NEGOLAT: NEGOTIATING WHAT TO COMMUNICATE IN MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

A central yet under-explored question in multi-agent systems (MAS) is *what to communicate*. Existing protocols, whether textual or latent, remain sender-centric: an upstream agent composes one message from its own context and delivers the same payload to every downstream agent, which participates only after the message has been formed. Such one-shot communication mismatches receivers that need different information and leaves incomplete or conflicting messages unrepaired. We propose NegoLat, a training-free receiver-negotiated latent communication paradigm. On each edge, the sender first exposes a semantic probe; the downstream receiver audits it against its own role and selectively requests refinement; the sender then regenerates a receiver-specific message only for those who asked. We further analyze when receiver-guided regeneration is beneficial and how a bounded request budget affects information coverage. Across ten benchmarks spanning mathematical reasoning, code generation, and visual question answering, NegoLat consistently outperforms text-based and fixed latent communication, improving over the best baseline by $+1.07$ pp (Qwen3-4B-Instruct) and $+2.47$ pp (Qwen3-VL-4B) on average under Hierarchical MAS, with limited overhead. Communication analyses show that NegoLat adaptively adjusts transmitted information according to downstream requirements. Code will be released.

1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have emerged as an important paradigm for solving complex tasks, with applications spanning mathematical reasoning (Pezeshkpour et al., 2024; Zhou et al., 2025), code generation (Yang et al., 2024; Ding et al., 2025), and question answering (Zhao et al., 2024). In such systems, inter-agent communication determines what information downstream agents receive from upstream agents and directly affects the effectiveness of subsequent collaborative reasoning.

Most existing MAS use natural language as the communication medium, where an upstream agent decodes its internal representations into discrete tokens that are then re-encoded by a downstream agent (Chen et al., 2024b; Hong et al., 2024; Zhang et al., 2025b). Such text-to-text communication can discard latent semantics, limit information fidelity, and accumulate redundant context over repeated interactions (Du et al., 2026; Zhang et al., 2025a; Cemri et al., 2026). To alleviate these limitations, recent studies communicate directly in the continuous latent space, exchanging final-layer hidden states (Du et al., 2026; Peng et al., 2026), KV caches (Fu et al., 2026; Yu et al., 2026b), or latent working memories (Zou et al., 2025). Compared with text, these methods substantially advance *how* inter-agent information is represented and transferred.

However, these methods share a common limitation: the communicated content remains *sender-centric*. An upstream agent composes a message from its own role and context, and the same message is delivered to every downstream agent, which participates only after the message has been formed. As illustrated in Figure 1(a), this is problematic in both fan-out and fan-in communication. Under fan-out, downstream agents require different aspects of information, yet receive the same sender-generated representation. Under fan-in, a receiver aggregates multiple upstream messages even when some are incomplete, redundant, or mutually inconsistent. Neither failure is caused by lossy encoding, and neither is removed by a higher-fidelity channel. Beyond improving the representation, an equally basic question is therefore *what should be communicated*: as in networked

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

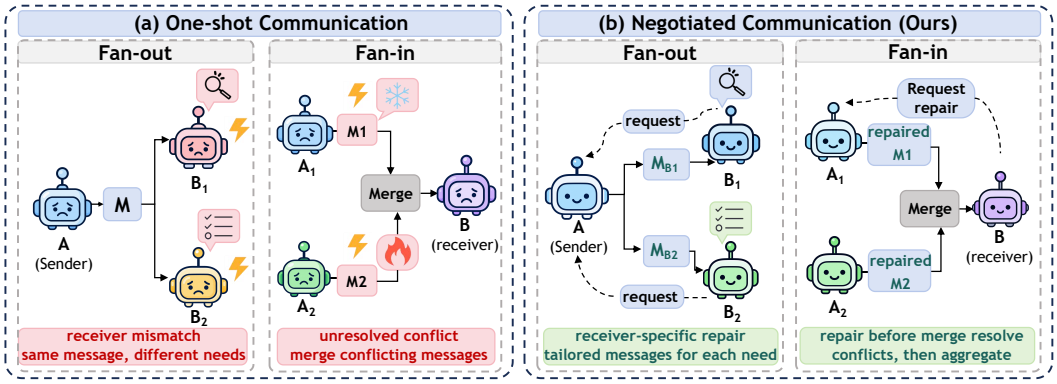

Figure 1: **Communication protocols in multi-agent systems.** (a) One-shot communication transfers fixed messages and suffers from receiver mismatch and unresolved conflicts. (b) NegoLat negotiates with receivers and performs selective repair before communication.

systems, where a receiver acknowledges what suffices and requests what does not, communication content should be shaped by the receiver's information needs before transfer occurs.

To address this problem, we propose NegoLat, a **training-free receiver-negotiated latent communication protocol** that separates a semantic control plane from a latent data plane. On each communication edge, the sender first generates a textual *semantic probe* while caching final-layer hidden states. The downstream receiver audits the probe conditioned on the task and its role, and returns either an ACK or a structured REQUEST. Upon ACK, the cached states are reused directly; upon REQUEST, the sender regenerates a receiver-specific message that resolves the named deficiencies (Figure 1(b)). The corresponding hidden states are then aligned to the receiver's input embedding space and injected as a continuous prefix (Peng et al., 2026). In this way, the semantic control plane determines *what to communicate*, while the latent data plane determines *how the negotiated information is transferred*. The same mechanism naturally extends to different MAS topologies and adapts communication content according to receiver-specific information needs.

We evaluate NegoLat on ten benchmarks across visual question answering, mathematical reasoning, and code generation, under both Sequential and Hierarchical MAS settings. NegoLat consistently outperforms text-based MAS and fixed latent communication, validating the effectiveness of receiver-negotiated latent communication.

Our main contributions are: (i) we revisit inter-agent communication from the perspective of *what should be communicated*, and identify a limitation shared by text and latent methods: message content is committed by the sender before receivers express their instance-specific needs, mismatching heterogeneous receivers under fan-out and leaving conflicts unresolved under fan-in; (ii) we propose NegoLat, a training-free receiver-negotiated communication paradigm that uses semantic probes, receiver-side ACK/REQUEST feedback, and selective receiver-specific regeneration to make communication content a joint decision of sender and receiver; (iii) we theoretically characterize the conditions under which receiver-guided regeneration yields positive expected benefit and analyze how a bounded semantic request budget affects the coverage of receiver-relevant information, providing theoretical support for selective negotiation; (iv) across ten benchmarks and both MAS settings, NegoLat improves over the best baseline by $+1.07$ pp (Qwen3-4B-Instruct) on reasoning and code and $+2.47$ pp (Qwen3-VL-4B) on VQA under Hierarchical MAS, with limited overhead, while ablations and communication analyses show that requests adapt to receiver and task needs.

2 Related Work

2.1 Multi-Agent LLM Systems

LLM-based multi-agent systems improve complex problem solving by coordinating specialized agents through role assignment, structured workflows, and explicit interaction topologies (Wu et al., 2023; Li et al., 2023; Hong et al., 2024; Qian et al., 2024). Prior work has explored role specialization and workflow design (Chen et al., 2024b), topology and architecture optimization (Zhang et al.,

2024; Nie et al., 2026; Li et al., 2026), as well as feedback and iterative coordination (Du et al., 2023). More recently, inter-agent communication itself has become an important design dimension, with studies exploring learned communication interfaces, adaptive protocols, and communication scheduling (Yang et al., 2026). Despite these advances, most LLM-based MAS still rely on natural-language messages as the primary medium for exchanging intermediate reasoning states (Chen et al., 2024b; Hong et al., 2024; Zhang et al., 2025b), motivating richer communication mechanisms beyond discrete text.

