# NeuralFSM: Adaptive Multi-Agent Coordination via Learning Finite-State Execution Policy

**Jiye Wang[1,2], Yu Wang[1], Jianbin Li[1]*, Shiduo Yang[1], Kenan Guo[1], Yuanhe Zhao[1]**

[1]North China Electric Power University, Beijing, China
[2]State Grid Corporation of China, Beijing, China
{lijb87,yuwang}@ncepu.edu.cn

## Abstract

LLM-powered multi-agent systems (MAS) have demonstrated strong performance on complex tasks. However, most existing approaches still rely on hand-crafted communication protocols or automatically designed communication topologies, which generalize poorly across tasks. We introduce NeuralFSM, a state-driven framework that formulates multi-agent problem solving as a finite-state execution process. NeuralFSM learns both the state transition distribution and inter-agent communication weights from interaction traces using a Temporal Coordination Controller. Rather than prioritizing explicit structure generation, the proposed framework uses task context to modulate transition and routing decisions, enabling flexible coordination without manual protocol design. To improve robustness against noisy or adversarial agents, we incorporate graph regularization during training and apply trust-aware message attenuation at runtime. Experiments on diverse benchmarks show that NeuralFSM consistently outperforms previous baselines by an average margin of $6.74\% \sim 19.39\%$, while substantially reducing token consumption. Moreover, NeuralFSM exhibits strong inherent robustness, which is further enhanced by the protection layer, resulting only in a $1.82\%$ performance drop under attack. The code is available at https://github.com/DisseverYOLO/NeuralFSM.

## 1 Introduction

Large language models (LLMs) have enabled strong reasoning and generation capabilities, motivating LLM-based multi-agent systems (MAS) that decompose complex tasks into specialized roles and collaborative interactions (Wu et al., 2024a; Hong et al., 2024; Qian et al., 2024). In many domains, such role specialization and coordination can outperform single-agent prompting, while also improving interpretability through explicit intermediate artifacts (Guo et al., 2024; Li et al., 2024; Tran et al., 2025).

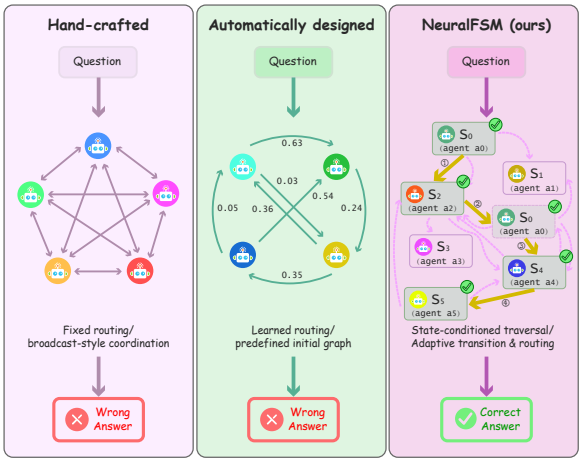


Figure 1: Comparison of multi-agent system design paradigms. Hand-crafted approaches rely on fixed inter-agent communication routing, lacking cross-domain flexibility, while broadcast-style collaboration incurs significant communication overhead. Automatically designed methods can learn problem-specific communication routing but still require partially specifying the initial agent interaction graph. In contrast, our approach dynamically and adaptively performs both state-conditioned transitions and communication routing within an FSM for each problem, guided by a temporal coordination controller.

Despite this progress, existing MAS are often limited by (i) fixed or hand-crafted communication topologies that fail to adapt to task context (Park et al., 2023; Zhang et al., 2025c), (ii) manually specified state-machine rules that require extensive domain engineering (Zhang et al., 2025e; Wu et al., 2024b), (iii) inefficient information sharing via broadcast or ad-hoc routing (Chen et al., 2025; Wu et al., 2024a), and (iv) fragility under noisy or adversarial agents as both coordination scale and domain complexity increase (Wang et al., 2025; Li et al., 2025b).

*Corresponding author.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33414–33436
July 2-7, 2026 

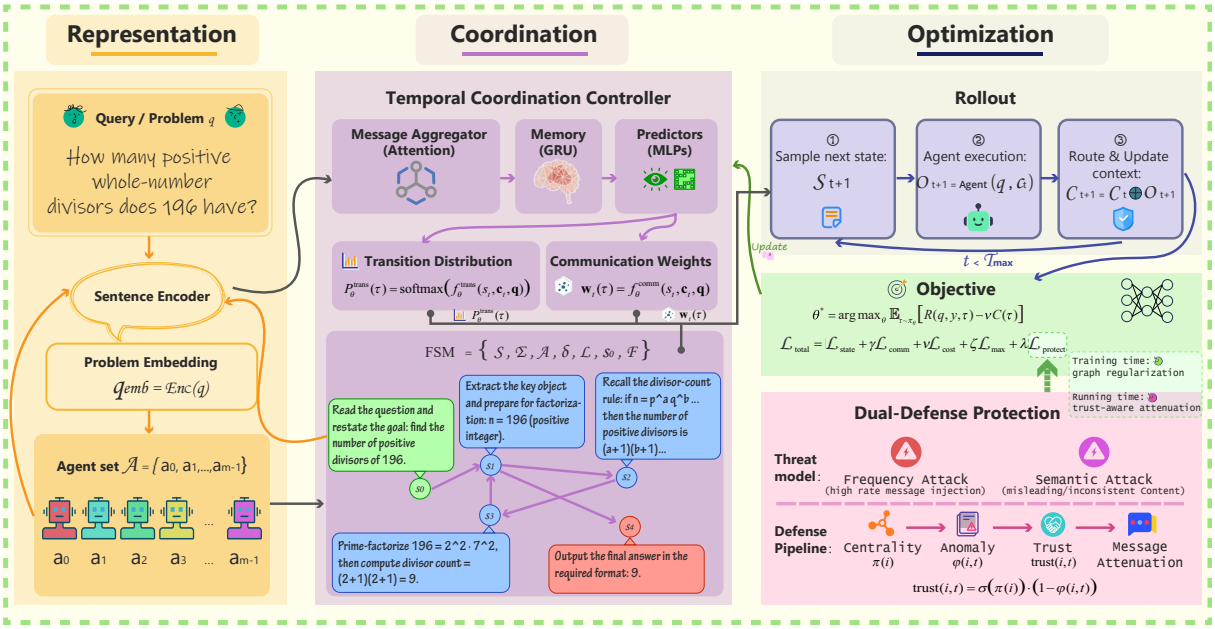


Figure 2: Overview of the NeuralFSM Framework. ***(Left)*** Initialization of the FSM by embedding the problem instance, agents, and states into a shared vector space. ***(Middle)*** A temporal coordination controller processes interaction trajectories to produce state transition distribution and communication routing weights. ***(Right)*** Joint optimization of the controller parameters and FSM execution by maximizing the effectiveness-efficiency objective under a fixed horizon, with robustness further enhanced by a dual-defense protection mechanism.

Recent work has explored automated MAS construction and workflow search (Zhuge et al., 2024; Zhang et al., 2025d; Yuan et al., 2025; Hu et al., 2025; Zhang et al., 2025b,a; Ke et al., 2025; Yang et al., 2025), topology learning and communication optimization (Li et al., 2026; Yue et al., 2025; Li et al., 2025a; Zhou et al., 2025; Feng et al., 2025; Chen et al., 2025), FSM-driven system synthesis (Zhang et al., 2025e; Wu et al., 2024b; Liu et al., 2023; Xinjie et al., 2025), and robustness-oriented defenses (Wang et al., 2025; Li et al., 2025b). However, these directions are typically studied in isolation, lacking a unified learning-based coordination framework.

To address these challenges, we propose NeuralFSM: a framework that uses Temporal Graph Networks (TGN) (Rossi et al., 2020) as a task-adaptive finite-state temporal coordination controller to learn multi-agent coordination. TGNs extend graph neural networks to dynamic graphs evolving by updating node memories through temporal message passing. Our key insight is that inter-agent coordination naturally forms a dynamic interaction graph, making TGNs well suited for learning finite-state execution policies.

NeuralFSM performs task-conditioned coordination modeling through a reusable FSM-based coordination structure and a learned temporal execution policy. The controller conditions TGN-based predictors on task representations and execution context, enabling adaptive state transitions and communication routing without manually designed protocols or task-specific rules. Unlike workflow-based or graph-based coordination methods that often rely on static or task-specific interaction structures, NeuralFSM provides a more reusable and temporally adaptive coordination mechanism. By learning probabilistic distributions over both state transitions and inter-agent communications from interaction traces within a Finite State Machine (FSM), NeuralFSM captures shared execution regularities across tasks while producing structured and controllable execution trajectories.

NeuralFSM also includes a dual-defense protection layer, which regularizes anomalous communications during training and attenuates low-trust messages at runtime, improving robustness without altering the normal coordination mechanism.

Our main contributions are summarized as follows:

**Adaptive FSM traversal.** We cast coordination as traversing a finite state machine and learn a history-aware temporal coordination controller that jointly predicts state transitions and sparse communication routing from interaction traces.

**Dual-Defense Protection.** We introduce a protection layer combining training-time graph regularization with runtime trust-aware message attenuation to mitigate frequency and semantic attacks.

**Empirical evaluation of effectiveness, efficiency, and robustness.** Across six benchmarks, NeuralFSM achieves the best average performance, surpassing prior baselines by $6.74\% \sim 19.39\%$, while reducing token cost via sparse routing and improving robustness under attack.

## 2 Related Work

**Automated MAS Design and Structured Coordination.** Prior work constructs LLM-based multi-agent systems through workflow-oriented protocols and predefined templates (e.g., CAMEL, AutoGen, MetaGPT, and ChatDev) (Li et al., 2023; Wu et al., 2024a; Hong et al., 2024; Qian et al., 2024), as well as automated workflow search and composition methods (Zhang et al., 2025e; Hu et al., 2025; Yuan et al., 2025; Zhang et al., 2025a,d,b; Ke et al., 2025; Yang et al., 2025). Another related line of work explores graph-structured prompting and reasoning graph frameworks that facilitate multi-step reasoning and task decomposition (Besta et al., 2024; Jin et al., 2024; Pandey et al., 2025). A complementary line of work focuses on learning or optimizing inter-agent communication topologies (Zhang et al., 2025c; Li et al., 2026; Yue et al., 2025; Zhou et al., 2025; Li et al., 2025a; Chen et al., 2025; Feng et al., 2025; Xinjie et al., 2025). In contrast, NeuralFSM does not produce a fixed workflow or topology; instead, it learns a history-aware controller that *traverses* a reusable FSM via task-conditioned state transitions and sparse routing decisions.

**Robustness and Security.** Prior work studies the safety and robustness of LLM-based MAS from multiple perspectives, including topology-aware safety analyses (Yu et al., 2025; Wang et al., 2025), defenses against unknown or evolving attacks (Miao et al., 2025; Zeng et al., 2024), trust management (He et al., 2025; Amayuelas et al., 2024), and anomaly-aware mitigation for unreliable communications (Li et al., 2025b).

## 3 Problem Formulation

Given a domain problem $q \in \Sigma$, a finite state machine autonomously generates a set of states $\mathcal{S} = \{s_0, s_1, \ldots, s_{n-1}\}$ and a corresponding set of agents $\mathcal{A} = \{a_0, a_1, \ldots, a_{m-1}\}$ from the problem description. Our goal is to learn a coordination policy within the FSM that reliably solves $q$ based on generated states, while minimizing unnecessary communication and tool or LLM calls.

### 3.1 Coordination policy

NeuralFSM executes in discrete steps $t \in \{0, \ldots, T_{\max} - 1\}$ with state $s_t \in \mathcal{S}$. At each step, the controller makes two coupled decisions: **state transition**, i.e., sampling the next state $s_{t+1}$; and **communication routing**, i.e., selecting a sparse receiver set $\{a_r\}_t \subseteq \mathcal{A}$ with $|\{a_r\}_t| \leq k$, which obtains the output of state $s_t$.

