer Filtering. In terms of answer filtering, as shown in Algorithm 2 we first count the most frequent prediction to satisfy Preference equilibrium. Specifically, we count all y^* satisfy $y^* \in [y_1, y_2]$ and compute their frequency. Subsequently, we

Core LLM	Methods	SingleEQ	AddSub	MultiArith	GSM8K	AQuA	SVAMP	Avg.
Mistral-Instruct (7B)	zero-shot	15.3±0.8	12.0±2.8	3.3±1.3	2.7±2.0	20.8±1.5	7.7±2.0	10.3±7
	zero-shot CoT	76.0±0.8	82.5±2.0	75.4±6.1	44.3± 4.0	27.9±2.3	63.4±6.9	61.6±19
	self-consistency (20 Paths)	82.5±0.8	86.3±5.1	86.3±2.8	58.5±2.8	34.4±6.1	76.5±2.8	70.8±19
	Nash CoT (10 Paths)	81.4±0.8	86.3±6.0	86.3±4.7	55.7±5.8	39.9±5.4	77.0±3.5	71.1±17
GLM4-chat (9B)	zero-shot	1.1±1.5	1.1±1.5	12.6±3.9	12.0±2.0	22.4±4.1	4.4±2.8	8.9±8
	zero-shot CoT	90.7±1.5	90.7±1.5	98.4±1.3	80.9±2.8	20.8± 3.1	86.9±3.5	78.1±26
	self-consistency (20 Paths)	92.3±2.0	90.2±2.3	98.4±2.3	89.3±0.2	20.8±3.1	91.5±1.1	80.4±27
	Nash CoT (10 Paths)	91.3±0.8	90.2±2.7	96.7±3.3	80.3±1.3	20.8±3.1	88.0±2.0	77.9±26

Table 1: Experimental results on Arabic Reasoning benchmarks. We test Zero-Shot CoT and Nash CoT with the core LLM includes Mistral-Instruct (7B) and GLM4-chat (9B) on mathematical benchmarks including AddSub, MultiArith, SingleEQ, SVAMP, GSM8K, and AQuA. Nash CoT performs the best.

return the most frequent answer. Otherwise, if no cases satisfy $y^* \in [y_1, y_2]$, we adapt answer filtering by selecting the most frequent prediction among all generated answers.

Preference Templates : utilized to confine the prompt for preference model.

Q: Current issue is **{query}**, and the best player is who? Please give us the number of that player from the options below: **{description}**. There are total **N({key(player)})** players including **{key(player)}**. Please point out the most appropriate player for the following task: **candidate questions**

A: Let us think step by step. → z (obtain the rational z)

A: Let us think step by step. + z+ Therefore, the most appropriate player in this game is who? (please direct give us the number)

Practical Implementation of Preference model $\mathcal{P}$. In the process of practical implementation, we do not explicitly train a reward model r_θ to confine the player template x^c utilizing Equation 1. Instead, we directly utilize the template (shown in **Preference Template**) to guide the LLM in determining the most suitable player template for a given question. For example, when presented with a coin flip question as shown in Figure 4, we fill the **Preference Template** with the given question and **Player Templates**. This filled template is then input into the LLM to provide the id of the most suitable player template from the available options. In particular, we consider utilizing LLM to search templates to be effective, this is because most of the baselines we selected have been turned to reflect human preference, thus LLM can be directly utilized as the preference model to point out the most preferred option among candidate options (we provide cases in section A). Subsequently, we propose Nash CoT, which iterates through Algorithm 1 and Algorithm 2 to perform inference on all sampled questions, where τ represents the candidate answers from the Answer Gathering stage.

5 Experiments

The goal of our experiment is to 1) showcase the performance advantage and effectiveness of Nash CoT. 2) showcase whether Nash CoT helps reduce inference paths and inference time.