2.2 Latent Communication

Recent work increasingly explores continuous latent representations for inter-agent communication. Existing approaches can be broadly grouped by the representation used for information exchange. *Hidden-state and latent-token methods* directly transmit internal representations or compact continuous tokens derived from them, preserving information that would otherwise be lost during textual decoding (Du et al., 2026; Peng et al., 2026; Chen et al., 2026; Yu et al., 2026a). *KV-cache and latent-memory methods* instead communicate through internal attention states or persistent latent memories, including projected or shared KV caches, latent working memory, and decoupled perception–reasoning memories (Fu et al., 2026; Zou et al., 2025; Yu et al., 2026b). A third line of work develops *structured or hybrid latent communication*, for example by identifying and selectively sharing latent thoughts or by routing different information through complementary latent and textual channels (Zheng et al., 2026; Mou et al., 2026). Together, these methods substantially advance how information is represented, selected, and transferred beyond discrete text.

Despite these advances, existing methods largely follow sender-centric one-shot communication, where content is formed before downstream receivers express their information needs. NegoLat instead moves feedback upstream of transmission and applies it to the communication act itself. To our knowledge, NegoLat is the first MAS communication paradigm where message content is a joint decision of sender and receiver rather than a sender commitment broadcast to downstream agents.

3 Method

Problem Formulation. We consider a role-specialized multi-agent system organized as a directed acyclic graph (DAG) $G = (V, E)$, where each node $i \in V$ denotes an agent $A_i = (f_\theta, r_i)$ and each edge $(i, j) \in E$ represents information flow from an upstream agent A_i (the sender) to a downstream agent A_j (the receiver). All agents share the same frozen backbone f_θ and are specialized only by their role specifications r_i. We write $\mathrm{Pa}(j) = \{i \mid (i, j) \in E\}$ and $\mathrm{Ch}(i) = \{j \mid (i, j) \in E\}$ for the direct senders of A_j and the direct receivers of A_i, respectively; $|\mathrm{Ch}(i)| > 1$ corresponds to a *fan-out* pattern and $|\mathrm{Pa}(j)| > 1$ to a *fan-in* pattern. Given a task instance x and the upstream context $\mathcal{C}_i$ available to A_i, conventional one-pass communication produces a single sender-centric message:

$$(Y_i, H_i) = f_\theta(x, r_i, \mathcal{C}_i), \qquad \mathcal{M}_{i\to j} = (Y_i, H_i), \quad \forall\, j \in \mathrm{Ch}(i), \tag{1}$$

where Y_i is the decoded token sequence and H_i its final-layer hidden states. The second equality exposes the limitation: the transmitted content is independent of the receiver index j, and is thus agnostic to the receiver's role r_j and its instance-specific needs. Let $\mathcal{U}_j$ denote the semantic items that A_j requires from its senders; a message is *deficient* for A_j when it omits high-importance items of $\mathcal{U}_j$. Because the message is formed before the receiver can express these needs, $\mathcal{U}_j$ never enters message formation, and deficiency can be neither detected nor repaired before transfer. NegoLat instead explicitly asks *what this sender should communicate to this receiver* prior to transfer.

3.1 Receiver-Negotiated Communication

As illustrated in Figure 2, NegoLat separates *what to communicate* from *how to communicate* through a semantic control plane and a latent data plane. For each edge $(i, j) \in E$, the upstream agent A_i first exposes a candidate message for semantic inspection. The downstream agent A_j then evaluates the candidate according to its own role, either accepting it or requesting receiver-specific refinement before latent transfer.

Semantic Vocabulary and Request Schema. We define a universal operation set $\mathcal{O} = \{\mathrm{ADD}, \mathrm{REFINE}, \mathrm{VERIFY}, \mathrm{DISAMBIGUATE}, \mathrm{RESOLVE}\}$ and a task-family semantic vocab-

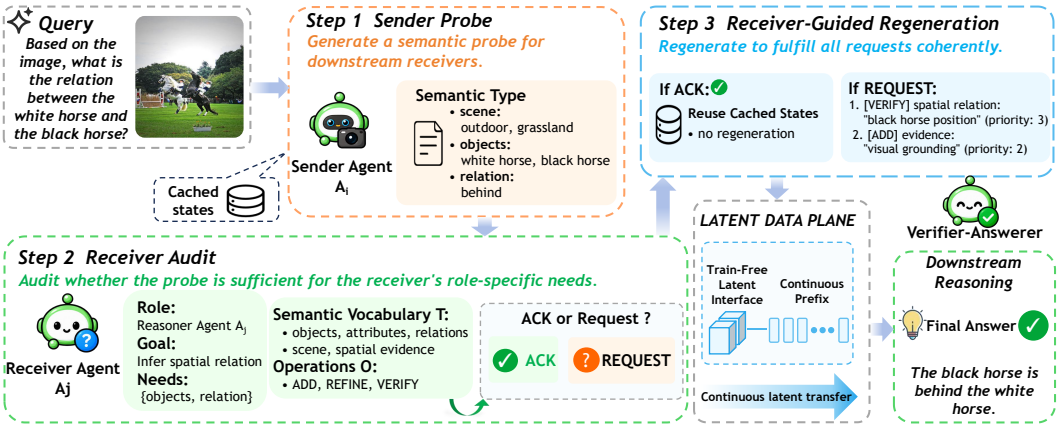

Figure 2: **Overview of NegoLat.** NegoLat separates a semantic control plane from a latent data plane. In the control plane, the upstream agent generates a semantic probe, and the downstream agent provides an ACK or structured REQUEST to determine communication content. In the data plane, the resulting message (either cached or regenerated) is transferred through a training-free latent interface for downstream reasoning.

ulary $\mathcal{T}_\tau$, where τ indexes the task family. $\mathcal{T}_\tau$ specifies task-relevant semantic types, such as visual evidence and spatial relations for VQA, constraints for mathematical reasoning, and algorithms for code generation; complete vocabularies are provided in Appendix B.1. A structured request issued by receiver A_j to sender A_i is a set of semantic clauses:

$$\mathcal{Q}_{j\to i} = \{q_m\}_{m=1}^{M_{j\to i}}, \qquad q_m = (o_m, t_m, z_m, \pi_m), \qquad M_{j\to i} \le M_j, \tag{2}$$

where $M_{j\to i}$ is the number of clauses requested from A_i and M_j is the maximum request budget of receiver A_j. Each clause q_m specifies the requested operation $o_m \in \mathcal{O}$, semantic type $t_m \in \mathcal{T}_\tau$, instance-specific target z_m, and priority π_m. This schema allows the receiver to explicitly name the semantic deficiency to be addressed rather than issuing an unrestricted free-form regeneration prompt, keeping feedback auditable and bounded in size.

Probe Exposure and Receiver Audit. Let $\mathcal{G}_\theta^P$ and $\mathcal{G}_\theta^R$ denote the probe-generation and receiver-conditioned regeneration modes of the frozen backbone f_θ. Before transfer, agent A_i generates a semantic probe $(Y_i^P, H_i^P) = \mathcal{G}_\theta^P(x, r_i, \mathcal{C}_i)$, where Y_i^P is the decoded text used for receiver-side inspection and H_i^P is cached for constructing the continuous payload. The probe is generated once per sender as its role-conditioned message, replacing one-pass generation. Let $\mathcal{P}_j = \{Y_i^P\}_{i\in \mathrm{Pa}(j)}$ collect probes from all senders of A_j; the receiver audits them jointly, conditioned on its role, the semantic vocabulary, and request budget:

$$\{F_{j\to i}\}_{i\in \mathrm{Pa}(j)} = \Psi_\theta\left(x, r_j, \mathcal{P}_j; \mathcal{O}, \mathcal{T}_\tau, M_j\right), \qquad F_{j\to i} \in \{\mathrm{ACK},\ \mathrm{REQUEST}(\mathcal{Q}_{j\to i})\}, \tag{3}$$

where Ψ_θ denotes the receiver-side semantic audit performed by the same frozen backbone, and $F_{j\to i}$ is the feedback from A_j to A_i. An ACK indicates that the corresponding candidate is sufficient for the receiver, whereas a REQUEST carries a structured request identifying the semantic information to be revised or supplemented.