Let $\mathbf{q} = \mathrm{Enc}(q)$ denote the problem embedding and $\mathbf{c}_t$ denote the accumulated execution context at step $t$. The coordination policy is parameterized by $\theta$ and factorized as: $\pi_\theta(s_{t+1}, \{a_r\}_t \mid s_t, \mathbf{c}_t, \mathbf{q}) = P_\theta^{\text{trans}}(s_{t+1} \mid s_t, \mathbf{c}_t, \mathbf{q}) \cdot P_\theta^{\text{route}}(\{a_r\}_t \mid s_t, \mathbf{c}_t, \mathbf{q})$, where $P_\theta^{\text{trans}}$ and $P_\theta^{\text{route}}$ are distinct conditional distributions that share parameters $\theta$ through a common encoder, corresponding to state transition control and sparse communication routing.

Given $s_{t+1}$, the responsible agent $\dot{a}_{t+1} = \text{responsible}(s_{t+1})$, which may be reused across multiple states, is executed and produces an output $o_{t+1}$, which is routed to $\{a_r\}_{t+1}$ and used to update the context $\mathbf{c}_{t+1}$. A rollout yields a trajectory $\tau$ comprising accessed states, agent executions, routing decisions, and context updates, whose length is bounded by a fixed horizon $T_{\max}$.

### 3.2 Objective

We optimize a coordination policy that explicitly trades off performance and cost. Let $R(q, y, \tau)$ denote the execution reward of an FSM rollout $\tau$, provided by a task-specific evaluator $u(q, \tau)$ that compares the prediction $\hat{y}$ with the ground-truth $y$. Let $C(\tau)$ denote the execution cost incurred during the rollout. The learning objective is formulated as the maximization of an expected utility:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta}\big[R(q, y, \tau) - \nu C(\tau)\big], \quad (1)$$

where $\nu > 0$ controls the trade-off between performance and cost. In Section 4, we instantiate $\pi_\theta$ as a temporal coordination controller and optimize a differentiable multi-objective loss that jointly captures execution performance, communication sparsity, cost awareness, and robustness.

## 4 NeuralFSM

As shown in Fig. 2, this section presents NeuralFSM as a task-adaptive coordination mechanism

formalized as a finite-state process. It consists of three components: (i) an FSM backbone that defines the coordination search space, (ii) a temporal coordination controller that predicts state transitions and communication routing conditioned on the task and execution context, and (iii) a specific protection layer designed for adversarial settings.

### 4.1 FSM Backbone

Following the classical formulation of finite state machines (Moore, 1956), we define an FSM tuple extended with an explicit agent set and a communication mapping:

$$\text{FSM} = (\mathcal{S}, \Sigma, \mathcal{A}, \delta, \mathcal{L}, s_0, \mathcal{F}), \tag{2}$$

where $\mathcal{S}$ represents the set of states, corresponding to distinct phases in the problem-solving process, $\Sigma$ is the input alphabet representing the set of domain problems, $\delta : \mathcal{S} \times \mathcal{X} \to \mathcal{S}$ is the state transition function determining the next state given the current state and contextual information $\mathcal{X}$ accumulated during execution, $\mathcal{L} : \mathcal{S} \to 2^{\mathcal{A}}$ is the communication mapping enabling selective information flow based on relevance, $s_0 \in \mathcal{S}$ is the initial state and $\mathcal{F} \subseteq \mathcal{S}$ is the set of final states, and the rollout returns a final answer by executing a final state.

Neural finite state machine dynamically traverses a task-adaptive sequence of states $s_0 \to s_2 \to \cdots \to s_{n-1}$, with the ability to backtrack from the current state to a previously visited state. Both forward transitions and backtracking behaviors are determined by the controller $\pi_\theta$, as shown in Fig. 1.

### 4.2 Temporal Coordination Controller

Section 3 formulates coordination as state transitions with sparse routing. At step $t$, the controller produces (i) a transition distribution over $\mathcal{S}$ and (ii) communication weights over $\mathcal{A}$, both conditioned on feature embeddings and execution context:

$$P_\theta^{\text{trans}}(\tau) = \text{softmax}\big(f_\theta^{\text{trans}}(s_t, \mathbf{c}_t, \mathbf{q})\big), \tag{3}$$

$$\mathbf{w}_t(\tau) = f_\theta^{\text{comm}}(s_t, \mathbf{c}_t, \mathbf{q}) \in [0,1]^m, \tag{4}$$

where $\mathbf{w}_t(\tau)$ parameterizes the routing distribution $P_\theta^{\text{route}}$ induced by sparse sampling: $P_\theta^{\text{route}}(\{a_r\}_t \mid s_t, \mathbf{c}_t, \mathbf{q}) := \Pr\big(\text{Sample}_k(\mathbf{w}_t(\tau), k) = \{a_r\}_t\big)$. The receiver budget $k$ enforces communication sparsity and controls context growth.

The two heads $f_\theta^{\text{trans}}$ and $f_\theta^{\text{comm}}$ share a temporal interaction encoder that encodes how information propagates across agents over time. Static graph designs compress the entire interaction history into a single fixed topology, thereby discarding temporal credit assignment information and preventing the collaboration pattern from being adjusted according to task-solving progress, i.e., state. Consequently, NeuralFSM models the multi-agent collaboration pattern within each rollout $\tau$ as a time-stamped directed communication graph $G = (V, E_\tau)$, and learns a history encoder $\phi_\theta$ that maps past interactions on $G$ into per-agent memories, which are then used as representations for predicting the next state and communication routing decisions.

In a graph $G$, the node set $V$ corresponds to agents, and the time-stamped directed edge set $E_\tau$ represents routed communications. The controller maintains a memory state $\mathbf{m}_v(t)$ for each node $v$ and updates it through message passing over observed interactions. The resulting node representation is denoted as $\mathbf{h}_v(t) = g_\theta(\mathbf{m}_v(t), \mathbf{e}_v)$, where $\mathbf{e}_v$ is the embedding of node $v$ and $g_\theta$ is a learnable decoder.

**Temporal message.** For an interaction $(u \to v)$ at time $t$ with edge features $\mathbf{e}_{uv}$ and time gap $\Delta t_{uv}$, we compute

$$\mathbf{m}(u \to v, t) = \phi_\theta\big([\mathbf{h}_u(t), \mathbf{h}_v(t), \mathbf{e}_{uv}, \psi(\Delta t_{uv})]\big), \tag{5}$$

where $\psi(\Delta t)$ is a temporal encoding (Fourier features with learnable frequencies $\{\omega_\ell\}_{\ell=1}^{d_\psi}$):

$$\begin{aligned}\psi(\Delta t) = \big[&\cos(\omega_1 \Delta t), \sin(\omega_1 \Delta t), \ldots, \\ &\cos(\omega_{d_\psi} \Delta t), \sin(\omega_{d_\psi} \Delta t)\big].\end{aligned} \tag{6}$$

**Aggregation and update.** Incoming messages are aggregated to update memory:

$$\begin{aligned}\mathbf{M}_v^t &= \text{Agg}\big(\{\mathbf{m}(u \to v, t) : u \in \mathcal{N}(v)\}\big), \\ \mathbf{m}_v(t) &= \text{Upd}_\theta(\mathbf{m}_v(t^-), \mathbf{M}_v^t),\end{aligned} \tag{7}$$

where $t^-$ denotes the previous time step. We then derive per-agent features and a global interaction summary:

$$\begin{aligned}\mathbf{z}_{a_j}(t) &= p_\theta^a\big(\mathbf{h}_{a_j}(t)\big), \\ \mathbf{g}_t &= \text{Pool}\big(\{\mathbf{h}_{a_j}(t)\}_{j=0}^{m-1}\big),\end{aligned} \tag{8}$$

where $p_\theta^*(\cdot)$ is a projection and $\text{Pool}(\cdot)$ denotes attention pooling. We augment the execution context as

$$\widetilde{\mathbf{c}}_t = \big[\mathbf{c}_t, \mathbf{g}_t\big], \tag{9}$$

so the controller conditions on both $\mathbf{c}_t$ and interaction history.

**Transition head.** Let $\mathbf{e}_{s_t}$ be the embedding of state $s_t$. We compute transition logits by fusing state, task, and context features:

$$\begin{aligned}&\mathbf{z}_s = p_\theta^s(\mathbf{e}_{s_t}), \mathbf{z}_q = p_\theta^q(\mathbf{q}), \mathbf{z}_c = p_\theta^c(\widetilde{\mathbf{c}}_t),\\ &f_\theta^{\text{trans}}(s_t, \mathbf{c}_t, \mathbf{q}) = r_\theta^{\text{trans}}\big([\mathbf{z}_s, \mathbf{z}_q, \mathbf{z}_c]^T\big),\end{aligned} \tag{10}$$

which instantiates Eq. (3). Here, $r_\theta^{\text{trans}}(\cdot)$ is a learnable readout mapping the fused representation $[\mathbf{z}_s, \mathbf{z}_q, \mathbf{z}_c]$ to a $|\mathcal{S}|$-dimensional logit vector over next states. Similarly, $r_\theta^Q(\cdot)$ and $r_\theta^K(\cdot)$ are learnable readouts that produce query and key vectors for computing routing scores. In our implementation, these readouts are lightweight feed-forward networks whose parameters are included in $\theta$.

**Routing head.** We compute per-agent routing scores $w_{t,j} \in [0, 1]$ via query-key compatibility between the fused representation and per-agent features:

$$\mathbf{Q}_t = r_\theta^Q\big([\mathbf{z}_s, \mathbf{z}_q, \mathbf{z}_c]\big), \mathbf{K}_j(t) = r_\theta^K\big(\mathbf{z}_{a_j}(t)\big).$$

$$u_{t,j} = \frac{\mathbf{Q}_t^\top \mathbf{K}_j(t)}{\sqrt{d_k}}, w_{t,j} = \sigma\big(u_{t,j}\big), \tag{11}$$

where $\sigma(\cdot)$ is the sigmoid and $d_k$ is the query/key dimension. The vector $\mathbf{w}_t(\tau) = [w_{t,0}, ..., w_{t,m-1}]$ instantiates Eq. (4).

To optimize Eq. (1) with discrete state traversal, we use a Monte Carlo policy-gradient estimator (REINFORCE; (Williams, 1992)) with trajectory-level regularizers, yielding the multi-objective loss:

$$\begin{aligned}\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{state}} + \gamma\mathcal{L}_{\text{comm}} + \nu\mathcal{L}_{\text{cost}}\\ + \zeta\mathcal{L}_{\text{max}} + \lambda\mathcal{L}_{\text{protect}},\end{aligned} \tag{12}$$

where each term corresponds to a specific coordination requirement and is computed on the sampled trajectory $\tau$.

**Unified state-transition learning signal.** We merge an outcome-driven policy gradient signal with a transition-likelihood regularizer into a single state-transition loss:

$$\mathcal{L}_{\text{state}} = -\sum_{t=0}^{T_{\max}-1} \log P_\theta^{\text{trans}}(\tau) \cdot \big(\alpha(R - b) + \beta\big), \tag{13}$$

where $b$ is the average reward based on historical batches. The term $\alpha(R - b)$ provides outcome-driven credit assignment, while $\beta$ stabilizes learning under sparse rewards.