Datasets. Our majority benchmarks are composed of three different kinds of inference tasks. 1) *Arabic Reasoning:* SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), and SVAMP (Patel et al., 2021). 2) *Symbolic Reasoning:* Last Letters, Coin Flip (Wei et al., 2023), and Object Tracking, Bigbench Date. 3) *Commonsense Question Answering:* CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). For more details about the dataset please refer to Appendix A.

LLMs. To validate that Nash CoT is a general CoT method, we selected different large models as test models, including Mistral-Instruct (7B) (Jiang et al., 2023), Chat-GLM4 (9B) (Zeng et al., 2022; Du et al., 2022). In particular, all of these selected LLMs are turned via RLHF, and the difference between LLM turned with RLHF and the original foundation models have been detailed by Ouyang et al..

Baselines. The preliminary baselines we utilized include zero-shot, zero-shot CoT (Wei et al., 2023), and self-consistency (Wang et al., 2023). We test these approaches with frozen LLMs.

Core LLM	Methods	Coin-Flipping	Last Letters	Object Tracking	Bigbench Date	Avg.
Mistral-Instruct (7B)	zero-shot	26.8±5.1	0.0±0.0	35.5±4.1	31.1±7.6	23.4±14
	zero-shot CoT	27.9±4.0	0.0±0.0	30.1±2.8	36.6±5.4	23.6±14
	self-consistency (20 Paths)	21.9±4.7	0.0±0.0	38.8±0.8	47.0±1.5	26.9±18
	Nash CoT (10 Paths)	29.0±5.4	0.5±0.08	44.8±2.0	41.1±1.2	28.9±17
GLM4-chat (9B)	zero-shot	27.3±6.3	0.0±0.0	38.8±0.8	16.4±2.3	20.7±14
	zero-shot CoT	87.4±0.8	0.0±0.0	37.7±2.3	16.4±4.8	35.4±33
	self-consistency (20 Paths)	98.9±1.5	0.0±0.0	37.7±2.3	16.4±4.8	38.3±38
	Nash CoT (10 Paths)	93.4±2.7	0.0±0.0	37.7±2.3	16.4±4.8	36.9±35

Table 2: Experimental results on symbolic inference benchmarks. We test Zero-Shot CoT and Nash CoT with Mistral-Instruct (7B) and GLM4-chat (9B) on Symbolic QA benchmarks including Coin-Flipping, Last Letters, and Object Tracking. Among these baselines, Nash CoT performs the best.

Core LLM	Methods	StrategyQA	CommonsensQA	Avg.
Mistral-Instruct (7B)	zero-shot	49.2±8.8	62.3±4.8	55.8±7
	zero-shot CoT	57.4±2.3	70.5±2.7	64.0±7
	self-consistency (20 Paths)	59.6±2.0	71.0±3.4	65.3±6
	Nash CoT (10 Paths)	56.8±2.0	69.4±4.7	63.1±6
GLM4-chat (9B)	zero-shot	56.8±4.7	17.5±2.0	37.2±20
	zero-shot CoT	63.9±2.3	18.0±2.3	41.0±23
	self-consistency (20 Paths)	69.9±3.3	18.0±2.3	44.0±26
	Nash CoT (10 Paths)	66.7±0.8	18.0±2.3	42.4±24

Table 3: Experimental results on Commonsense Reasoning. We test Zero-Shot CoT and Nash CoT with Mistral-Instruct (7B) and GLM4-chat (9B) on Commonsense Reasoning datasets includes StrategyQA and CommonsenseQA .

Settings. Our evaluation of all selected tasks utilizes the same experimental settings below:

- **zero-shot and zero-shot CoT.** We follow the method proposed by Wei et al. (2023) and use the original template (e.g., "`Let's think step by step`") for evaluation.
- **self-consistency.** We follow Wang et al. to evaluate the performance of self-consistency with selected LLMs, utilizing the zero-shot CoT template. Additionally, we set the number of inference paths to 20.
- **Nash CoT.** We set up n_{outer} as 3 and n_{mini} as 2, resulting in a total of $n_{\text{outer}} \times (n_{\text{mini}} + 1) + 1 = 10$ paths. Additionally, we have provided the **Player Templates** x^t in Table 1 and the Appendix, meanwhile, we utilizing the same CoT template x^c as in zero-shot CoT.