Receiver-Guided Message Formation. For an upstream source receiving an ACK, the cached probe state is reused directly, requiring no second message generation. If A_j instead issues a REQUEST to A_i, the upstream agent regenerates a complete receiver-specific message:

$$(Y_{i\to j}^R, H_{i\to j}^R) = \mathcal{G}_\theta^R\left(x, r_i, \mathcal{C}_i\,;\, r_j, Y_i^P, \mathcal{Q}_{j\to i}\right), \tag{4}$$

where $Y_{i\to j}^R$ is the regenerated token sequence for receiver A_j and $H_{i\to j}^R$ its final-layer hidden states. The message state selected locally at the sender after negotiation is therefore

$$(Y_{i\to j}^\star, H_{i\to j}^\star) = \begin{cases} (Y_i^P, H_i^P), & F_{j\to i} = \mathrm{ACK}, \\ (Y_{i\to j}^R, H_{i\to j}^R), & F_{j\to i} = \mathrm{REQUEST}(\mathcal{Q}_{j\to i}). \end{cases} \tag{5}$$

Here, $(Y_{i\to j}^\star, H_{i\to j}^\star)$ denotes the message state selected locally at the sender after negotiation. $Y_{i\to j}^\star$ is retained only for constructing the reference embeddings required by the alignment interface; it is

not transmitted through the data plane. ACK directly reuses the cached candidate states, whereas REQUEST regenerates a coherent complete message.

Negotiation remains local to direct communication dependencies. Sequential MAS is the special case $|\mathrm{Pa}(j)| = 1$, while in Hierarchical MAS the receiver may jointly audit multiple direct upstream agents and selectively request regeneration only from sources whose information is insufficient. Feedback is not recursively propagated to earlier ancestors.

Continuous Representation Transfer. After negotiation, the selected sender-side state is converted into the continuous payload used by the data plane. Following StateBridge (Peng et al., 2026), the decoded tokens $Y^{\star}_{i\to j}$ are used locally to obtain reference embeddings $R^{\star}_{i\to j} = W_{\mathrm{emb}}[Y^{\star}_{i\to j}]$, where W_{emb} is shared because all agents instantiate the same frozen backbone. The selected hidden states are then aligned to the receiver's input embedding space and concatenated as continuous prefixes:

$$Z_{i\to j} = \Phi_{\mathrm{SB}}\left(H^{\star}_{i\to j}, R^{\star}_{i\to j}\right), \qquad X_j = \left[\mathrm{Concat}_{i\in\mathrm{Pa}(j)}\, Z_{i\to j}; E_j\right], \tag{6}$$

where Φ_{SB} is the training-free StateBridge alignment interface, $Z_{i\to j}$ is the continuous prefix, E_j denotes the prompt embeddings of A_j, and incoming prefixes are concatenated in a fixed upstream-agent order. The downstream agent thus receives only the negotiated continuous prefixes: $Y^{\star}_{i\to j}$ never crosses the edge and is retained only to construct $R^{\star}_{i\to j}$. Since alignment is applied only after negotiation, any latent transfer mechanism can be substituted without changing the protocol.

3.2 Theoretical Characterization

We next characterize two properties of receiver-side negotiation: when regeneration is beneficial, and how the request budget governs coverage. Additional details are provided in Appendix C.

Effectiveness of Receiver-Guided Regeneration. Consider a communication edge $(i, j) \in E$. Let $D_{i\to j} \in \{0, 1\}$ indicate whether the initial semantic probe Y^{P}_{i} is deficient with respect to the information needs of receiver A_j, and let $I_{j\to i} \in \{0, 1\}$ indicate whether A_j issues a REQUEST to A_i. Specifically, $I_{j\to i} = 1$ iff $F_{j\to i} = \mathrm{REQUEST}(\mathcal{Q}_{j\to i})$. We define $p = \Pr(D_{i\to j} = 1)$ as the *deficiency prevalence*, i.e., the probability that an initial probe is genuinely insufficient for the receiver; $t = \Pr(I_{j\to i} = 1 \mid D_{i\to j} = 1)$ as the *true-request rate*, i.e., the probability that the receiver requests regeneration when a deficiency is indeed present; and $f = \Pr(I_{j\to i} = 1 \mid D_{i\to j} = 0)$ as the *false-request rate*, i.e., the probability of unnecessarily requesting regeneration for an already sufficient probe. Let $\delta > 0$ denote the expected downstream improvement from regenerating a genuinely deficient probe, and let $\varepsilon \geq 0$ denote the expected degradation from unnecessarily regenerating an already sufficient one.

Theorem 1 characterizes when receiver-guided regeneration is effective. Regeneration is beneficial when genuinely deficient probes are identified frequently enough that the cumulative improvement from correcting them outweighs the degradation introduced by spurious regeneration. In other words, effective negotiation requires both accurate deficiency detection and selective triggering, rather than indiscriminately issuing more REQUESTs.

Theorem 1 (When Receiver-Guided Regeneration Is Beneficial).
On edge (i, j), the expected downstream performance change induced by receiver-guided regeneration is

$$G_{i\to j} = pt\delta - (1-p)f\varepsilon. \tag{7}$$

Therefore, receiver-guided regeneration improves expected downstream performance iff

$$pt\delta > (1-p)f\varepsilon.$$

Theorem 2 (Diminishing Returns of Semantic Requests).
Let $\mathcal{U}_j$ denote receiver-relevant semantic items, with weight $w_u \geq 0$ and recovery probability $s_u \in [0, 1]$. Under budget M_j,

$$C_j(M_j) = \frac{\sum_{u\in\mathcal{U}_j} w_u \left[1 - (1-s_u)^{M_j}\right]}{\sum_{u\in\mathcal{U}_j} w_u}. \tag{8}$$

Then $C_j(M_j)$ increases with M_j but exhibits diminishing marginal gains.

Semantic Coverage under a Bounded Request Budget. For receiver A_j, $\mathcal{U}_j$ contains the semantic information relevant to its downstream reasoning, while w_u reflects the relative importance of each item. We model each additional request clause as providing an independent opportunity to recover an uncovered item u with probability s_u, without removing previously recovered information.

Theorem 2 characterizes the resulting diminishing-return behavior. Increasing the request budget enables the receiver to recover additional semantic information, yet the marginal coverage con-

Table 1: **Mathematical reasoning and code generation under Sequential and Hierarchical MAS. Improve**: NegoLat's gain (pp) over the best baseline.

Tasks	Qwen3-4B-Instruct				Improve	Llama-3.2-3B-Instruct				Improve
	Single	TextMAS	StateBridge	NegoLat		Single	TextMAS	StateBridge	NegoLat	
Sequential MAS Setting										
AQuA	80.71	79.92	79.13	**83.07**	↑2.36	41.73	44.49	38.98	**45.28**	↑0.79
AIME24	56.67	60.00	**66.67**	63.33	↓3.34	2.50	6.67	5.00	**8.33**	↑1.66
AIME25	43.33	43.33	**50.00**	**50.00**	0.00	0.48	0.95	0.95	**5.83**	↑4.88
HMMT26	30.30	33.33	30.30	**36.36**	↑3.03	0.76	0.76	0.00	**4.55**	↑3.79
HumanEval	**89.02**	**89.02**	87.20	**89.02**	0.00	56.10	59.15	57.93	**60.37**	↑1.22
HumanEval+	82.32	83.54	81.71	**84.15**	↑0.61	50.61	54.27	54.27	**54.88**	↑0.61
MBPP	82.80	83.07	80.69	**85.71**	↑2.64	65.61	65.87	63.76	**66.14**	↑0.27
MBPP+	67.46	68.25	68.25	**71.69**	↑3.44	52.38	53.44	52.12	**53.70**	↑0.26
Hierarchical MAS Setting										
AQuA	80.71	75.20	78.35	**81.10**	↑0.39	41.73	**42.13**	40.94	**42.13**	0.00
AIME24	**56.67**	50.00	**56.67**	**56.67**	0.00	2.50	5.00	4.17	**7.50**	↑2.50
AIME25	43.33	36.67	40.00	**46.67**	↑3.34	0.48	0.95	0.48	**5.00**	↑4.05
HMMT26	30.30	24.24	30.30	**33.33**	↑3.03	0.76	0.76	0.00	**6.82**	↑6.06
HumanEval	89.02	88.41	80.49	**89.63**	↑0.61	56.10	57.93	58.54	**59.15**	↑0.61
HumanEval+	82.32	76.22	73.78	**82.93**	↑0.61	50.61	52.44	51.83	**53.05**	↑0.61
MBPP	82.80	80.69	77.25	**83.10**	↑0.30	**65.61**	62.70	63.49	64.81	↓0.80
MBPP+	67.46	60.05	61.38	**67.72**	↑0.26	**52.38**	50.00	50.79	51.59	↓0.79

tributed by each successive request clause decreases monotonically. Consequently, a small number of well-targeted semantic requests can already capture a substantial fraction of the useful information, whereas further requests yield progressively smaller gains in coverage.