**Routing likelihood.** Let $y_{t,j} \in \{0, 1\}$ indicate whether agent $a_j$ is selected as a receiver at step $t$. Given routing scores $w_{t,j}$, we use a Bernoulli cross-entropy objective:

$$\begin{aligned}\mathcal{L}_{\text{comm}} = -\sum_{t=0}^{T_{\max}-1}\sum_{j=0}^{m-1}\Big(y_{t,j}\log(w_{t,j} + \epsilon)\\ + (1 - y_{t,j})\log(1 - w_{t,j} + \epsilon)\Big).\end{aligned} \tag{14}$$

**Cost and horizon regularization.** We set $\mathcal{L}_{\text{cost}} = (C(\tau) - C_{\text{baseline}})^2$ with a running baseline $C_{\text{baseline}}$, and use $\mathcal{L}_{\text{max}} = \mathbb{1}[|\tau| = T_{\max}]$ to discourage degenerate long traversals.

$\mathcal{L}_{\text{protect}}$ is enabled only when the protection module is active. The complete optimization and execution procedures in this section are provided in Appendix B (Algorithms 1–2).

### 4.3 Protection Layer

To address robustness issues arising from adversarial agents, we augment NeuralFSM with a protection layer.

**Threat model.** A subset of agents may be compromised and can affect coordination via the communication channel of the induced graph $G$. We consider two representative attacks: (i) frequency attacks, where compromised agents inject messages at an abnormally high rate, and (ii) semantic attacks, where compromised agents inject misleading content into the execution context $\mathbf{c}_t$. We parameterize attacks by an attacked-agent ratio $\rho \in (0, 1]$ and an attack strength $\eta > 0$.

The protection layer computes (i) a structural priority $\pi(i)$ on $G$ and (ii) an online anomaly score $\varphi(i, t)$, then derives a bounded trust score $\text{trust}(i, t) \in [0, 1]$ used for runtime attenuation and training-time regularization.

**Centrality analysis.** On the induced directed communication graph $G(V, E_\tau)$, we compute a priority score from two centrality measures, betweenness centrality $\text{BC}(i)$ and PageRank $\text{PR}(i)$, whose formal definitions are provided in Appendix D.1:

$$\pi(i) = w_{\text{BC}}\,\text{BC}(i) + w_{\text{PR}}\,\text{PR}(i), \tag{15}$$

where $w_{\text{BC}}, w_{\text{PR}} \geq 0$ and $w_{\text{BC}} + w_{\text{PR}} = 1$. The score $\pi(i)$ serves as a graph-centric prior on structural influence.

**Anomaly detection.** We compute an online anomaly score from two complementary observable anomalous signals, frequency anomaly $\varphi_{\text{freq}}$ and semantic anomaly $\varphi_{\text{sem}}$, whose computations are described in Appendix D.2:

$$\varphi(i,t) = \sigma\Big(\omega_{\text{freq}}\varphi_{\text{freq}}(i,t) + \omega_{\text{sem}}\varphi_{\text{sem}}(i,t)\Big), \tag{16}$$

where $\omega_{\text{freq}}, \omega_{\text{sem}} \geq 0$ are fusion weights, and $\omega_{\text{freq}} + \omega_{\text{sem}} = 1$.

We convert $\pi(i)$ and $\varphi(i,t)$ into a bounded trust score that penalizes anomalous senders while accounting for the graph structure:

$$\text{trust}(i,t) = \sigma\big(\pi(i)\big) \cdot \big(1 - \varphi(i,t)\big). \tag{17}$$

**Dual-defense strategy.** We deploy the protection layer in two complementary ways. (i) **Runtime defense**. The protection layer attenuates message representations produced by the temporal coordination controller using sender trust:

$$\widetilde{\mathbf{m}}(u \to v, t) = \text{trust}(u,t) \cdot \mathbf{m}(u \to v, t), \tag{18}$$

which reduces the influence of low-trust senders while preserving useful signals in the clean setting. (ii) **Training-time defense**. During training, a protection regularizer $\mathcal{L}_{\text{protect}}$ (see Appendix D.3) discourages strong messages from low-trust sources.

# 5 Experiments

## 5.1 Experiment Setup

**Tasks and Benchmarks.** We evaluate NeuralFSM on a diverse set of domains spanning mathematical reasoning, GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021); code synthesis, HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021); knowledge-intensive QA, GPQA (Rein et al., 2024); multi-hop QA, HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022); embodied decisions, ALFWorld (Shridhar et al., 2020); and tool use, GAIA (Mialon et al., 2024).

**Baselines.** We compare against three baseline families: Single-agent prompting: IO (OpenAI, 2025), CoT (Wei et al., 2022), and self-consistency (CoT×5) (Wang et al., 2022). Hand-crafted MAS: DyLAN (Liu et al., 2024), LLM-Debate (Du et al., 2024), AgentVerse (Chen et al., 2024b), and MacNet (Qian et al., 2025). Automatically designed MAS: GPTSwarm (Zhuge et al., 2024), AutoAgents (Chen et al., 2024a), AFlow (Zhang et al., 2025d), G-Designer (Zhang et al., 2025c), MaAS (Zhang et al., 2025b), and MasHost (Yang et al., 2025). For multi-hop QA, we additionally include strong reasoning baselines KAG (Liang et al., 2025), CoA (Zhang et al., 2024), and ReAgent (Xinjie et al., 2025).

**LLM Backbones.** We use four closed-source models in our paper: `gpt-5-nano`, `gpt-4o-mini`, `gpt-4.1-nano`, and `grok-3-mini`. All LLMs are accessed via APIs, with the temperature set to 1.

**Parameter Configuration.** Unless otherwise stated, we use a rollout horizon $T_{\max} = 8$ and a sparse receiver budget $k = 3$. We use all-MiniLM-L6-v2 (Wang et al., 2020) to instantiate the embedding function $\text{Enc}(\cdot)$. Loss weights follow Section 4.2 with $(\alpha, \beta, \gamma, \nu, \zeta, \lambda) = (1.0, 0.3, 0.2, 0.1, 0.5, 0.3)$. For attack settings, we set $(\rho, \eta) = (0.5, 3.0)$ by default. For the protection layer, the following parameters are fixed across all robustness runs: $(w_{\text{BC}}, w_{\text{PR}}) = (0.6, 0.4)$, $d_{\text{PR}} = 0.85$, $(\omega_{\text{freq}}, \omega_{\text{sem}}) = (0.3, 0.7)$.

## 5.2 Performance Analysis

We compare NeuralFSM with 13 baselines on the GSM8K, MATH, GPQA, HumanEval, MBPP and ALFWorld benchmarks in Table 1, and with 7 baselines on multi-hop QA benchmarks in Table 2. In addition, we analyze the backbone sensitivity of NeuralFSM in Table 5. The experimental results are analyzed as follows:

**NeuralFSM achieves the best overall performance across six benchmarks**. Compared to state-of-the-art (SOTA) designed MAS, NeuralFSM produces an average improvement of 6.74%; relative to the single-agent IO baseline, the average gain increases to 19.39%. The improvements are most pronounced in benchmarks with high interaction demands. For example, on MATH, NeuralFSM outperforms IO by 21.69% and the SOTA baseline by 12.75%. On GPQA and ALFWorld, it further surpasses the SOTA by 14.22% and 5.63%, respectively. These results are consistent with the advantages conferred by task-conditioned finite-state traversal.

**On three multi-hop QA benchmarks, NeuralFSM achieves the best overall performance**. Compared to the strongest multi-hop baseline ReAgent, NeuralFSM achieves an average improvement of 15.21%, with consistent gains of 6.29% on HotpotQA, 13.88% on 2Wiki, and 32.98% on

| Methods | GSM8K | MATH | GPQA | HumanEval | MBPP | ALFWorld | Avg. |
|---|---|---|---|---|---|---|---|
| IO(OpenAI, 2025) | 92.19 | 76.31 | 48.99 | 90.15 | 78.73 | 41.63 | 71.33 |
| CoT(Wei et al., 2022) | 91.95$_{-0.24}$ | 76.12$_{-0.19}$ | 49.08$_{+0.09}$ | 91.26$_{+1.11}$ | 78.52$_{-0.21}$ | 42.81$_{+1.18}$ | 71.62$_{+0.29}$ |
| SC (CoT×5)(Wang et al., 2022) | 92.76$_{+0.57}$ | 77.85$_{+1.54}$ | 49.17$_{+0.18}$ | 91.49$_{+1.34}$ | 80.24$_{+1.51}$ | 43.42$_{+1.79}$ | 72.49$_{+1.16}$ |
| DyLAN(Liu et al., 2024) | 94.72$_{+2.53}$ | 78.41$_{+2.10}$ | 50.36$_{+1.37}$ | 91.70$_{+1.55}$ | 83.90$_{+5.17}$ | 55.10$_{+13.47}$ | 75.70$_{+4.37}$ |
| LLM-Debate(Du et al., 2024) | 94.55$_{+2.36}$ | 77.94$_{+1.63}$ | 50.24$_{+1.25}$ | 91.38$_{+1.23}$ | 84.11$_{+5.38}$ | 48.67$_{+7.04}$ | 74.48$_{+3.15}$ |
| AgentVerse(Chen et al., 2024b) | 94.85$_{+2.66}$ | 77.56$_{+1.25}$ | 51.53$_{+2.54}$ | 92.13$_{+1.98}$ | 82.45$_{+3.72}$ | 47.39$_{+5.76}$ | 74.32$_{+2.99}$ |
| MacNet(Qian et al., 2025) | 93.03$_{+0.84}$ | 75.29$_{-1.02}$ | 50.75$_{+1.76}$ | 89.66$_{-0.49}$ | 74.27$_{-4.46}$ | 47.01$_{+5.38}$ | 71.67$_{+0.34}$ |
| GPTSwarm(Zhuge et al., 2024) | 93.74$_{+1.55}$ | 78.09$_{+1.78}$ | 52.65$_{+3.66}$ | 91.81$_{+1.66}$ | 84.39$_{+5.66}$ | 55.88$_{+14.25}$ | 76.09$_{+4.76}$ |
| AutoAgents(Chen et al., 2024a) | 92.68$_{+0.49}$ | 74.63$_{-1.68}$ | 52.73$_{+3.74}$ | 90.30$_{+0.15}$ | 78.53$_{-0.20}$ | 49.75$_{+8.12}$ | 73.10$_{+1.77}$ |
| AFlow(Zhang et al., 2025d) | 95.50$_{+3.31}$ | 80.96$_{+4.65}$ | 53.83$_{+4.84}$ | 93.10$_{+2.95}$ | 88.31$_{+9.58}$ | 61.94$_{+20.31}$ | 78.94$_{+7.61}$ |
| G-Designer(Zhang et al., 2025c) | 96.17$_{+3.98}$ | 80.78$_{+4.47}$ | 53.06$_{+4.07}$ | 91.47$_{+1.32}$ | 87.87$_{+9.14}$ | 54.32$_{+12.69}$ | 77.28$_{+5.95}$ |
| MaAS(Zhang et al., 2025b) | 97.09$_{+4.90}$ | 81.22$_{+4.91}$ | 54.91$_{+5.92}$ | 94.72$_{+4.57}$ | 88.16$_{+9.43}$ | 62.55$_{+20.92}$ | 79.78$_{+8.45}$ |
| MasHost(Yang et al., 2025) | 98.41$_{+6.22}$ | 82.36$_{+6.05}$ | 55.28$_{+6.29}$ | 93.15$_{+3.00}$ | 87.93$_{+9.20}$ | 62.27$_{+20.64}$ | 79.90$_{+8.57}$ |
| **NeuralFSM (Ours)** | **98.81**$_{+6.62}$ | **92.86**$_{+16.55}$ | **63.14**$_{+14.15}$ | **98.73**$_{+8.58}$ | **91.36**$_{+12.63}$ | **66.07**$_{+24.44}$ | **85.16**$_{+13.83}$ |

Table 1: Performance comparison across benchmarks with single-agent systems, hand-crafted multi-agent systems, and automatically designed multi-agent systems. `gpt-5-nano` is used as the LLM backbone for all baselines. We **bold** the best results and underline the runners-up.

| Models / Methods | HotpotQA | 2Wiki | MuSiQue | Avg. |
|---|---|---|---|---|
| GPT-4o-mini | 35.97 | 50.43 | 23.06 | 36.49 |
| GPT-4.1-nano | 37.92 | 51.25 | 24.54 | 37.90 |
| GPT-5-nano | 43.69 | 57.11 | 29.68 | 43.49 |
| Grok-3-mini | 49.71 | 59.39 | 31.05 | 46.72 |
| CoA(Zhang et al., 2024) | 44.54 | 58.19 | 30.82 | 44.52 |
| KAG(Liang et al., 2025) | 58.70 | 65.32 | 32.25 | 52.09 |
| ReAgent(Xinjie et al., 2025) | 62.16 | 70.40 | 36.63 | 56.40 |
| **NeuralFSM (Ours)** | 66.07 | 80.17 | 48.71 | 64.98 |

Table 2: Performance of different models and methods across three multi-hop QA datasets. All four LLM-based methods use `gpt-5-nano` as the backbone.