Specifically, in Table 1, 2 and 3, we set the total number of reasoning paths for Nash CoT to 10 and for self-consistency to 20. The primary goal is to demonstrate that Nash CoT can achieve the same or even better results with only half the number of reasoning paths compared to self-consistency.

Additionally, all evaluations are conducted on the inference of 60 random sampled questions per seed. We have provided the mean and standard error in the majority tables.

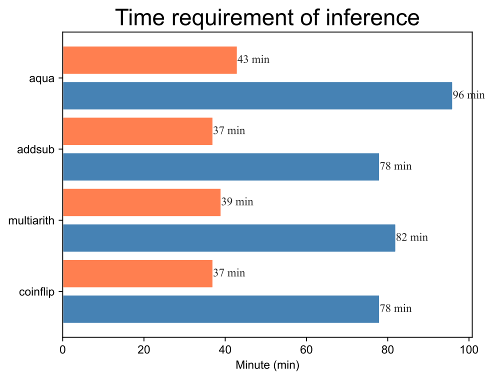

Figure 3: We used GLM4-chat (9B) on the same type of GPU (A100) to evaluate Nash CoT and self-consistency across selected tasks (60 questions per task). Nash CoT, employing a total of 10 paths, requires nearly half the time of self-consistency, which has 20 paths in total.

5.1 Experimental Results

Evaluated Scores. The majority experimental results are demonstrated in Table 1, 2 and 3. Nash CoT can improve Mistral-Instruct (7B) on almost all selected inference tasks while showcasing similar performance to self-consistency with twice inference paths on GLM4-chat (9B). In particular, we have provided the total paths of Nash CoT that it only require half of self-consistency, thus our claim in section 3 can be validated. When focusing on Mistral-Instruct (7B), Nash CoT has better per-

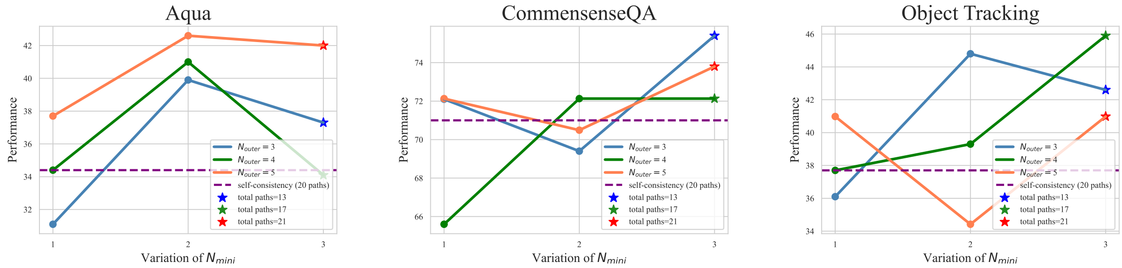

Figure 4: We use Mistral-Instruct (7B) to examine the impact of loop numbers on the inference performance of the large language model. Specifically, we used solid lines of specific colors to represent the experimental performance under certain N_{outer} as the N_{mini} changed. We marked self-consistency with 20 paths using dashed lines, and some results of Nash CoT, with total paths close to 20, were marked with stars.

Models	**Settings**	**Cnt and Prob**	**Precalculus**	**Inter-Alg**	**Number Theory**	**Avg.**
Mistral-Instruct (7B)	zero-shot-cot	40.9	16.4	16.4	24.6	24.6
	Nash CoT	64.8	**38.5**	36.9	**27.9**	**42.0**
	self-consistency	**72.1**	34.4	34.4	21.3	40.6
GLM4 (large)	zero-shot-cot	83.6	90.2	85.2	91.9	87.7
	Nash CoT	**95.1**	90.2	**98.3**	98.3	**95.5**
	self-consistency	93.3	90.2	95.1	100.0	94.7