4 Experiments

4.1 Experimental Settings

Benchmarks. We evaluate NegoLat on ten benchmarks spanning mathematical reasoning (AQuA (Ling et al., 2017), AIME24 (Maxwell-Jia, 2024), AIME25 (math ai, 2025), and HMMT26 (Dekoninck et al., 2026)), code generation (HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021)), and visual question answering (MMBench (Liu et al., 2024), MM-Star (Chen et al., 2024a), RealWorldQA (xAI, 2024), and HallusionBench (Guan et al., 2024)). We report accuracy for mathematical reasoning and VQA, and pass@1 for code generation.

Baselines. We compare NegoLat with three representative settings: **Single**, where a single agent directly solves the task; **TextMAS**, which uses the same agent roles and topology as NegoLat but communicates through natural language; and **StateBridge** (Peng et al., 2026), a training-free latent communication baseline that transfers sender-generated continuous representations without receiver negotiation. StateBridge is our most direct comparison, since NegoLat adopts the same data plane and differs only by adding the semantic control plane.

Implementation Details. We experiment with four backbones: Qwen3-4B-Instruct (Yang et al., 2025) and Llama-3.2-3B-Instruct (Grattafiori et al., 2024) for language tasks, and Qwen3-VL-4B (Bai et al., 2025) and InternVL3.5-4B (Wang et al., 2025) for multimodal tasks, under both **Sequential** and **Hierarchical** settings. NegoLat is entirely training-free and introduces no additional verifier or orchestration model. Role specifications, prompt templates, and semantic vocabularies are provided in Appendices B.2 and B.1.

4.2 Main Results

Reasoning and Code Generation. Table 1 reports the results on mathematical reasoning and code generation. On Qwen3-4B-Instruct, NegoLat gains $+1.09$ pp on average over the best baseline under the Sequential setting, including $+2.36$ on AQuA and $+3.44$ on MBPP+. Under the Hierarchical setting, NegoLat gains $+1.07$ pp on average, with $+3.34$ on AIME25 and $+3.03$ on HMMT26. Compared with StateBridge, our most direct latent-communication baseline, the gains are larger: $+2.42$ pp (Sequential) and $+5.37$ pp (Hierarchical) on average. The widening gap under Hierar-

Table 2: **Visual question answering under Sequential and Hierarchical MAS. Improve**: NegoLat's gain (pp) over the best baseline.

Tasks	Qwen3-VL-4B				Improve	InternVL3.5-4B				Improve
	Single	TextMAS	StateBridge	NegoLat		Single	TextMAS	StateBridge	NegoLat	
Sequential MAS Setting										
MMBench-en	83.59	83.59	83.33	**84.78**	↑1.19	80.15	79.55	78.69	**82.54**	↑2.39
MMStar	63.25	69.60	69.20	**72.49**	↑2.89	63.76	**67.25**	64.73	66.20	↓1.05
RealWorldQA	70.85	71.11	71.76	**72.03**	↑0.27	66.01	61.05	66.80	**68.37**	↑1.57
HallusionBench	57.60	69.30	**76.34**	75.29	↓1.05	70.98	75.81	74.03	**76.24**	↑0.43
Hierarchical MAS Setting										
MMBench-en	83.59	84.71	83.76	**89.00**	↑4.29	80.15	79.90	79.30	**83.30**	↑3.15
MMStar	63.25	71.40	68.40	**76.08**	↑4.68	63.76	67.63	65.52	**71.53**	↑3.90
RealWorldQA	70.85	72.42	72.03	**72.68**	↑0.26	66.01	65.36	66.93	**70.72**	↑3.79
HallusionBench	57.60	74.45	74.97	**75.60**	↑0.63	70.98	76.55	75.18	**77.50**	↑0.95

Table 3: **Protocol ablations under Sequential and Hierarchical MAS.** Each variant removes one ingredient of the semantic control plane; *Fixed Latent* is transfer without negotiation.

Variant	Sequential MAS Setting				Hierarchical MAS Setting			
	AQuA	HumanEval+	MBPP+	RealWorldQA	AQuA	HumanEval+	MBPP+	RealWorldQA
Fixed Latent	79.13	81.71	68.25	71.76	78.35	73.78	61.38	72.03
Unconditioned Two-Pass	81.10	76.20	64.55	66.14	79.92	78.66	60.85	67.58
w/o Receiver Role	81.89	81.71	70.11	71.76	79.92	76.22	64.02	71.90
Free-form Request	80.31	82.32	69.05	71.63	79.13	79.27	62.96	71.63
NegoLat (Ours)	**83.07**	**84.15**	**71.69**	**72.03**	**81.10**	**82.93**	**67.72**	**72.68**

chical settings is expected: fan-out and fan-in edges make receiver heterogeneity unavoidable, so deciding what to communicate matters more. Notably, StateBridge often falls *below* Single under Hierarchical, showing that a higher-fidelity channel alone does not help.

Visual Question Answering. Table 2 shows the same pattern on VQA. On Qwen3-VL-4B, NegoLat gains $+0.83$ pp on average over the best baseline under the Sequential setting, with $+2.89$ on MMStar and $+1.19$ on MMBench-en. Under the Hierarchical setting, NegoLat gains $+2.47$ pp on average, with $+4.29$ on MMBench-en and $+4.68$ on MMStar. The consistency across modalities and settings supports our central claim: negotiating *what* to communicate is more effective than transmitting a fixed sender-generated message.

4.3 Ablation Studies

Protocol Ablations. We ablate three design choices on Qwen3-4B-Instruct and Qwen3-VL-4B: receiver-role conditioning, structured vs. free-form feedback, and two-pass generation without negotiation (Table 3). Removing the receiver role consistently degrades performance, confirming that effective feedback should be conditioned on the downstream agent's information needs. Replacing structured requests with free-form feedback also leads to consistent performance drops, demonstrating the benefit of an explicit semantic control interface. Moreover, the two-pass variant remains clearly below NegoLat, particularly on HumanEval+ and MBPP+, indicating that the improvement stems from receiver-guided negotiation rather than simply an additional generation pass.

Hierarchical-Specific Ablations. Table 4 further investigates the two components introduced for Hierarchical communication. Sharing a single payload across downstream receivers degrades performance, supporting the necessity of receiver-specific message formation under fan-out. Similarly, replacing joint multi-source auditing with independent audits consistently hurts performance, demonstrating the importance of jointly inspecting multiple upstream messages under fan-in. These results collectively validate the receiver-specific and multi-source designs employed by NegoLat in complex Hierarchical collaboration.

Table 4: Ablations under Hierarchical MAS.

Variant	AQuA	MBPP+	RealWorldQA
w/ Shared Payload	80.31	66.14	72.29
w/ Independent Audit	79.92	65.61	71.90
NegoLat (Ours)	**81.10**	**67.72**	**72.68**

Table 5: **Efficiency analysis under Sequential and Hierarchical MAS settings.** We report task accuracy, average generated tokens, and end-to-end latency per sample.