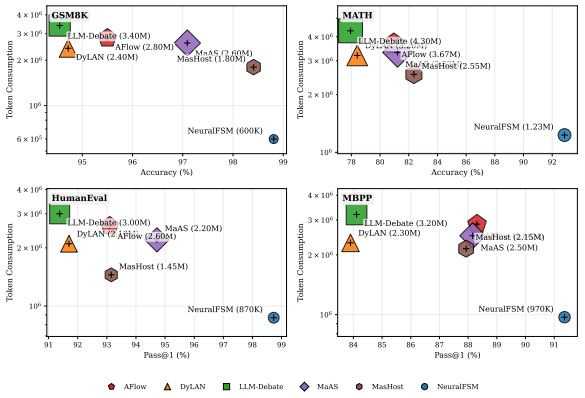


Figure 3: Performance-cost evaluation. Accuracy/-pass@1 versus token consumption compared to representative baselines.

MuSiQue. Notably, the performance of single-model prompting varies substantially with the choice of backbone (avg. 36.49–46.72), whereas the coordinated execution enabled by NeuralFSM yields a markedly larger performance improvement (45.96% over CoA) than simply switching to a stronger backbone.

**On GAIA, NeuralFSM benefits from stronger LLM backbones while effectively controlling cost.** The average score increases from 4.18 (`gpt-4o-mini`) to 12.75 (`gpt-5-nano`), corresponding to a 205.0% relative improvement, while the cost increases by only 135.0%. Switching to `grok-3-mini` yields a modest additional performance gain of 17.5% over `gpt-5-nano`, but nearly doubles the cost. Especially at Level 2, performance improves by only 10.0% while the cost increases by 80.5%. At Level 3, performance even drops by 49.9%, despite a 147.6% increase in cost. These results indicate that Levels 2–3 remain challenging across different backbones, suggesting that the remaining failures are primarily attributable to limitations of the underlying base models rather than coordination collapse.

### 5.3 Cost Analysis

NeuralFSM improves the effectiveness–efficiency frontier across representative benchmarks in Fig. 3. On GSM8K, it achieves comparable accuracy while using 66.67% fewer tokens than MasHost. On MATH, NeuralFSM improves accuracy by 12.77% while using 51.76% fewer tokens than MasHost. Similar Pareto-dominant behavior is observed on HumanEval, where NeuralFSM reduces token usage by approximately 60.45% compared to MaAS while improving pass@1 by 4.23%,

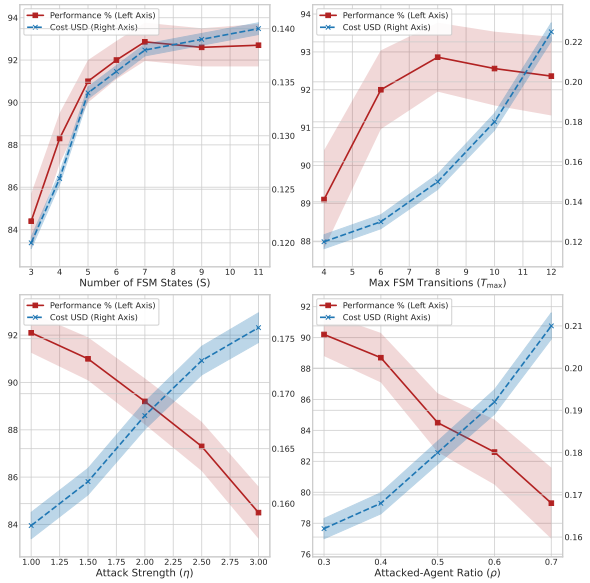


Figure 4: Parameter sensitivity of NeuralFSM. We vary FSM size $S$, rollout horizon $T_{\max}$, attack strength $\eta$, and attacked-agent ratio $\rho$, reporting performance and cost.

| Datasets | GSM8K | | MATH | |
|---|---|---|---|---|
| | Perf. | Cost ($) | Perf. | Cost ($) |
| NeuralFSM | 98.81 | 0.164 | 92.86 | 0.395 |
| w/o Protection | 98.67 | 0.159 | 92.78 | 0.389 |
| w/o TGN | 96.52 | 0.206 | 89.35 | 0.471 |
| w/o Trans | 96.13 | 0.197 | 87.64 | 0.623 |
| w/o Comm | 96.03 | 0.192 | 88.61 | 0.432 |

Table 3: Ablation study of NeuralFSM.

as well as on MBPP, where it uses about 66.08% fewer tokens than AFlow and improves pass@1 by 3.45%. These results support our claim that task-conditioned traversal combined with sparse routing effectively reduces redundant deliberation and uncontrolled context growth.

### 5.4 Framework Analysis

**Sensitivity Analysis.** As shown in Fig. 4, increasing the FSM size and rollout horizon reveals clear diminishing returns. Increasing the FSM size from $S = 3$ to $S = 7$ yields a 10.02% relative performance improvement at the expense of a 15.21% increase in cost, whereas further enlarging the state space provides negligible additional benefit. Similarly, extending the rollout horizon from $T_{\max} = 4$ to $T_{\max} = 8$ improves performance by 4.22%, while increasing $T_{\max}$ beyond 8 leads to a 0.54% performance drop accompanied by a sharp 50.45% increase in cost, suggesting that adopting $T_{\max} = 8$ as the default setting in this paper is reasonable. Under attack settings without protection, increasing the attack strength $\eta$ and the attacked-agent ratio $\rho$ monotonically degrades performance while increasing cost. Specifically, raising $\eta$ from 1.0 to 3.0 results in an 8.25% performance decrease and an 11.39% increase in cost, while increasing $\rho$ from 0.3 to 0.7 causes performance to drop by 12.08% and cost to rise by 29.63%.

**Ablation Study.** Table 3 shows that removing the protection layer under no-attack settings leads to almost no change in either performance or cost for NeuralFSM, suggesting that the protection mechanism does not interfere with normal execution. Replacing TGN with a standard GNN also weakens coordination, with a 3.78% drop in performance on MATH and a 25.61% increase in cost on GSM8K, suggesting that temporal interaction modeling is important when coordination patterns change across reasoning stages. Disabling transition prediction significantly degrades performance, resulting in a 5.62% drop in accuracy and a 57.72% increase in cost on MATH. This variant resembles graph-based coordination in that it samples sparse inter-agent communication. Without explicit state transitions, reasoning progress remains implicit in the interaction graph, which hurts long-horizon problem solving. Disabling communication sampling also reduces performance, with a 2.82% decrease in accuracy and a 17.07% increase in cost on GSM8K. This variant resembles workflow-based coordination, as it preserves state transitions while making information flow fixed rather than state-adaptive. These results suggest that NeuralFSM benefits from jointly learning adaptive state transitions and adaptive communication routing.

### 5.5 Robustness Analysis

As shown in Fig. 5, the protection layer provides nearly complete isolation of NeuralFSM from communication channel attacks, and even without a protection layer, NeuralFSM demonstrates strong inherent robustness to attacks. Under attack, all baselines suffer substantial relative performance degradations: DyLAN drops by 17.11%, MaAS by 26.47% and MasHost by 18.60%. In contrast, NeuralFSM exhibits markedly greater resilience without protection, with a relative performance drop of 9.31%. Enabling the proposed protection layer further improves accuracy by 8.26% relative to the unprotected, reducing the remaining gap to clean

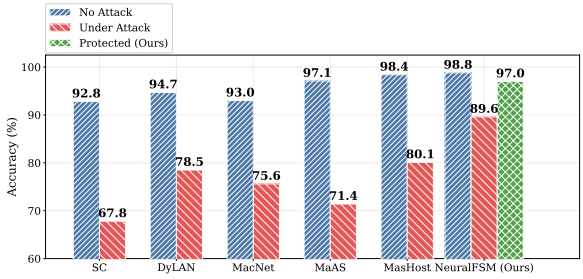


Figure 5: Robustness of NeuralFSM and other baselines under attack on GSM8K: accuracy in the clean setting, under attack, and with our proposed protection layer enabled.

performance to just 1.82%. These results are consistent with our dual-defense design, indicating that the protection layer effectively mitigates the influence of abnormal messages without disrupting the coordination mechanism. Additional robustness experiments are shown in Appendix E.2.

### 5.6 Case Study

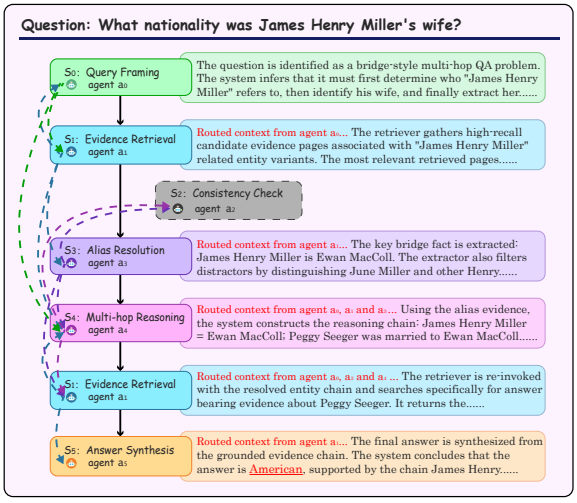


Figure 6: Case study of NeuralFSM on HotpotQA, showing the executed state trajectory and communication routing among agents for a representative question-answering instance.

To further illustrate how NeuralFSM behaves in concrete problem-solving scenarios, Fig. 6 presents a HotpotQA example. The figure visualizes both the executed state trajectory and the communication routing among agents. Rather than following a fixed left-to-right pipeline, NeuralFSM activates only a subset of the available states for this instance, with each next state selected from a learned transition distribution. The routed information is passed selectively to agents that may benefit from it, reducing irrelevant communication while preserving a coherent reasoning trace. Notably, although the state inventory includes a consistency-check state, this example reaches a sufficiently grounded answer without activating it, illustrating that unnecessary states can be skipped when the current execution path already provides sufficient support.

## 6 Conclusion

In this paper, we present NeuralFSM, a task-adaptive coordination framework that executes a finite-state process per problem and learns both state transitions and sparse communication routing via a temporal coordination controller. Unlike approaches that rely on a fixed collaboration topology, NeuralFSM generates time-step execution trajectories specific to the internal state of a reusable FSM, enabling adaptive execution policies that balance effectiveness and efficiency. Experiments across a broad range of benchmarks demonstrate that NeuralFSM achieves superior performance at markedly lower token cost. The protection layer further enhances robustness during coordination under adversarial attacks.