Table 4: Equal Path Evaluation. This experiment is conducted to verify the performance of Nash CoT when the number of inference paths is set equal to self-consistency. In particular, Cnt and Prob denote Counting and Probability, while Inter-Alg denotes Intermediate Algebra.

formance on Arabic and Symbolic inference tasks, showcasing its superior performance on logic/math inference tasks. However, Nash CoT does not showcase improved performance in Commonsense Question Answering tasks. We argue that this is because Commonsense Question Answering tasks are more diverse, and the player template can't cover all topics. Therefore, the predefined player template limits Nash CoT on Commonsense Question Answering tasks. Importantly, we evaluate Nash CoT by utilizing only a total of 10 paths for inference in this section. However, additional experimental results in the ablation section show that Nash CoT outperforms self-consistency by increasing the inference loops. Meanwhile, we also evaluate by setting equal paths on much more challenging MATH (Hendrycks et al., 2021), showcasing that Nash CoT can outperform self-consistency.

Comparing Nash CoT with self-consistency with equal paths. To further evaluate the performance of Nash CoT, we utilize Mistral-Instruct (7B) while selecting one of state-of-the-art (SOTA) LLMs named GLM4 (large) (GLM et al., 2024), for comparison using an equal number of reasoning paths (10 paths). As shown in Table 4, Nash CoT outperforms self-consistency, which demonstrates the superiority of Nash CoT compared to self-consistency when the same number of path settings are set.

Inference Time. The path of Nash CoT is composed of three different kinds of types *i.e.* zero-shot CoT for problem inference (in loop N_{mini}), zero-shot CoT for player confining (in the outside loop of N_{outer}), and player role-immersed CoT inference. Accordingly, the different path requires different inference time. Therefore, we further count the total time requirement of self-consistency and Nash CoT in Figure 3, Nash CoT requires fewer inference time.

6 Ablation Study

We further conduct extensive ablations to answer the following questions: 1) Is there a correlation between the performance of Nash CoT and the setting of the number of inference paths and can the performance of Nash CoT be improved to surpass self-consistency by adjusting the number of inference paths? 2) Does the template improve the accuracy of path predictions, and what impact does it have on experimental performance? 3) Is the Nash Equilibrium the reason for the performance improvement observed in Nash CoT?

As the number of inference paths increases, Nash CoT can obviously surpass self-consistency with fewer inference paths. To address question 1), we selected Mistral-Instruct (7B) and evaluated three different reasoning tasks, adjusting the N_{mini} and N_{outer}. As shown in Figure 4, as the number of loops increases, Nash CoT has a high possibility of outperforming self-consistency with fewer paths. However, different from self-consistency, the experimental results of Nash CoT do not show a monotonic (linear) relationship with the total number of paths. This indicates that there is a significant difference between Nash CoT and self-consistency. Unlike Nash CoT, the experimental results of self-consistency show a clear improvement in performance as the number of paths increases.

The performance is impacted by the player template. To illustrate the impact of the template, we removed the mathematical templates from the **Player Templates** and then evaluated Nash CoT on selected Arabic reasoning. Results are shown in Table 5, showing an approximately 9.2% decrement in **GSM8K** and 6.2% decrement in **SVAMP**. Therefore, the performance of Nash CoT is impacted by the **Player Templates**. As shown in Table 6, we used three role templates with varying degrees of relevance to mathematical problems: mathematician, student, and poker player, to test the performance of Mistral-Instruct (7B) on GSM8K. Role templates with higher relevance to mathematical problems can bring more improvements to Mistral, further confirming the rationality of enhancing reasoning performance by substituting LLM into roles using templates.

GSM8K	AQuA	SVAMP
55.7→ 50.6	39.9→ 39.8	77.0 →72.2

Table 5: Performance decreasing. We remove the mathematics from **Player Templates** and test Nash CoT on selected Arabic Reasoning tasks.

mathematician	student	poker player
57.37	52.45	47.54

Table 6: Results on GSM8K. The templates for these roles are shown in Table 9 on the Appendix.