Tasks	Metrics	Sequential MAS Setting			Hierarchical MAS Setting		
		TextMAS	StateBridge	NegoLat	TextMAS	StateBridge	NegoLat
MBPP+	Acc. ↑	68.25	68.25	**71.69**	60.05	61.38	**67.72**
	Gen. Tokens ↓	1510.2	1120.9	1853.4	5121.3	2853.1	3371.7
	Latency (s) ↓	54.62	38.52	149.97	152.44	97.65	159.90
MMStar	Acc. ↑	69.60	69.20	**72.49**	71.40	68.40	**76.08**
	Gen. Tokens ↓	1654.6	1266.4	1325.3	1670.1	1351.6	1694.5
	Latency (s) ↓	57.55	37.54	82.57	67.59	39.07	73.96
MMBench	Acc. ↑	83.59	83.33	**84.78**	84.71	83.76	**89.00**
	Gen. Tokens ↓	1096.3	879.0	1027.6	1137.8	1009.6	1227.0
	Latency (s) ↓	30.40	24.07	51.91	33.23	28.62	75.94

Table 6: **Token-level cost breakdown of NegoLat.** Percentages indicate the ratios of feedback and regeneration tokens to total generated tokens.

Tasks	Topo	Probe Tok. ↓	Feedback Tok. ↓	Avg. Regen Tok. ↓	Total Gen. Tok. ↓	Request Rate
MMBench	Sequential	938.6	70.8 (6.9%)	18.2 (1.8%)	**1027.6**	12.20%
	Hierarchical	1052.8	133.4 (10.9%)	40.8 (3.3%)	**1227.0**	25.10%
MMStar	Sequential	1158.8	80.6 (6.1%)	85.9 (6.5%)	**1325.3**	8.90%
	Hierarchical	1451.7	150.3 (8.9%)	92.5 (5.5%)	**1694.5**	25.00%
MBPP+	Sequential	1348.5	243.6 (13.1%)	261.3 (14.1%)	**1853.4**	16.40%
	Hierarchical	2742.3	282.9 (8.4%)	346.5 (10.3%)	**3371.7**	13.80%

4.4 Efficiency and Negotiation Overhead

End-to-End Inference Cost. We measure MAS efficiency with Qwen3-4B-Instruct on code and Qwen3-VL-4B on VQA; Single is excluded as it involves no inter-agent communication. Table 5 shows that NegoLat achieves accuracy gains while generating tokens comparable to TextMAS, and fewer in several settings (e.g., Sequential MMStar 1325.3 vs. 1654.6, Hierarchical MBPP+ 3371.7 vs. 5121.3). This indicates NegoLat spends computation only when negotiation is needed.

Negotiation Overhead. Table 6 quantifies the overhead introduced by negotiation. Feedback accounts for 6.1%–13.1% of total tokens, regeneration contributes 1.8%–14.1%, and the request rate stays between 8.9%–25.1%, indicating regeneration is triggered on only a minority of edges. Thus, accuracy gains come with limited overhead: ACK edges reuse cached states, and regeneration occurs only on REQUEST.

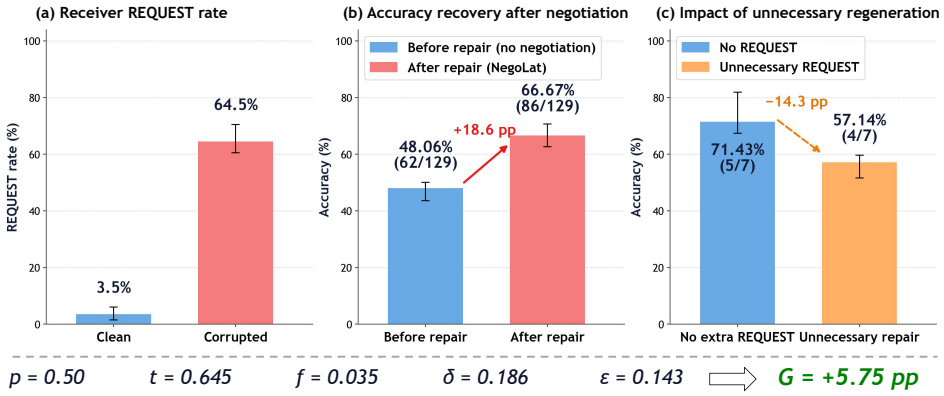

Figure 3: Receiver REQUEST behavior on clean and corrupted messages, accuracy recovery after negotiation, and the overall benefit of receiver-guided regeneration.

4.5 In-depth Analyses on NegoLat

When Is Receiver Negotiation Beneficial? Theorem 1 predicts negotiation helps only when requests are precise. On AQuA we construct 200 paired clean and corrupted Planner→Solver probes and allow REQUEST only on that edge (details in Appendix C.2). As shown in Fig. 3, the receiver requests corrupted probes far more often ($t = 0.645$ vs. $f = 0.035$). Regeneration lifts accuracy

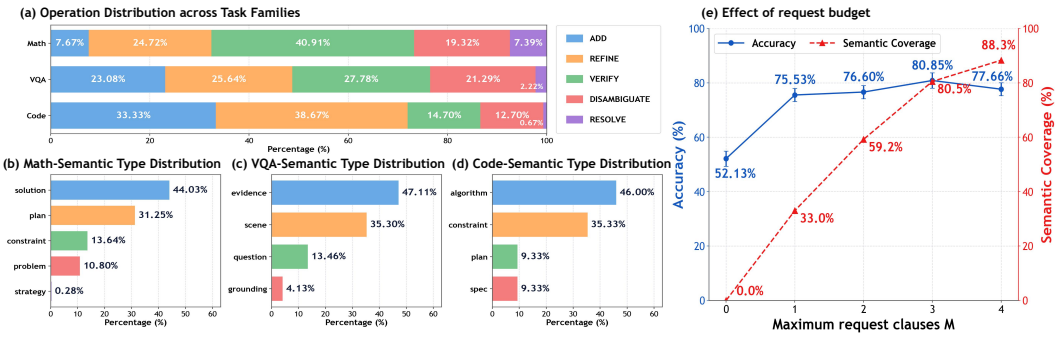

Figure 4: **Negotiation behavior and request budget effect.** (a)–(d) Operation and semantic-type distributions of receiver requests across mathematics, VQA, and code generation. (e) Effect of request budget M on downstream accuracy and semantic coverage.

from 48.06% to 66.67% ($\delta = 0.186$), while only $7/200$ clean probes are mistakenly requested with $\varepsilon = 0.143$. Substituting into Eq. 7 yields $G = +5.75$ pp: receivers detect missing information, and repairing deficient messages gains more than occasional unnecessary rewrites lose.

Request Budget and Semantic Coverage. Using the same AQuA setting, we vary $M \in \{0, 1, 2, 3, 4\}$, the maximum semantic clauses per REQUEST. Each corrupted probe contains multiple semantic defects, and the receiver request is truncated to its top-M clauses. As shown in Fig. 4(e), semantic coverage grows with M but exhibits diminishing gains, consistent with Theorem 2. Accuracy peaks at 80.85% when $M = 3$; further increasing M reduces accuracy despite higher coverage. This shows that more requests are not necessarily better: a small set of high-value semantic requests captures most critical needs, while extra requests may introduce redundancy. We adopt $M = 3$ as the default budget.

Negotiation Behavior Analysis. Figure 4(a)–(d) analyzes negotiation behavior. The requested operations are task-dependent: Math is dominated by VERIFY (40.91%), Code by REFINE and ADD (38.67% and 33.33%), while VQA is more balanced. Semantic types separate clearly, with Math requesting *solution* and *plan*, VQA *evidence* and *scene*, and Code *algorithm* and *constraint*. This shows that NegoLat adaptively adjusts *what* to request according to task-specific needs.