## Limitations

Although NeuralFSM outperforms fixed-protocol orchestration and other parameter optimization methods in both performance and cost by training a task-conditional coordinator to learn finite-state execution policies, ensuring policy stability and transferability still depends on appropriate hyperparameter settings and sufficient interaction traces. Especially under varying task distributions, further work is needed to automatically match each task domain with appropriate agent scales, trajectory lengths, and prompt templates, thereby achieving stronger cross-domain generalization capabilities; this is precisely one of our next research priorities. Moreover, the benchmarks adopted in our experiments are not particularly challenging for contemporary LLM backbones and are primarily selected to align with prior baselines. Finally, robustness remains an open and evolving challenge: as attack strategies continue to develop, corresponding defense mechanisms may require continual refinement to maintain reliable deployment in open environments.

## Acknowledgments

This work is supported by Beijing Natural Science Foundation (L251061) and the Industrial Foundation Reengineering and High-Quality Manufacturing Development project (ZC25T320057/100).

## References

Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Yang Wang. 2024. Multiagent collaboration attack: Investigating adversarial attacks in large language model collaborations via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6929–6948.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17682–17690.

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F Karlsson, Jie Fu, and Yemin Shi. 2024a. Autoagents: A framework for automatic agent generation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 22–30.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Cheng Qian, Chi-Min Chan, Yujia Qin, Ya-Ting Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024b. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing effectiveness and efficiency for llm-based multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11534–11557.

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. *arXiv preprint arXiv:2110.14168*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*.

Shangbin Feng, Zifeng Wang, Palash Goyal, Yike Wang, Weijia Shi, Huang Xia, Hamid Palangi, Luke Zettlemoyer, Yulia Tsvetkov, Chen-Yu Lee, and Tomas Pfister. 2025. Heterogeneous swarms: Jointly optimizing model roles and weights for multi-llm systems. In *Advances in Neural Information Processing Systems*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pages 8048–8057.

Pengfei He, Zhenwei Dai, Xianfeng Tang, Yue Xing, Hui Liu, Jingying Zeng, Qiankun Peng, Shrivats Agrawal, Samarth Varshney, Suhang Wang, Jiliang Tang, and Qi He. 2025. Attention knows whom to trust: attention-based trust management for llm multi-agent systems. *arXiv preprint arXiv:2506.02546*.

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. *arXiv preprint arXiv:2103.03874*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184.

Zixuan Ke, Austin Xu, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. Mas-zero: Designing multi-agent systems with zero supervision. *arXiv preprint arXiv:2505.14996*.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. 2025a. Adaptive graph pruning for multi-agent communication. *arXiv preprint arXiv:2506.02951*.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. 2026. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23142–23150.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. 2024. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9.

Zherui Li, Yan Mi, Zhenhong Zhou, Houcheng Jiang, Guibin Zhang, Kun Wang, and Junfeng Fang. 2025b. Goal-aware identification and rectification of misinformation in multi-agent systems. *arXiv preprint arXiv:2506.00509*.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2025. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 334–343.

Jia Liu, Jie Shuai, and Xiyao Li. 2023. State machine of thoughts: Leveraging past reasoning trajectories for enhancing problem solving. *arXiv preprint arXiv:2312.17445*.

Jie Liu, Guohua Wang, Ronghui Yang, Jiajie Zeng, Mengchen Zhao, and Yi Cai. 2025. Rtadev: Intention aligned multi-agent framework for software development. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1548–1581.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2024. A dynamic llm-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2024. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Rui Miao, Yixin Liu, Yili Wang, Xu Shen, Yue Tan, Yiwei Dai, Shirui Pan, and Xin Wang. 2025. Blindguard: Safeguarding llm-based multi-agent systems under unknown attacks. *arXiv preprint arXiv:2508.08127*.

Edward F Moore. 1956. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153.

OpenAI. 2025. Gpt-5 nano model documentation. Accessed: 2025-10-01.

Tushar Pandey, Ara Ghukasyan, Oktay Goktas, and Santosh Kumar Radha. 2025. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *arXiv preprint arXiv:2502.05078*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*.

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

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Shilong Wang, Guibin Zhang, Miao Yu, Guancheng Wan, Fanci Meng, Chongye Guo, Kun Wang, and Yang Wang. 2025. G-safeguard: A topology-guided security lens and treatment on llm-based multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7261–7276.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang (Eric) Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Ahmed Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2024a. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. 2024b. Stateflow: Enhancing llm task-solving through state-driven workflows. *arXiv preprint arXiv:2403.11322*.

Zhao Xinjie, Fan Gao, Xingyu Song, Yingjian Chen, Rui Yang, Yanran Fu, Yuyang Wang, Yusuke Iwasawa, Yutaka Matsuo, and Irene Li. 2025. Reagent: Reversible multi-agent reasoning for knowledge-enhanced multi-hop qa. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4067–4089.

Kuo Yang, Xingjie Yang, Linhui Yu, Qing Xu, Yan Fang, Xu Wang, Zhengyang Zhou, and Yang Wang. 2025. Mashost builds it all: Autonomous multi-agent system directed by reinforcement learning. *arXiv preprint arXiv:2506.08507*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Miao Yu, Shilong Wang, Guibin Zhang, Junyuan Mao, Chenlong Yin, Qijiong Liu, Kun Wang, Qingsong Wen, and Yang Wang. 2025. Netsafe: Exploring the topological safety of multi-agent system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2905–2938.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deqing Yang. 2025. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6192–6217.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. 2025. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15549–15572.

Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. Autodefense: Multi-agent llm defense against jailbreak attacks. *arXiv preprint arXiv:2403.04783*.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. 2025a. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373*.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. 2025c. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2025d. Aflow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Yaolun Zhang, Xiaogeng Liu, and Chaowei Xiao. 2025e. Metaagent: Automatically constructing multi-agent systems based on finite state machines. In *Proceedings of the 42nd International Conference on Machine Learning*.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. 2025. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. Gptswarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning*.

# A Notation

We present a comprehensive review of commonly used notation and its definitions in Table 10.

# B Algorithm

We provide the complete pseudocode referenced in Section 4. The algorithmic steps are consistent with the coordination policy and the learning objective.

**Algorithm 1** performs end-to-end learning of the task-conditioned coordination policy described in Section 3. Each episode rolls out a trajectory $\tau$ by repeatedly sampling the next state from $P_\theta^{\text{trans}}(\tau)$ and selecting a sparse receiver set from $\mathbf{w}_t(\tau)$, while executing the responsible agent of the visited state and updating the shared context $\mathbf{c}_t$. The parameters $\theta$ are optimized with the integrated objective $\mathcal{L}_{\text{total}}$ in Section 4.2, combining state-control learning, routing regularization, cost-aware shaping, and robustness regularization.

**Algorithm 2** is the deployment-time executor. Given an input $q$, it traverses the master FSM for up to $T_{\max}$ steps, producing a sequence of states and sparse communications that is conditioned on the evolving context $\mathbf{c}_t$. The resulting execution log exposes which state was chosen, which agent acted, and which receivers were routed at each step, enabling both interpretability and controllable decoding.

**Algorithm 3** describes the protection layer in Section 4.3. It computes a graph-centric prior $\pi(i)$ from centrality on the induced communication graph and an online anomaly score $\varphi(i,t)$ from frequency and semantic signals, then converts them into a bounded trust score $\text{trust}(i,t)$. At runtime, the controller attenuates message representations through $\widetilde{\mathbf{m}}(\cdot,t)$ *before* producing state-transition and routing predictions, reducing the influence of suspicious senders while preserving the coordination mechanism used in the clean setting. The protection loss $\mathcal{L}_{\text{protect}}$ is used only during training as a regularizer and is not computed in this forward pass routine.

# C Dataset Statistics and Splits

Table 4 summarizes the dataset statistics used in our pipeline, where all datasets are initially obtained from authenticated Hugging Face repositories, and we have not used this data for commercial purposes. Unless otherwise specified, we construct a lightweight benchmark subset by stratified sampling based on the difficulty of different benchmarks and split them into train/test with a fixed ratio of 4:1 to standardize training signals and evaluation difficulty across domains. For train-only benchmarks (HumanEval, GPQA, and GAIA), the test size is marked as "–", with GAIA containing 165 questions ($53/86/26$ across three levels). For multi-hop QA tasks, EM denotes *Exact Match*, i.e., the percentage of predictions that match the ground-truth answer string exactly after standard normalization.

# D Protection Mechanism Implementation Methods

## D.1 Centrality Measures

The definitions of the two centrality measures used in the paper are as follows: First, Betweenness centrality:

$$\text{BC}(i) = \sum_{\substack{s,t \in V \\ s \neq i \neq t,\ s \neq t}} \frac{\sigma_{st}(i)}{\sigma_{st}}, \tag{19}$$

where $\sigma_{st}$ is the number of shortest paths from $s$ to $t$ and $\sigma_{st}(i)$ counts those paths that pass through node $i$. Second, PageRank with damping factor $d_{\text{PR}} \in (0,1)$:

$$\text{PR}(i) = \frac{1 - d_{\text{PR}}}{|V|} + d_{\text{PR}} \sum_{j \in \mathcal{N}_{\text{in}}(i)} \frac{\text{PR}(j)}{\max(1, \deg_{\text{out}}(j))} \tag{20}$$

, where $\mathcal{N}_{\text{in}}(i)$ denotes in-neighbors and $\deg_{\text{out}}(j)$ is out-degree.

## D.2 Anomalous Signals

The definitions and computations for the two anomalous signals are as follows: (i) Frequency anomaly. We define frequency anomaly as the case where an agent injects messages into the system at

**Algorithm 1** NeuralFSM Training

**Require:** Dataset $\mathcal{D} = \{(q, y)\}$; horizon $T_{\max}$; receiver budget $k$; loss weights $(\alpha, \beta, \gamma, \nu, \zeta, \lambda)$; learning rate $\eta_{\text{lr}}$; max epochs $E_{\max}$; batch size $B$

**Ensure:** Trained parameters $\theta$

Initialize master FSM $= (\mathcal{S}, \Sigma, \mathcal{A}, \delta, \mathcal{L}, s_0, \mathcal{F})$
Initialize $\theta$; initialize encoder $\mathrm{Enc}(\cdot)$; initialize optimizer with $\eta_{\text{lr}}$
Initialize baseline $b \leftarrow 0$; cost history cost_hist $\leftarrow [\,]$
**for** epoch = 1 to $E_{\max}$ **do**
  **for** minibatch $\{(q_i, y_i)\}_{i=1}^{B}$ **do**
    $L_{\text{batch}} \leftarrow 0$
    **for** $i = 1$ **to** $B$ **do**
      $\mathbf{q} \leftarrow \mathrm{Enc}(q_i)$; $s_t \leftarrow s_0$; $\mathbf{c}_t \leftarrow \emptyset$; $t \leftarrow 0$
      trans_hist $\leftarrow [\,]$; comm_hist $\leftarrow [\,]$; $C_{\text{episode}} \leftarrow 0$
      **while** $t < T_{\max}$ **do**
        $(P_\theta^{\text{trans}}(\tau), \mathbf{w}_t(\tau), \text{aux}) \leftarrow \mathrm{Controller}_\theta(s_t, \mathbf{c}_t, \mathbf{q})$
        Sample next state $s_{t+1} \sim P_\theta^{\text{trans}}(\tau)$; append $(t, s_t, s_{t+1})$ to trans_hist
        Let executing agent $\dot{a}_{t+1} \leftarrow \mathrm{responsible}(s_{t+1})$; $(o_{t+1}, \text{cost}) \leftarrow \dot{a}_{t+1}.\mathrm{execute}(q_i, \mathbf{c}_t)$
        $C_{\text{episode}} \leftarrow C_{\text{episode}} + \text{cost}$
        Sample receivers $\{a_r\}_t \leftarrow \mathrm{Sample}_k(\mathbf{w}_t(\tau), k)$
        For each $a_j \in \mathcal{A}$, record $y_{t,j} \leftarrow 1$ if $a_j \in \{a_r\}_t$ else 0; append $(t, a_j, w_{t,j}, y_{t,j})$ to comm_hist
        $\mathbf{c}_{t+1} \leftarrow \mathrm{UpdateContext}(\mathbf{c}_t, \{a_r\}_t, o_{t+1})$; $s_t \leftarrow s_{t+1}$; $t \leftarrow t + 1$
      **end while**
      Compute episode reward $R \leftarrow R(q_i, y_i, \tau)$ from the final output and evaluator
      Compute $\mathcal{L}_{\text{state}}, \mathcal{L}_{\text{comm}}, \mathcal{L}_{\text{cost}}, \mathcal{L}_{\max}, \mathcal{L}_{\text{protect}}$ following Section 4
      $\mathcal{L}_i \leftarrow \mathcal{L}_{\text{state}} + \gamma\mathcal{L}_{\text{comm}} + \nu\mathcal{L}_{\text{cost}} + \zeta\mathcal{L}_{\max} + \lambda\mathcal{L}_{\text{protect}}$
      $L_{\text{batch}} \leftarrow L_{\text{batch}} + \mathcal{L}_i$; append $C_{\text{episode}}$ to cost_hist; update baseline $b \leftarrow \mathrm{EMA}(b, R)$
    **end for**
    $L_{\text{batch}} \leftarrow L_{\text{batch}}/B$; update $\theta$ by one optimizer step using $\nabla_\theta L_{\text{batch}}$
  **end for**
**end for**
**return** $\theta$