Effectiveness of Nash Equilibrium. While Nash CoT has demonstrated effectiveness across various LLMs and benchmarks, however, the extent to which Nash CoT's performance is driven by Nash Equilibrium remains to be determined. To further investigate, we compare the self-consistency integrated template (denoted as `self-con.`+*temp* (10 paths)) with Nash CoT (10 paths) using Mistral-Instruct (7B). As shown in Table 7, Nash CoT consistently outperforms `self-con.`+*temp* when applying the same mathematical templates across selected tasks.

Settings	AQuA	SVAMP	GSM8K	AddSub	Avg.
`self-con.`+*temp*	31.7	71.6	53.6	85.8	60.6
Nash CoT	**39.9**	**77.0**	**55.7**	**86.3**	**64.7**

Table 7: Ablation of Nash Equilibrium. This experiment is conducted to verify the necessity of Nash Equilibrium in the performance of Nash CoT.

7 Conclusion

We prove the existence of a Nash equilibrium in preferences between role-immersed and normal-status LLM generations, and develop a gaming system within each path of the multi-path CoT to introduce Nash CoT. Experimental results show that Nash CoT can perform equally or even better than self-consistency while requiring fewer paths.

Limitations and Future Work

Since Nash CoT relies on predefined templates, its performance will decline in new scenarios due to limited adaptability in template selection. Future efforts will concentrate on automating the balancing of task feedback and template selection.

Acknowledgement

This work was supported by the National Science and Technology Innovation 2030 - Major Project (Grant No.2022ZD0208800), and NSFC General Program (Grant No.62176215). Meanwhile, we thank the reviewers for their valuable suggestions.

Ethics Claims

Pre-training or fine-tuning an LLM requires much more computing resources. Multi-path CoT is an ideal approach that has been proven to enhance the performance of LLMs' inference. Meanwhile, Nash CoT effectively reduces the inference paths needed and times of multi-path inference. We believe our approach can further elevate the effectiveness of multi-path inference, thereby further improving the effectiveness of LLM.

References

Riad Akrour, Marc Schoenauer, and Michèle Sebag. 2011. Preference-based policy learning. In *ECML/PKDD*.

Tom Brown, Benjamin Mann, and etc. Ryder. 2020a. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020b. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Róbert Istvan Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. 2013. Preference-based evolutionary direct policy search.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. 2011. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *ECML/PKDD*.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Hugo Touvron etc. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Preprint*, arXiv:2101.02235.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *Preprint*, arXiv:2210.11610.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Ziqi Jin and Wei Lu. 2023. Tab-cot: Zero-shot tabular chain of thought. *Preprint*, arXiv:2305.17812.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation : Learning to solve and explain algebraic word problems. *Preprint*, arXiv:1705.04146.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, Marco Selvi, Sertan Girgin, Nikola Momchev, Olivier Bachem, Daniel J. Mankowitz, Doina Precup, and Bilal Piot. 2023.

Nash learning from human feedback. *Preprint*, arXiv:2312.00886.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2022. Scaling language models: Methods, analysis & insights from training gopher. *Preprint*, arXiv:2112.11446.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *Preprint*, arXiv:1608.01413.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2012. A bayesian approach for policy learning from trajectory preference queries. In *Neural Information Processing Systems*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

A Dataset

Our majority dataset are composed of three different kinds of inference tasks.

- *Arabic Reasoning:* SingleEq (Koncel-Kedziorski et al., 2015), AddSub (Hosseini et al., 2014), MultiArith (Roy and Roth, 2016), GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), SVAMP (Patel et al., 2021), and MATH dataset (Hendrycks et al., 2021). In particular, MATH dataset are process by our own, : When constructing the dataset, we identify the answers to each question in the MATH dataset through ground truth (GT) reasoning, retaining a certain number of questions for each category. When testing the LLM, we input the questions, collect the responses from the deployed LLM, and determine whether the LLM's reasoning is correct by comparing whether the inferred results match the GT, examples are shown in Table 11. Meanwhile, the dataset or its url will be released at our codebase page.