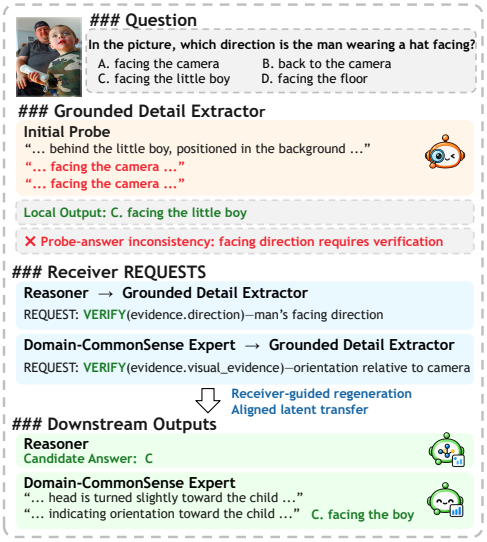

Figure 5: Case of NegoLat.

Case Study. Figure 5 shows a VQA instance where the initial probe asserts "facing the camera," contradicting its local answer. The receivers detect this inconsistency and issue VERIFY requests; after regeneration the agents correctly conclude that the man faces the child, demonstrating how NegoLat repairs semantic errors before transfer.

5 Conclusion

We presented NegoLat, a training-free framework for receiver-negotiated latent communication in multi-agent systems. NegoLat enables receivers to inspect semantic probes, provide structured feedback, and selectively trigger receiver-specific message refinement prior to latent transfer. Extensive experiments across mathematical reasoning, code generation, and visual question answering demonstrate consistent improvements over both fixed latent communication and text-based MAS baselines, with limited additional overhead. Our findings underscore that effective latent collaboration requires not only richer representations, but also receiver-aware decisions regarding *what* information to communicate.

AI USE STATEMENT

This section discloses how language models were used in preparing this submission, as required. General-purpose large language models were used solely as writing assistance: rewriting sentences for readability, tightening phrasing, polishing transitions, and correcting typos. No language model was used to generate scientific claims, design experiments, produce technical content, or decide on methodology. All results, figures, analyses, and conclusions were produced and verified by the authors, who take full responsibility for the content.

REPRODUCIBILITY STATEMENT

Appendix A.1 gives the notation summary and Table 7 the symbol definitions; the full NegoLat algorithm is Algorithm 1. Appendix B.1 specifies the semantic vocabularies for all three task families (Math, VQA, Code), and Appendix B.2 provides the agent role definitions and prompt templates. Appendix C.1 and C.3 contain the full proofs of Theorems 1 and 2; Tables 8 and 9 report the controlled-study setups. Appendix D.1 states the experimental protocol with Table 10, and Appendix D.2 provides implementation details. Appendix E provides detailed case studies with full agent outputs. Code and configuration files will be released upon acceptance.

REFERENCES

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *Advances in Neural Information Processing Systems*, 38, 2026.

Kerui Chen, Jinglu Wang, Jianrong Zhang, Ming Li, Yan Lu, and Hehe Fan. Scaling video understanding via compact latent multi-agent collaboration. *arXiv preprint arXiv:2605.00444*, 2026.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *Advances in Neural Information Processing Systems*, 37:27056–27087, 2024a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024b.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

Hanxing Ding, Shuchang Tao, Liang Pang, Zihao Wei, Jinyang Gao, Bolin Ding, Huawei Shen, and Xueqi Cheng. Toolcoder: A systematic code-empowered tool learning framework for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17876–17891, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27106–27129, 2026.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations*, volume 2026, pp. 43130–43158, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14375–14385. IEEE, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

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

Shiyuan Li, Yixin Liu, Yu Zheng, Mei Li, Quoc Viet Hung Nguyen, and Shirui Pan. Ofa-mas: One-for-all multi-agent system topology design based on mixture-of-experts graph generative models. In *Proceedings of the ACM Web Conference 2026*, pp. 1333–1344, 2026.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 158–167, 2017.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024.

math ai. AIME 2025 Dataset. `https://huggingface.co/datasets/math-ai/aime25`, 2025.

Maxwell-Jia. AIME 2024 Dataset. `https://huggingface.co/datasets/Maxwell-Jia/AIME_2024`, 2024.

Xinyi Mou, Siyuan Wang, Zejun Li, Yulan He, and Zhongyu Wei. Hylat: Efficient multi-agent communication via hybrid latent-text protocol. *arXiv preprint arXiv:2605.25421*, 2026.

Zheng Nie, Ruolin Shen, Xinlei Yu, Bo Yin, Jiangning Zhang, and Xiaobin Hu. Skillgraph: Self-evolving multi-agent collaboration with multimodal graph topology. *arXiv preprint arXiv:2604.17503*, 2026.

Yanwen Peng, Delvin Ce Zhang, Xi Wang, and Nikolaos Aletras. Statebridge: Training-free hidden-state alignment for latent communication in llm multi-agent systems. *arXiv preprint arXiv:2608.13317*, 2026.

Pouya Pezeshkpour, Eser Kandogan, Nikita Bhutani, Sajjadur Rahman, Tom Mitchell, and Estevam Hruschka. Reasoning capacity in multi-agent systems: Limitations, challenges and human-centered solutions. *arXiv preprint arXiv:2402.01108*, 2024.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

xAI. RealWorldQA: A benchmark for real-world spatial understanding, 2024. URL `https://huggingface.co/datasets/xai-org/RealworldQA`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Zhen Yang, Xiaogang Xu, Wen Wang, Cong Chen, Xander Xu, and Ying-Cong Chen. Streaming communication in multi-agent reasoning. *arXiv preprint arXiv:2606.05158*, 2026.

Xinlei Yu, Chengming Xu, Guibin Zhang, Yongbo He, Zhangquan Chen, Zhucun Xue, Jiangning Zhang, Yue Liao, Xiaobin Hu, Yu-Gang Jiang, et al. Visual multi-agent system: Mitigating hallucination snowballing via visual flow. In *International Conference on Learning Representations*, volume 2026, pp. 43581–43606, 2026a.

Ye Yu, Heming Liu, Haibo Jin, Xiaopeng Yuan, Peng Kuang, and Haohan Wang. Learning to communicate: Toward end-to-end optimization of multi-agent language systems. *arXiv preprint arXiv:2604.21794*, 2026b.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *International Conference on Learning Representations*, volume 2025, pp. 75389–75428, 2025a.

Xinnong Zhang, Jiayu Lin, Xinyi Mou, Shiyue Yang, Xiawei Liu, Libo Sun, Hanjia Lyu, Yihang Yang, Weihong Qi, Yue Chen, et al. Socioverse: A world model for social simulation powered by llm agents and a pool of 10 million real-world users. *arXiv preprint arXiv:2504.10157*, 2025b.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuan-Jing Huang. Longagent: achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16310–16324, 2024.

Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *Advances in Neural Information Processing Systems*, 38:123389–123418, 2026.

Heng Zhou, Hejia Geng, Xiangyuan Xue, Li Kang, Yiran Qin, Zhiyong Wang, Zhenfei Yin, and Lei Bai. Reso: A reward-driven self-organizing llm-based multi-agent system for reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 15990–16009, 2025.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, et al. Latent collaboration in multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025.

APPENDIX CONTENTS

A Notation and Algorithm 15
A.1 Notation Summary 15
A.2 Algorithm 16
A.3 Communication Cost 16

B Protocol Specification 17
B.1 Semantic Vocabulary 17
B.2 Agent Roles and Prompt Templates 18

C Theoretical Analysis 22
C.1 Proof of Theorem 1 22
C.2 Controlled Study for Theorem 1 23
C.3 Proof of Theorem 2 25
C.4 Controlled Study for Theorem 2 25

D Experimental Details 26
D.1 Benchmarks and Evaluation Protocol 26
D.2 Implementation Details 27

E Case Studies 27
E.1 Success Case 27
E.2 Case Study on Theorem 1 30

F Limitations and Future Work 31

A Notation and Algorithm

A.1 Notation Summary

Table 7 summarizes the notation used throughout the paper, covering parameters for MAS structure, semantic negotiation protocol, latent transfer, and the theoretical and cost analyses.