an abnormal frequency. Let $n_i(t; W)$ be the number of messages sent by agent $i$ within a window of size $W$ ending at $t$, and let $\mu_W(t)$, $\sigma_W(t)$ be the mean and standard deviation of $\{n_j(t; W)\}_{j=0}^{m-1}$. We define

$$\varphi_{\text{freq}}(i, t) = \sigma\Big(\frac{n_i(t; W) - \mu_W(t)}{\sigma_W(t) + \epsilon}\Big), \tag{21}$$

where $\sigma(\cdot)$ is the sigmoid and $\epsilon > 0$ is a small constant. (ii) Semantic anomaly. We define semantic anomaly as the case where an agent's execution output is inconsistent with the problem and the current context. Let $\mathrm{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|}$ be cosine similarity. For a sender message $M_{i,t}$, we compute

$$\varphi_{\text{sem}}(i, t) = 1 - \mathrm{sim}\Big(\mathrm{Enc}(M_{i,t}), \mathrm{Enc}([q, \mathbf{c}_t])\Big). \tag{22}$$

### D.3 Protection Loss

A protection loss function is used to regularize the communication graph during training by reducing anomalous communication routing.

$$\mathcal{L}_{\text{protect}} = \sum_{t=0}^{T_{\max}-1} \sum_{(u,v)\in E_\tau} \Big(\big(1 - \text{trust}(u, t)\big) \cdot \|\mathbf{m}(u \to v, t)\|^2\Big). \tag{23}$$

## E Supplementary Results

Performance is measured using accuracy or pass@1, both of which are reported as percentages (%), and all experimental results of the methods evaluated on the benchmarks are averaged over three runs. The parameter sensitivity analysis reflects the variance in the experimental results.

**Algorithm 2** Task-Adaptive FSM Execution

**Require:** Problem $q$; trained parameters $\theta$; master FSM $= (\mathcal{S}, \Sigma, \mathcal{A}, \delta, \mathcal{L}, s_0, \mathcal{F})$; horizon $T_{\max}$; receiver budget $k$
**Ensure:** Answer $\hat{y}$ and execution log

$\mathbf{q} \leftarrow \mathrm{Enc}(q)$; $s_t \leftarrow s_0$; $\mathbf{c}_t \leftarrow \emptyset$; log $\leftarrow [\,]$
**for** $t = 0$ to $T_{\max} - 1$ **do**
  $(P_\theta^{\text{trans}}(\tau), \mathbf{w}_t(\tau), \text{aux}) \leftarrow \text{Controller}_\theta(s_t, \mathbf{c}_t, \mathbf{q}_{\text{emb}})$
  Sample the next state $s_{t+1}$ from $P_\theta^{\text{trans}}(\tau)$; $\dot{a}_{t+1} \leftarrow \text{responsible}(s_{t+1})$
  $o_{t+1} \leftarrow \dot{a}_{t+1}.\text{execute}(q, \mathbf{c}_t)$
  $\{a_r\}_t \leftarrow \text{Sample}_k(\mathbf{w}_t(\tau), k)$
  $\mathbf{c}_{t+1} \leftarrow \text{UpdateContext}(\mathbf{c}_t, \{a_r\}_t, o_{t+1})$; append $(t, s_{t+1}, \dot{a}_{t+1}, \{a_r\}_t)$ to log
  $s_t \leftarrow s_{t+1}$
**end for**
$\hat{y} \leftarrow \text{Extract}(o_t)$; **return** $(\hat{y}, \log)$

**Algorithm 3** Protected Controller Forward Pass

**Require:** $s_t, \mathbf{c}_t, \mathbf{q}$; induced edges $E_\tau$ and messages; parameters $\theta$
**Ensure:** $(P_\theta^{\text{trans}}(\tau), \mathbf{w}_t(\tau), \text{aux})$

Build induced graph $G = (V, E_\tau)$
Compute priority $\pi(i)$ from centrality on $G$ (Eq. (15))
Compute anomalies $\varphi_{\text{freq}}(i,t), \varphi_{\text{sem}}(i,t)$ and fuse to $\varphi(i,t)$ (Eq. (16))
Compute trust$(i,t) = \sigma(\pi(i)) \cdot (1 - \varphi(i,t))$
Run base controller to compute raw messages $\mathbf{m}(u \to v, t)$ on $E_\tau$
Apply trust-aware attenuation: $\widetilde{\mathbf{m}}(u \to v, t) = \text{trust}(u,t) \cdot \mathbf{m}(u \to v, t)$
Read out predictions from attenuated messages to obtain $(P_\theta^{\text{trans}}(\tau), \mathbf{w}_t(\tau))$; set aux $\leftarrow \{\pi, \varphi, \text{trust}\}$
**return** $(P_\theta^{\text{trans}}(\tau), \mathbf{w}_t(\tau), \text{aux})$

### E.1 Backbone Sensitivity on GAIA

Table 5 reports the backbone sensitivity analysis of NeuralFSM on GAIA, corresponding to the discussion in Section 5.2.

### E.2 Additional Robustness Experiments

As shown in Table 6, the results on MBPP closely mirror the robustness trends observed in the main text. The protection layer attenuates suspicious messages before they enter the shared context, leading to additional robustness improvements for NeuralFSM. Under attack, all baseline methods experience pronounced relative performance degradations: SC decreases by 33.67%, DyLAN by 25.15%, MacNet by 27.58%, MaAS by 19.95%, and MasHost by 20.93%. In contrast, NeuralFSM exhibits a markedly smaller relative drop of 11.05%, and enabling the protection layer further improves pass@1 by 10.58% compared to the unprotected setting under attack, reducing the remaining gap to clean performance to only 1.64%.

#### E.2.1 Selfish Attack

**Threat model.** Beyond the frequency and semantic attacks considered in Section 4.3, we additionally study a selfish attack setting. Under this setting, a compromised agent reduces its contribution to the collaborative process by limiting communication with other agents and providing incomplete outputs. This attack is particularly harmful for NeuralFSM, since the framework relies on state-conditioned transitions and selective communication routing.

**Anomaly detection.** We use the same protection mechanism as in Appendix D. Although selfish attacks differ from frequency flooding and semantic corruption, they can still be captured by the existing dual anomaly detector. On the one hand, selfish agents exhibit reduced outgoing communication, which can be reflected in the frequency anomaly signal. On the other hand, their incomplete intermediate outputs can likewise be reflected in the semantic anomaly signal.

**Robustness analysis.** Table 7 reports an additional robustness analysis on HotpotQA under self-

| Domain | Dataset | #Train | #Test | Metric |
|---|---|---|---|---|
| **Code Generation** | HumanEval | 164 | – | pass@1 |
| | MBPP | 699 | 175 | pass@1 |
| **Math Reasoning** | GSM8K | 640 | 160 | Accuracy |
| | MATH | 640 | 160 | Accuracy |
| **Knowledge QA** | GPQA | 198 | – | Accuracy |
| **Multi-hop QA** | HotpotQA | 720 | 180 | EM |
| | 2WikiMultiHopQA | 720 | 180 | EM |
| | MuSiQue | 720 | 180 | EM |
| **Embodied** | ALFWorld | 219 | 55 | Success rate |
| **Tool-use** | GAIA (L1/L2/L3) | 165 | – | Accuracy |

Table 4: Dataset statistics for the benchmarks used in this paper.

| GAIA | Level 1 | | Level 2 | | Level 3 | | Avg. | |
|---|---|---|---|---|---|---|---|---|
| NeuralFSM | Perf. | Token. | Perf. | Token. | Perf. | Token. | Perf. | Token. |
| - GPT-4.1-nano | 13.21 | 0.25 | 6.98 | 0.39 | 0.00 | 0.16 | 6.73 | 0.27 |
| - GPT-4o-mini | 7.55 | 0.24 | 5.00 | 0.24 | 0.00 | 0.13 | 4.18 | 0.20 |
| - GPT-5-nano | 18.92 | 0.45 | 11.63 | 0.77 | 7.69 | 0.21 | 12.75 | 0.47 |
| - Grok-3-mini | 28.30 | 0.76 | 12.79 | 1.39 | 3.85 | 0.52 | 14.98 | 0.89 |

Table 5: Performance and token consumption of NeuralFSM with different LLM backbones on the GAIA dataset (Token units: 1M tokens).

ish attack with the attacked-agent ratio fixed to $\rho = 0.5$. Without protection, performance degrades monotonically as the attack strength $\eta$ increases: raising $\eta$ from 1.0 to 3.0 reduces NeuralFSM from 64.73 to 50.23, corresponding to a 22.40% relative drop. In contrast, protected NeuralFSM remains much more stable, decreasing only from 65.12 to 64.25 (1.34% relative drop). The performance gap also widens with stronger attacks, reaching 14.02 at $\eta = 3.0$, indicating that the proposed protection mechanism effectively mitigates selfish behaviors that reduce effective communication among agents. Compared with the no-attack performance reported in Table 2 (66.07 on HotpotQA), the protected at $\eta = 3.0$ shows only a 2.75% relative decrease, whereas the unprotected suffers a much larger 23.97% relative drop.

### E.3 Evaluation on a Challenging Long-Horizon Benchmark

To further examine whether NeuralFSM remains effective beyond the benchmarks used in Section 5, we additionally evaluate it on FSD-Bench (Liu et al., 2025), a challenging software development benchmark involving long-horizon execution, sustained planning, dynamic subtask allocation, and complex multi-agent coordination. FSD-Bench contains three task categories: Website, Desktop, and Game.