Dataset	Capacity
Algebra	778
Counting and Probability	291
Geometry	235
Intermediate Algebra	443
Number Theory	474
Prealgebra	618
Precalculus	133

Table 8: Data type and data capacity

- *Symbolic Reasoning:* Last Letters, Coin Flip (Wei et al., 2023), and Object Tracking, Bigbench Date.
- *Commonsense Question Answering:* CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021). For more details about the dataset please refer to (Wang et al., 2023).

B Experimental Settings

In this section, we compare Nash CoT with self-consistency, focusing specifically on self-consistency as a representative method. As illustrated in Figure 1, Nash CoT operates with two distinct loops, where the final answer is derived from voting on the response that achieves Preference Equilibrium most frequently. In contrast, self-consistency determines the most frequent answer across all predictions.

C Usage of LLM.

We utilize LLM to rectify grammar errors.

D Computing Resources

Our experiments were run on a computer cluster with 32GB RAM, 4-Core CPU, and NVIDIA-A100 (80G, 32G)/NVIDIA-V100 (32G) GPU, Linux platform.

E Source Code.

We have provided source code for reference. Additionally, our code are based on `https://github.com/amazon-science/auto-cot` and refer to the coding manner from `https://github.com/eureka-research/Eureka`.

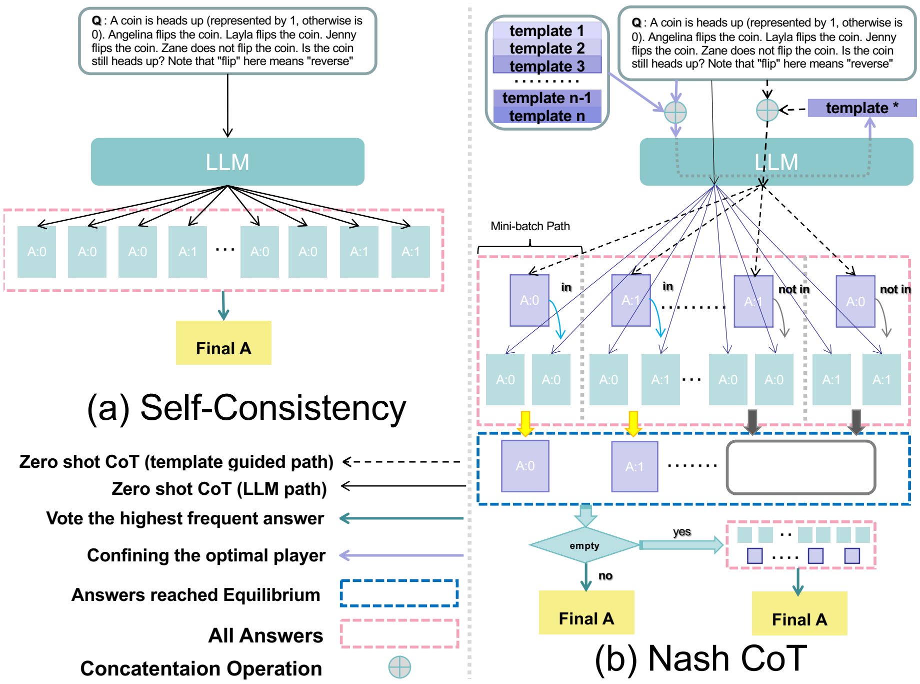

Figure 5: Comparison of Nash CoT and self-consistency. (right) Nash Chain of thought (Nash CoT). (left) self-consistency.

F Proof of theorem 3.1.