Table 7: Notation summary.

Symbol	Description
MAS Structure	
$G = (V, E)$	Directed acyclic graph (DAG) representing the multi-agent system
$A_i = (f_\theta, r_i)$	Agent i with frozen backbone f_θ and role specification r_i
$\mathrm{Pa}(j)$	Set of direct senders (parents) of agent A_j; $\mathrm{Pa}(j) = \{i \mid (i,j) \in E\}$
$\mathrm{Ch}(i)$	Set of direct receivers (children) of agent A_i; $\mathrm{Ch}(i) = \{j \mid (i,j) \in E\}$
Message and Context	
x	Task instance (input problem)
$\mathcal{C}_i$	Upstream context available to agent A_i
(Y_i, H_i)	Decoded token sequence and final-layer hidden states from agent A_i
$\mathcal{U}_j$	Set of semantic items required by receiver A_j for downstream reasoning
Semantic Negotiation	
$\mathcal{O}$	Universal operation set; see Appendix B.1 for the full specification
$\mathcal{T}_\tau$	Task-family semantic vocabulary indexed by task family τ
$\mathcal{Q}_{j\to i}$	Structured request from receiver A_j to sender A_i
$q_m = (o_m, t_m, z_m, \pi_m)$	Request clause: operation, semantic type, target, and priority
$M_{j\to i}$	Number of request clauses issued from A_j to A_i; bounded by M_j
M_j	Maximum request budget of receiver A_j
Protocol	
$\mathcal{G}_\theta^P$	Probe-generation mode of frozen backbone f_θ
$\mathcal{G}_\theta^R$	Receiver-conditioned regeneration mode of frozen backbone f_θ
(Y_i^P, H_i^P)	Semantic probe (text) and its cached final-layer hidden states
$\mathcal{P}_j$	Probe set for receiver A_j from all direct senders in $\mathrm{Pa}(j)$
Ψ_θ	Receiver-side semantic audit function
$F_{j\to i}$	Feedback from receiver A_j to sender A_i; $\in \{\mathrm{ACK}, \mathrm{REQUEST}\}$
$(Y_{i\to j}^R, H_{i\to j}^R)$	Regenerated message and hidden states for receiver A_j
$(Y_{i\to j}^\star, H_{i\to j}^\star)$	Selected message state after negotiation for edge (i,j)
Latent Transfer	
$R_{i\to j}^\star$	Reference embeddings; $R_{i\to j}^\star = W_{\mathrm{emb}}[Y_{i\to j}^\star]$
Φ_{SB}	Training-free StateBridge alignment interface
$Z_{i\to j}$	Continuous prefix representation transferred to receiver A_j
X_j	Final input to downstream agent A_j; concatenation of prefixes and E_j
Theoretical Analysis	
$D_{i\to j}$	Indicator: probe Y_i^P is deficient for receiver A_j; $\in \{0,1\}$
$I_{j\to i}$	Indicator: A_j issues REQUEST to A_i; $\in \{0,1\}$
p	Deficiency prevalence; probability that an initial probe is insufficient
t	True-request rate; probability of requesting regeneration when deficiency exists
f	False-request rate; probability of requesting regeneration when probe is sufficient
δ	Expected downstream improvement from regenerating a deficient probe
ε	Expected degradation from unnecessarily regenerating a sufficient probe
$G_{i\to j}$	Expected downstream performance change induced by receiver-guided regeneration
w_u	Importance weight of semantic item $u \in \mathcal{U}_j$
s_u	Recovery probability of semantic item u per request clause
$C_j(M_j)$	Semantic coverage achieved under request budget M_j
Communication Cost	
C_i^P	Probe generation cost for sender A_i
C_j^F	Joint audit cost for receiver A_j
$C_{i\to j}^R$	Regeneration cost on edge (i,j)
$\rho_{i\to j}$	Request probability on edge (i,j); $\rho_{i\to j} = \Pr(I_{j\to i} = 1)$

A.2 ALGORITHM

Algorithm 1 presents the complete procedure of NegoLat, which separates semantic negotiation from latent transfer through four stages: probe generation, receiver audit, message formation, and downstream reasoning.

Algorithm 1 Receiver-Negotiated Latent Communication (NegoLat)

Require: Task instance x, MAS graph $G = (V, E)$
Require: Frozen backbone f_θ, role specifications $\{r_i\}_{i \in V}$
Require: Operation set $\mathcal{O}$, task-family vocabulary $\mathcal{T}_\tau$
Require: Request budgets $\{M_j\}_{j \in V}$, StateBridge interface Φ_{SB}

// Stage 1: Probe Generation
for each sender $A_i \in V$ **do**
  Obtain upstream context $\mathcal{C}_i$ from preceding agents
  $(Y_i^P, H_i^P) \leftarrow \mathcal{G}_\theta^P(x, r_i, \mathcal{C}_i)$ ▷ Generate semantic probe and cache hidden states
end for
// Stage 2: Receiver Audit and Feedback
for each receiver $A_j \in V$ with $\text{Pa}(j) \neq \emptyset$ **do**
  $\mathcal{P}_j \leftarrow \{Y_i^P\}_{i \in \text{Pa}(j)}$ ▷ Collect probes from all direct senders
  $\{F_{j \to i}\}_{i \in \text{Pa}(j)} \leftarrow \Psi_\theta(x, r_j, \mathcal{P}_j; \mathcal{O}, \mathcal{T}_\tau, M_j)$ ▷ Joint semantic audit
  for each sender $A_i \in \text{Pa}(j)$ **do**
    if $F_{j \to i} = \text{ACK}$ **then**
      continue ▷ Probe is sufficient; reuse cached states
    else ▷ $F_{j \to i} = \text{REQUEST}(\mathcal{Q}_{j \to i})$
      Send structured request $\mathcal{Q}_{j \to i}$ to sender A_i
    end if
  end for
end for
// Stage 3: Receiver-Guided Message Formation and Latent Transfer
for each edge $(i, j) \in E$ **do**
  if $F_{j \to i} = \text{ACK}$ **then**
    $(Y_{i \to j}^\star, H_{i \to j}^\star) \leftarrow (Y_i^P, H_i^P)$ ▷ Reuse cached probe states
  else
    $(Y_{i \to j}^R, H_{i \to j}^R) \leftarrow \mathcal{G}_\theta^R(x, r_i, \mathcal{C}_i; r_j, Y_i^P, \mathcal{Q}_{j \to i})$ ▷ Regenerate
    $(Y_{i \to j}^\star, H_{i \to j}^\star) \leftarrow (Y_{i \to j}^R, H_{i \to j}^R)$
  end if
  $R_{i \to j}^\star \leftarrow W_{\text{emb}}[Y_{i \to j}^\star]$ ▷ Obtain reference embeddings
  $Z_{i \to j} \leftarrow \Phi_{\text{SB}}(H_{i \to j}^\star, R_{i \to j}^\star)$ ▷ Align to receiver's embedding space
end for
// Stage 4: Downstream Reasoning
for each receiver $A_j \in V$ with $\text{Pa}(j) \neq \emptyset$ **do**
  $X_j \leftarrow [\text{Concat}_{i \in \text{Pa}(j)} Z_{i \to j}; E_j]$ ▷ Construct multimodal input
  Execute downstream reasoning with input X_j
end for
Return Final output from terminal agents

A.3 COMMUNICATION COST

Per sender, probe generation costs C_i^P; per receiver, the joint audit of all its senders costs C_j^F; per edge, regeneration costs $C_{i \to j}^R$ and is incurred only when requested, with probability $\rho_{i \to j} = \Pr(I_{j \to i} = 1)$. The expected total cost is

$$\mathbb{E}[C_{\text{NegoLat}}] = \sum_{i \in V} C_i^P + \sum_{j \in V} C_j^F + \sum_{(i,j) \in E} \rho_{i \to j} \, C_{i \to j}^R. \tag{9}$$

Since the probe replaces rather than supplements the sender's message, the first term substitutes for the generation cost that one-shot communication incurs anyway; the genuine overhead is confined

to the remaining two terms. This overhead is structurally small: audits are short structured replies, probes are amortized over $\mathrm{Ch}(i)$, ACK edges incur no regeneration, and the third term is paid only on the $\rho_{i\to j}$ fraction of edges.