Table 8 shows that NeuralFSM achieves the best performance across all three task categories. Specifically, on Website tasks, NeuralFSM outperforms the strongest baseline MasHost by 5.35%; on Desktop tasks, it surpasses MasHost by 7.53%; and on Game tasks, it exceeds RTADev by 7.66%. In addition, NeuralFSM improves the overall average performance by 7.33% over the best competing method. These results indicate that NeuralFSM remains highly effective on more challenging long-horizon problems, suggesting that the fixed FSM state abstraction does not prevent expressive multi-agent coordination, but instead supports strong performance through more structured and interpretable execution.

| Method | No Attack | Under Attack | Protected (Ours) |
|---|---|---|---|
| SC | 80.2 | 53.2 | – |
| DyLAN | 83.9 | 62.8 | – |
| MacNet | 74.3 | 53.8 | – |
| MaAS | 88.2 | 70.6 | – |
| MasHost | 87.9 | 69.5 | – |
| NeuralFSM (Ours) | 91.4 | 81.3 | 89.9 |

Table 6: Robustness of NeuralFSM and other baselines under attack on MBPP: pass@1 in the clean setting, under attack, and with our proposed protection layer enabled.

| $\eta$ | w/o protection | + protection |
|---|---|---|
| 1.0 | 64.73 | 65.12 |
| 1.5 | 62.15 | 65.03 |
| 2.0 | 58.46 | 64.58 |
| 2.5 | 55.67 | 64.47 |
| 3.0 | 50.23 | 64.25 |

Table 7: Robustness of NeuralFSM under selfish attack on HotpotQA with varying attack strength $\eta$.

| Methods | FSD-Bench | | | |
|---|---|---|---|---|
| | Website | Desktop | Game | Avg. |
| GPT-5-nano | 55.23 | 53.66 | 55.17 | 54.69 |
| MacNet | 61.30 | 65.15 | 67.64 | 64.70 |
| RTADev (Liu et al., 2025) | 63.24 | 67.50 | 76.38 | 69.04 |
| AutoAgents | 60.73 | 65.42 | 73.54 | 66.56 |
| MasHost | 69.56 | 71.02 | 75.44 | 72.01 |
| **NeuralFSM (Ours)** | **73.28** | **76.37** | **82.23** | **77.29** |

Table 8: Performance of different methods on FSD-Bench. All five LLM-based methods use `gpt-5-nano` as the backbone.

### E.4 Task Generalization via Cross-Dataset Transfer

To further evaluate task generalization, we perform direct cross-dataset transfer without additional retraining. For each setting, the NeuralFSM controller is trained on a source benchmark (e.g., MATH) and directly evaluated on a different target benchmark (e.g., GSM8K).

Table 9 shows that NeuralFSM consistently achieves the best transferred performance across all three settings. Relative to in-domain training, MATH $\rightarrow$ GSM8K yields a slight $0.59\%$ improvement, whereas GSM8K $\rightarrow$ MATH and MBPP $\rightarrow$ MATH incur only $1.73\%$ and $3.75\%$ drops, respectively. These degradations are still substantially smaller than those of MaAS ($3.80\%$ and $6.12\%$) and MasHost ($8.17\%$ and $14.24\%$). This pattern is consistent with the results in Section 5.2, where NeuralFSM shows greater gains on benchmarks that place stronger demands on multi-step coordination. These results indicate that NeuralFSM learns transferable coordination policies that remain effective under cross-dataset distribution shift.

### E.5 Sensitivity Analysis

#### E.5.1 Protection Layer Sensitivity

To assess whether the robustness gains of the protection layer depend on delicate hyperparameter tuning, we conduct an additional sensitivity analysis on the GPQA benchmark. Specifically, we further vary the centrality-prior weights $w_{\text{BC}} \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$ with $w_{\text{PR}} = 1 - w_{\text{BC}}$, and the anomaly-fusion weights $\omega_{\text{freq}} \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ with $\omega_{\text{sem}} = 1 - \omega_{\text{freq}}$. Following the formulation of the protection layer in Section 4.3, we visualize the results in a three-dimensional surface where the x-axis corresponds to $1 + w_{\text{BC}} - w_{\text{PR}}$, the y-axis corresponds to $1 + \omega_{\text{freq}} - \omega_{\text{sem}}$, and the z-axis reports the protected performance under attack.

As shown in Fig. 7, the protected performance exhibits a center-high and edge-low pattern. The default setting $(w_{\text{BC}}, w_{\text{PR}}, \omega_{\text{freq}}, \omega_{\text{sem}}) = (0.6, 0.4, 0.3, 0.7)$ achieves $61.33$, which is close to the best observed result of $61.46$, while more extreme settings near the boundary lead to lower performance. This suggests that the robustness gains of the protection layer are stable across a reasonable range of hyperparameter choices and do not rely on fine-grained tuning, while a moderately balanced combination of centrality priors and anomaly signals provides the most reliable protection effect.

| Methods | MATH → GSM8K | GSM8K → MATH | MBPP → MATH |
|---|---|---|---|
| MaAS | 98.24 | 78.13 | 76.25 |
| MasHost | 98.68 | 75.63 | 70.63 |
| **NeuralFSM (Ours)** | **99.39** | **91.25** | **89.38** |

Table 9: Cross-dataset transfer performance.

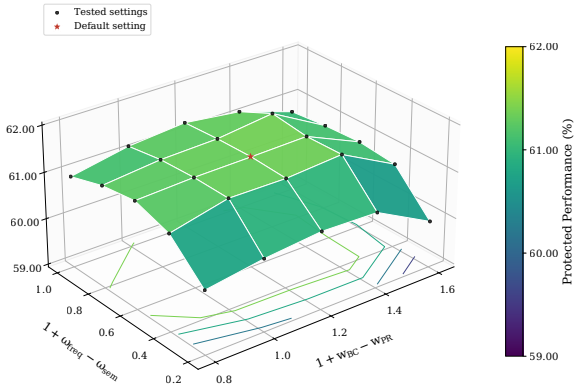


Figure 7: Sensitivity of the protection layer on GPQA. The red star marks the default setting.

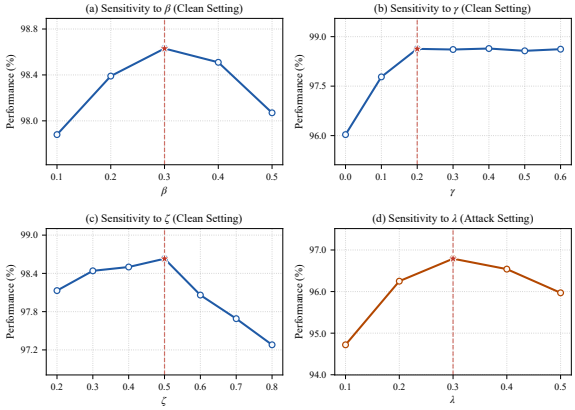


Figure 8: Sensitivity analysis of loss-component trade-off weights. The panels for $\beta$, $\gamma$, and $\zeta$ report performance under the clean setting, while the panel for $\lambda$ reports performance under the attack setting. Red stars indicate the adopted default settings.

### E.5.2 Loss Hyperparameter Sensitivity

We further analyze the sensitivity of the optimization objective to its loss-component trade-off weights. Starting from the default setting $(\alpha, \beta, \gamma, \nu, \zeta, \lambda) = (1.0, 0.3, 0.2, 0.1, 0.5, 0.3)$, we vary one hyperparameter at a time within a reasonable local range while fixing all remaining weights at their default values. Specifically, when varying $\beta$, $\gamma$, $\zeta$, or $\lambda$, the other hyperparameters are kept unchanged. Although $\alpha$ and $\nu$ are tunable in principle, we keep them at their default values in this analysis because these defaults already provide stable optimization and a reasonable performance-cost balance.

Fig. 8 shows that NeuralFSM remains stable under moderate perturbations of these weights. Under the clean setting, the curve for $\beta$ exhibits a clear inverted-U trend: decreasing $\beta$ from the default setting leads to a relative performance drop of $0.76\%$, while increasing it produces a smaller decline of $0.57\%$. This suggests that a moderate transition-stabilization term is sufficient, whereas either under-weighting or over-weighting this term slightly weakens state-transition learning.

For $\gamma$, when it is reduced to $0$, that is, when the communication routing loss is removed, the performance drops by $2.64\%$, indicating that explicit routing supervision is important for learning reliable communication patterns. However, once $\gamma$ enters a moderate range around the default setting, the curve becomes nearly flat and only fluctuates marginally. Therefore, the adopted default value remains a reasonable choice within the explored interval.

For $\zeta$, performance stays strong around the default region, but deviating from this region gradually hurts performance. Reducing $\zeta$ below the default causes at most a $0.51\%$ relative drop, whereas increasing it too much leads to a greater degradation of up to $1.37\%$. This indicates that a moderate horizon penalty helps discourage degenerate long traversals, while an excessively strong penalty may constrain useful exploration and slightly hurt final performance.

For $\lambda$, we report the attacked setting, where the effect of $\mathcal{L}_{\text{protect}}$ can be observed more directly. The curve shows that a moderate protection regularization strength is preferable: setting $\lambda$ too small causes a $2.14\%$ relative drop, while increasing it beyond the default leads to a $0.85\%$ decline. This suggests that insufficient protection weakens robustness, whereas over-emphasizing the protection

term can slightly interfere with the main coordination objective. Overall, across all tested hyperparameters, the red-star default settings remain robust and well-balanced choices within the explored reasonable ranges.

### E.6 Case Study

Fig. 9 presents an MBPP example. The trajectory $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_2 \rightarrow S_5$ reflects a validation–refinement loop. After generating an initial implementation, NeuralFSM identifies during testing that the no-repetition case should return the literal string "None" and routes this feedback to the refinement state for correction. The revised solution is then re-evaluated before final submission. The figure also makes the communication routing among agents explicit, illustrating how intermediate results are selectively routed to agents that may benefit from them. Notably, although the state inventory includes a code-inspection state, this trajectory reaches the final correct implementation without activating it, showing that NeuralFSM need not traverse all available states when a shorter execution path already suffices.

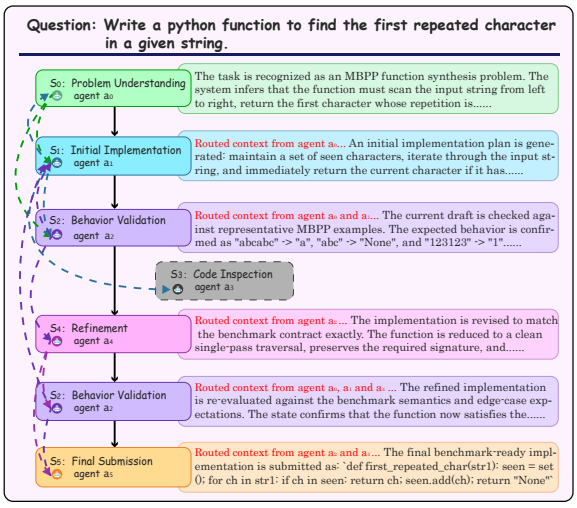


Figure 9: Case study of NeuralFSM on MBPP, showing the executed state trajectory and communication routing among agents for a representative code-generation instance.

## F Prompt Repository

This appendix enumerates the prompts used to generate a master FSM for each domain. In our implementation, the FSM generation process invokes an LLM twice: (i) Step 1 designs the workflow states, and (ii) Step 2 designs agents aligned with these states. The domain-specific FSM is then initialized. Notably, no directed edges are established in the initialized FSM, meaning that no message passing rules are defined. Both prompts are parameterized by a domain template.