Subsequently, we prove the existence of Nash equilibrium in this system. For any two given polices $\pi_1 \in \Pi$ and $\pi_2 \in \Pi$ We first define the pay-off of π_1 and π_2 as $R(\pi_1; \pi_2)$ and $R(\pi_2; \pi_1)$:

$$\begin{aligned} R(\pi_1; \pi_2) &= \mathcal{P}(\pi_1 \prec \pi_2) \\ R(\pi_2; \pi_1) &= \mathcal{P}(\pi_1 \succ \pi_2), \end{aligned} \tag{3}$$

we provide the proof of the existence of Nash equilibrium in this system. We define $\bar{\pi} = [\pi_1, \pi_2]$, $v(\bar{\pi}) = [R(\pi_1; \pi_2), R(\pi_1; \pi_2)]$. According to the Nash equilibrium, it should have to satisfy this relationship:

$$v(\bar{\pi}^*)(\bar{\pi}^* - \bar{\pi}) \leq 0 \tag{4}$$

Subsequently, refer to Munos et al., we can learn that if we want Equation 4 holds true, we just have to guarantee Equation 5 holds true.

$$(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') \leq 0, \tag{5}$$

where $\bar{\pi}$ and $\bar{\pi}'$ are any two given policy set. Subsequently, we can further arrive at the following relationships:

$$
\begin{aligned}
(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') &= \begin{pmatrix} R(\pi_1;\pi_2) - R(\pi_1';\pi_2') \\ R(\pi_2;\pi_1) - R(\pi_2';\pi_1') \end{pmatrix} \cdot \left(\pi_1 - \pi_1', \pi_2 - \pi_2'\right) \\
&= \Big(R(\pi_1;\pi_2) - R(\pi_1';\pi_2')\Big) \cdot (\pi_1 - \pi_1') + \Big(R(\pi_2;\pi_1) - R(\pi_2';\pi_1')\Big) \\
&\cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2')\Big) \cdot (\pi_1 - \pi_1') + \Big\{2 - \Big(\mathcal{P}(\pi_1 \prec \pi_2) + \\
&\mathcal{P}(\pi_1' \prec \pi_2')\Big)\Big\} \cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2')\Big) \cdot (\pi_1 - \pi_1' - \pi_2 + \pi_2') + 2 \cdot (\pi_2 - \pi_2') \\
&= \Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2') - 2\Big) \cdot (\pi_2' - \pi_2) + \\
&\Big(\mathcal{P}(\pi_1 \prec \pi_2) - \mathcal{P}(\pi_1' \prec \pi_2')\Big) \cdot (\pi_1 - \pi_1').
\end{aligned}
\tag{6}
$$

In particular, we can find that if $\bar{\pi} \equiv \bar{\pi}'$ then $(v(\bar{\pi}) - v(\bar{\pi}'))^{\mathrm{T}}(\bar{\pi} - \bar{\pi}') \equiv 0$, thus $\bar{\pi} \equiv \bar{\pi}'$ is one solution that π_1 and π_2 has reached equilibrium.

G Further clarify our settings

Why in Table 1, 2 and 3 do we set the number of pathways for Nash CoT to be half of self-consistency? If Nash CoT can surpass self-consistency with half the number of reasoning pathways, then it is easier to demonstrate the validity and efficiency of Nash CoT. Because that according to the experimental conclusion in Wang et al. that "`The experimental effect of self-consistency increases as the number of pathways increases.`", if Nash-CoT outperforms self-consistency with fewer paths, it indicates that Nash-CoT is better than self-consistency across all settings with an equal or greater number of paths.

Which one performs better between Nash CoT and self-consistency when they have the same number of pathways? In our ablation experiments to test the impact of the number of pathways on Nash CoT, we found that by adjusting the number of pathways, Nash CoT can surpass self-consistency with fewer pathways on Aqua, CommensenseQA, and Object Tracking. Additionally, in Table 4, we used GLM4 (large), Mistral-Instruct (7B) to outperform self-consistency on the MATH dataset with the same number of pathways.