**GSM8K template**

```
[task_description]
 Grade-school math word problem solving task
 Task Characteristics:
 - Problems require multi-step reasoning
 - Problems require precise numerical
calculation
 - The final answer is a specific numerical
value
 - Solution correctness depends on verifiable
calculation logic
 FSM Design Principles:
 1. Decompose the workflow into
understanding, planning, calculation,
verification, and submission
 2. Include an explicit verification stage
and allow refinement loops for error
correction
 [state_granularity_guide]
 1. Problem Understanding (1 state): parse
the problem text and extract key quantities,
conditions, and relations
 2. Planning and Calculation (1-2 states):
determine the solution strategy, perform the
required computations, and derive the result
 3. Verification and Correction (1-2 states):
check reasonableness, identify potential
errors, and revise the calculations if
necessary
 4. Final Submission (1 state): format and
output the final numerical answer
 Recommended total states: 4-6
```

**GPQA template**

```
[task_description]
 GPQA (graduate-level multiple-choice QA)
task
 Task Characteristics:
 - High-difficulty, graduate-level science
questions (often multi-step reasoning)
 - Multiple-choice format (A/B/C/D)
 - Requires careful option elimination and
consistency checks
 Global Output Contract (CRITICAL):
 - FINAL answer MUST be a SINGLE LETTER: A /
B / C / D
 - NO explanations, NO extra text, NO
markdown
 - If the prompt contains both a)/b)/c)/d)
and a mapping to A/B/C/D, FOLLOW the final
A/B/C/D mapping
 - If unsure, still choose the best option
letter based on reasoning
 FSM Design Principles:
 1. Organize the workflow around question
parsing, concept recall, option analysis,
cross-checking, and final decision
 2. Allocate enough states to explicit
option-by-option elimination and
contradiction analysis
```

```
 3. Keep the final state strictly
constrained to the option letter output
 [state_granularity_guide]
 1. Question Parsing (1 state): extract what
is being asked, key constraints, and units
 2. Concept Activation (1-2 states): recall
the relevant domain concepts, principles,
and formulas
 3. Option Analysis (1-2 states): evaluate
candidate options and identify
contradictions or distractors
 4. Cross-check and Elimination (1 state):
perform sanity checks, dimensional checks,
and consistency verification
 5. Final Decision (1 state): output ONLY
the final option letter (A/B/C/D)
 Recommended total states: 5-7
```

## GAIA template

```
[task_description]
 GAIA (general AI assistant benchmark) task
 Task Characteristics:
 - Mixed tool-using / multimodal /
document-grounded reasoning
 - Questions may reference external files
(e.g., .xlsx/.pdf/.png/.docx/.csv/.txt) via
a file_path
 - Requires step-by-step planning, extraction
from the given file when present, and final
short answers
 Global Output Contract:
 - Provide ONLY the final answer (short), no
extra commentary unless explicitly
requested.
 - If a numerical answer is required, return
the number in the requested format.
 - If the task requires reading an attached
file, treat the attachment as authoritative
evidence.
 FSM Design Principles:
 1. Separate planning from evidence
acquisition, especially for file-backed
questions
 2. Support both cases with attachments and
cases without external files
 3. Include explicit verification before the
final answer while keeping the structure
compact
 [state_granularity_guide]
 1. Task Parsing (1 state): parse the
question, constraints, required output
format, and detect file_path if present
 2. Planning and Tool Selection (1 state):
decide whether to use file reading, tool use,
or pure reasoning, and identify intermediate
quantities
 3. Evidence Acquisition (1-2 states): read
and interpret attached files or gather key
facts from tools and reasoning
 4. Reasoning / Computation (1-2 states):
perform calculations, logic, and multi-step
synthesis
 5. Verification and Final Answer (1 state):
check consistency with the evidence and emit
the final answer in the requested format
 Recommended total states: 5-7
```

## HumanEval template

```
[task_description]
 Python code generation task (HumanEval)
 Task Characteristics:
 - Implement a single Python function based
on signature and docstring
 - Must pass hidden test suite
 - Primary objective: correct, executable
function
 Global Output Contract (CRITICAL):
 - FINAL answer MUST be single, syntactically
valid Python FUNCTION
 - No explanations, no markdown, no extra
code
 - Function name/signature MUST match given
problem
 - Use proper newlines/indentation
 FSM Design Principles:
 1. Keep most states code-producing and
reserve only a small number of non-coding
diagnostic states
 2. Require every non-final coding state to
output a FULL updated function definition
 3. Organize the workflow around
understanding, drafting,
refinement/debugging, and final submission
 [state_granularity_guide]
 1. Initial Understanding (1 state): read
the signature and docstring, then summarize
requirements and edge cases without emitting
code
 2. First Implementation Draft (1 state):
produce the first COMPLETE function
definition
 3. Guided Refinement Loops (1-2 states):
analyze likely hidden-test failures and
output full updated function revisions
 4. Final Submission (1 state): output the
final polished Python function definition
only
 Recommended total states: 4-5
```

## MBPP template

```
[task_description]
 Basic Python programming task (MBPP)
 Task Characteristics:
 - Small self-contained Python tasks
 - Function signature + assertion-based test
cases
 - Goal: implement function passing all tests
 Global Output Contract (CRITICAL):
 - FINAL answer MUST be single, syntactically
valid Python FUNCTION
 - Function name/parameters MUST match
signature exactly
 - Function body MUST be executable; no
input()/print()
 - Output pure code (no explanations, no
markdown)
 - FORBID multiple statements on one line
(e.g., def f(): a=0; b=1)
 FSM Design Principles:
 1. Keep most states code-writing or
refinement states and avoid long non-coding
stretches
```

```
 2. Model an iterative workflow of
implementation, test-oriented diagnosis,
repair, and final submission
 3. Preserve the exact provided function
signature and keep the final output code-only
 [state_granularity_guide]
 1. Understand_Problem (1 state): identify
required behavior, edge cases, and the exact
signature from the prompt
 2. Initial_Implementation (1 state):
produce the first executable implementation
with the exact required signature
 3. Test-oriented Diagnosis (1-2 states):
inspect likely assertion failures and locate
probable bugs
 4. Refinement (1-2 states): revise the
implementation to address detected errors
and edge cases
 5. Final_Submission (1 state): output the
final function definition only
 Recommended total states: 5-7
```

## MATH template

```
[task_description]
 Competition mathematics task
 Task Characteristics:
 - Advanced mathematical reasoning
 - Requires proof and derivation
 - Symbolic manipulation and theorem
application
 - Multiple solution approaches possible
 FSM Design Principles:
 1. Support a deep reasoning workflow of
analysis, strategy selection, derivation,
verification, and submission
 2. Allocate multiple states to step-by-step
symbolic derivation when necessary
 3. Preserve the option to branch into
alternative solution strategies if
verification fails
 [state_granularity_guide]
 1. Problem Analysis (1 state): identify the
problem type, key concepts, and given
conditions
 2. Strategy Selection (1 state): choose a
solution approach and identify the relevant
theorems or techniques
 3. Solution Derivation (1-2 states): carry
out symbolic manipulation, derivation, and
intermediate calculations
 4. Verification / Alternative Attempt (1
state): check solution validity and revise
the strategy if necessary
 5. Final Submission (1 state): format and
submit the final answer
 Recommended total states: 5-6
```

## ALFWorld template

```
[task_description (condensed excerpt)]
 Embodied AI interactive task (ALFWorld)
 Task Characteristics:
 - Text-based environment simulation in
household scenarios
 - Sequential action planning with state
changes
 - Goal-oriented navigation and object
manipulation
 - Typical structure: locate -> pick up ->
(transform) -> place/examine
 Evaluation Protocol:
 - Subgoals are regex patterns matched
against action feedback
 - All subgoals must be satisfied
sequentially to complete the task
 FSM Design Principles:
 1. Follow the embodied workflow of goal
parsing -> exploration -> acquisition ->
optional transformation -> placement ->
verification
 2. Include explicit parsing of action
feedback and matching against subgoal regex
patterns
 3. Preserve an optional transformation
branch and require the final state to output
the complete action sequence
 [state_granularity_guide]
 1. Goal Parsing and Planning (1 state):
extract target object(s), transformation
type, destination, and the overall task plan
 2. Environment Exploration (1-2 states):
navigate the environment, locate relevant
objects, and open receptacles when needed
 3. Object Acquisition (1 state): execute
the required take actions and confirm
successful pickup from feedback
 4. Optional Transformation (0-1 state):
carry out heat / cool / clean operations
when required
 5. Placement and Verification (1-2 states):
complete the placement action and verify
progress from feedback and subgoal patterns
 6. Final Action Sequence Submission (1
state): compile and output the complete
action sequence
 Recommended total states: 5-8
```

## HotpotQA template

```
[task_description]
 Multi-hop question answering task
 Task Characteristics:
 - Requires reasoning across multiple
documents
 - Need to identify and connect supporting
facts
 - Questions involve bridge or comparison
reasoning
 - Long context with multiple document
paragraphs
 FSM Design Principles:
 1. Structure the workflow around question
analysis, evidence retrieval, hop-by-hop
reasoning, synthesis, and verification
 2. Support both bridge reasoning (A -> B ->
C) and comparison reasoning (A vs B)
 3. Include explicit supporting-fact
verification before the final answer
 [state_granularity_guide]
 1. Question Analysis (1 state): determine
the question type and identify the required
information
 2. Document Retrieval (1-2 states): extract
relevant passages from multiple documents
```

```
and identify key entities
 3. Multi-hop Reasoning (1-2 states):
connect facts hop-by-hop across documents or
entities
 4. Evidence Synthesis and Verification (1-2
states): combine the evidence and validate
the supporting facts
 5. Final Submission (1 state): format and
submit the final answer
 Recommended total states: 5-8
```

| Notation | Definition |
|---|---|
| $q \in \Sigma$ | Input domain problem |
| $y, \hat{y}$ | Ground-truth and prediction |
| $\mathcal{A} = \{a_0, \ldots, a_{m-1}\}$ | Agent set |
| $\mathcal{S} = \{s_0, \ldots, s_{n-1}\}$ | FSM state set |
| $\mathrm{FSM} = (\mathcal{S}, \Sigma, \mathcal{A}, \delta, \mathcal{L}, s_0, \mathcal{F})$ | FSM tuple |
| $s_0, \mathcal{F}$ | Initial state; Final state set |
| $t, T_{\max}$ | Time step index; Rollout horizon |
| $\mathbf{q} = \mathrm{Enc}(q)$ | Problem embedding |
| $\mathbf{c}_t$ | Accumulated execution context |
| $\pi_\theta$ | Temporal coordination controller |
| $\theta$ | Learnable parameters of the temporal controller |
| $\mathcal{D}$ | Training dataset |
| $\eta_{\text{lr}}$ | Learning rate of the optimizer |
| $E_{\max}, B$ | Max training epochs; Batch size |
| $b$ | Reward baseline (exponential moving average) |
| $P_\theta^{\text{trans}}(\tau)$ | Transition distribution |
| $\mathbf{w}_t(\tau)$ | Routing score vector over agents |
| $\{a_r\}_t, k$ | Receiver set; Receiver budget |
| $\mathbf{m}_v(t) \in \mathbb{R}^{d_m}$ | Memory state of node $v$; $d_m$ is memory dimension |
| $\mathbf{h}_v(t)$ | Node representation of node $v$ at time $t$ |
| $\mathbf{e}_{uv}$ | Edge features for interaction $(u, v)$ |
| $\Delta t, \Delta t_{uv}$ | Time difference and last-interaction time gap |
| $\psi(\Delta t)$ | Temporal encoding function for time gaps |
| $C_{\text{episode}}$ | Per-episode FSM execution cost |
| $\mathcal{L}_{\text{total}}$ | Integrated training objective |
| $\alpha, \beta, \gamma, \nu, \zeta, \lambda$ | Loss weights |
| $\pi(i)$ | Protection priority score |
| $\varphi(i, t)$ | Fused anomaly score |
| $\varphi_{\text{freq}}(i, t), \varphi_{\text{sem}}(i, t)$ | Frequency/semantic anomaly indicators |
| $\sigma(\cdot)$ | Sigmoid function |
| $\omega_{\text{freq}}, \omega_{\text{sem}}$ | Anomaly fusion weights |
| $\text{trust}(i, t)$ | Trust score derived from priority and anomaly |
| $w_{\text{BC}}, w_{\text{PR}}$ | Centrality coefficients |
| $d_{\text{PR}}$ | PageRank damping factor |
| $\eta$ | Attack strength |
| $\rho$ | Attacked-agent ratio |
| $\epsilon$ | Small constant for numerical stability |

Table 10: Notation and Definitions