H Case Study

H.1 LLM as preference model

In this experiment, we provided examples of LLM inferences to demonstrate that LLMs can select the most appropriate template based on the question. Specifically, in Table 9, we set up a series of templates for different roles, arranging them in different sequences and labeling them with numbers. These were input into the model, allowing it to rank the templates according to its preference for the question. We observe in Table 10 that the order given by the LLM corresponded to the performance strengths of different templates on the respective tasks, which proves that LLMs can directly serve as preference models to select the optimal template for a specific question. Meanwhile, we utilize ChatGPT to rank these templates based on their relevance to the topic of questions. Our inputs are shown in Table 10. The model's ranking of different roles is consistent with their performance levels on the mathematical dataset. Meanwhile, changing the input order of the templates simultaneously does not affect the model's ranking of the templates.

Player	Description	Correlation
Mathematician	You are a mathematician, You excel at analyzing problems from a Mathematical logical perspective and arrive at conclusions that align with your values.	High
Student	You are a student, please answer the following questions.	Medium
Poker Player	You are playing a poker game, please do your best to play poker games.	Low

Table 9: Player templates we utilized.

Input	Ranking
We have four templates: 1. Mathematician: You are a mathematician, You excel at analyzing problems from a Mathematical logical perspective and arrive at conclusions that align with your values. 2. Student: You are a student, please answer the following questions. 3. Poker Player: You are playing a poker game, please do your best to play poker games. Please rank their correlations to a math problem, and directly output the sequential ranked numerical tag.	1, 2, 3
We have four templates: 1. Student: You are a student, please answer the following questions. 2. Mathematician: You are a mathematician, You excel at analyzing problems from a Mathematical logical perspective and arrive at conclusions that align with your values. 3. Poker Player: You are playing poker game, please do your best to play poker games. Please rank their correlations to math problem, and directly output the sequential ranked numerical tag.	2, 1, 3

Table 10: We provide cases for the ranking provided by LLM. The roles of Mathematician, Student, and Poker Player achieved scores of 57.37, 52.45, and 47.54, respectively, on GSM8K. The performance of these different roles is consistent with the model's ranking.

I Format of our processed MATH dataset

We provide several cases for readers to understand the format of our processed MATH dataset.

Dataset	Question	Answer	Capacity
Algebra	How many vertical asymptotes does the graph of $y = \frac{2}{x^2+x-6}$ have?	2	778
Counting and Probability	n fair 6-sided dice are simultaneously rolled. The probability that exactly two of them show a number other than 1 is $\frac{25}{216}$. Find n.	4	291
Geometry	We have triangle $\triangle ABC$ where $AB = AC$ and AD is an altitude. Meanwhile, E is a point on AC such that $AB \parallel DE$. If $BC = 12$ and the area of $\triangle ABC$ is 180, what is the area of $ABDE$?	135	235
Intermediate Algebra	Find the sum of all complex roots of the equation $$\frac{1}{x-1} + \frac{1}{x-5} + \frac{1}{x-10} + \frac{1}{x-25} = 2,$$ given that there are no repeated roots.	43	443
Number Theory	Kirsty needs to hire a plumber to fix her house. The plumber charges 242_5 dollars for every hour of labor and 367_8 dollars for equipment. If the plumber works for 3.5_{10} hours, how many dollars (in base ten) will Kirsty owe the plumber?	499	474
Prealgebra	John and Gary are playing a game. John spins a spinner numbered with integers from 1 to 20. Gary then writes a list of all of the positive factors of the number spun except for the number itself. Gary then creates a new spinner with all of the numbers on his list. John then spins this spinner, and the process continues. The game is over when the spinner has no numbers on it. If John spins a 20 on his first spin, what is the maximum number of total spins (including the one he already made) that John can make before the game is over?	4	618
Precalculus	Let $\mathbf{a}$ and $\mathbf{b}$ be vectors such that $$\mathbf{v} = \operatorname{proj}_{\mathbf{a}} \mathbf{v} + \operatorname{proj}_{\mathbf{b}} \mathbf{v}$$ for all vectors $\mathbf{v}$. Enter all possible values of $\mathbf{a} \cdot \mathbf{b}$, separated by commas.	0	133

Table 11: Examples of MATH dataset. We give the format of our processed MATH dataset. Meanwhile, the dataset or its URL will be available on our codebase page upon release.