B Protocol Specification

B.1 Semantic Vocabulary

We instantiate a task-family ontology $\mathcal{T}_\tau$ for each benchmark family (Math, VQA, Code). During receiver-side audit, the model only sees edge-specific ontology entries:

$$\mathcal{T}_\tau^{i\to j} = \mathcal{O}_i \cap \mathcal{I}_j, \tag{10}$$

where $\mathcal{O}_i$ is the sender output schema and $\mathcal{I}_j$ is the receiver-required schema. The semantic vocabulary is shown below.

Math Ontology (51 entries)

Problem (8): `problem_type`, `known_quantity`, `unknown_quantity`, `goal`, `relation`, `variable`, `symbol`, `problem_representation`.

Strategy (4): `method`, `submethod`, `rationale`, `applicability`.

Constraint (13): `explicit_constraint`, `implicit_constraint`, `domain`, `range`, `sign`, `integrality`, `unit`, `unit_relation`, `unit_conversion`, `boundary_condition`, `special_case`, `assumption`, `constraint_source`.

Plan (7): `reasoning_step`, `operation`, `step_order`, `step_dependency`, `constraint_binding`, `required_input`, `expected_intermediate`.

Solution (7): `derivation`, `transformation`, `calculation`, `intermediate_result`, `candidate_result`, `plan_adherence`, `result_goal_alignment`.

Verification (12): `verification_method`, `verification_evidence`, `substitution_check`, `reverse_check`, `constraint_check`, `risk_status`, `risk_location`, `risk_type`, `risk_evidence`, `review_recommendation`, `uncertainty`, `final_answer`.

VQA Ontology (77 entries)

Scene (15): `scene_summary`, `entity`, `object`, `person`, `background`, `attribute`, `state`, `action`, `visible_text`, `number`, `layout`, `position`, `image_quality`, `occlusion`, `uncertainty`.

Question (9): `question_type`, `answer_type`, `target_entity`, `comparison_entity`, `qualifier`, `coreference`, `polarity`, `evidence_requirement`, `output_constraint`.

Grounding (7): `region`, `target_region`, `reference_object`, `relevant_region`, `localization`, `localization_evidence`, `region_scope`.

Evidence (21): `visual_evidence`, `attribute_evidence`, `text_evidence`, `action_evidence`, `count`, `count_evidence`, `comparison`, `comparison_evidence`, `spatial_relation`, `direction`, `orientation`, `viewpoint`, `distance`, `adjacency`, `overlap`, `containment`, `occlusion_relation`, `depth_order`, `observation`, `inference`, `confidence`.

Reasoning (9): visual_spatial_reasoning, counting_reasoning, comparison_reasoning, causal_relation, multi_step_reasoning, reasoning_step, candidate_answer, candidate_rationale, uncertainty.

Knowledge (9): domain_knowledge, scientific_knowledge, cultural_knowledge, commonsense_knowledge, functional_knowledge, causal_knowledge, knowledge_applicability, knowledge_source, knowledge_uncertainty.

Verification (7): target_consistency, evidence_sufficiency, reasoning_correctness, visual_knowledge_consistency, candidate_consistency, source_reliability, final_answer.

Code Ontology (91 entries)

Specification (15): task_goal, function_name, function_signature, parameter, parameter_type, return_type, input_semantics, output_semantics, input_structure, output_structure, element_type, order_requirement, input_mutability, example, acceptance_criteria.

Constraint (23): input_size, numeric_range, data_type, input_domain, empty_input, negative_value_case, duplicate_case, ordering_property, uniqueness_property, time_limit, space_limit, complexity_requirement, special_case, degenerate_case, hard_constraint, soft_requirement, assumption, constraint_source, overflow_risk, precision_risk, index_risk, recursion_risk, memory_risk.

Algorithm (13): algorithm, data_structure, state_representation, core_idea, key_operation, algorithm_stage, subproblem, invariant, recurrence_relation, correctness_argument, time_complexity, space_complexity, edge_case_strategy.

Plan (15): implementation_step, control_flow, data_flow, function_structure, call_relation, loop_structure, branch_condition, return_position, variable, state, initialization, update_rule, pseudocode, interface_binding, boundary_handling.

Implementation (12): candidate_code, syntax, identifier, import, dependency, api_usage, logic, plan_adherence, boundary_handling, output_format, return_value, numerical_safety.

Verification (13): test_case, example_trace, execution_trace, test_result, interface_check, syntax_check, logic_check, complexity_check, numerical_safety_check, boundary_check, output_format_check, final_code, uncertainty.

Operation set. Requests use $\mathcal{O} = \{\text{ADD}, \text{REFINE}, \text{VERIFY}, \text{DISAMBIGUATE}, \text{RESOLVE}\}$. Internal defect diagnosis maps to one of these operations before forming the final request tuple (o_m, t_m, z_m, π_m).

B.2 Agent Roles and Prompt Templates

Figure 6 introduces the task-specific role topology. Here we provide the role-prompt templates used in practice. Each role prompt is designed to enforce *stage-specific responsibility*: upstream roles provide structured intermediate products, while only the terminal verifier-answerer role outputs the authoritative final answer.

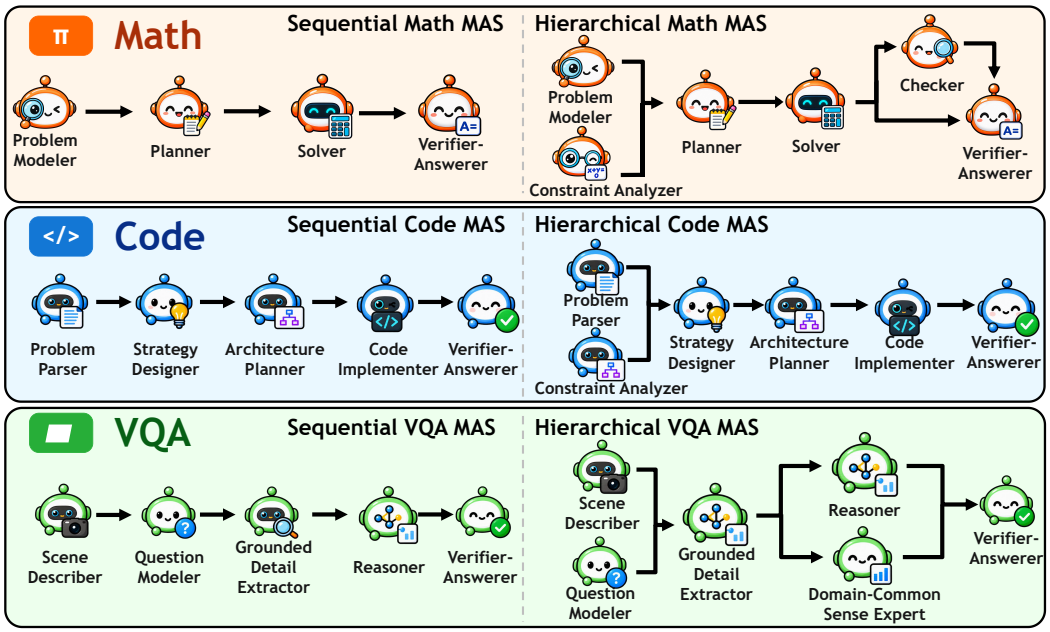

Figure 6: **MAS settings.** We instantiate Sequential and Hierarchical MAS structures for mathematical reasoning, code generation, and visual question answering tasks.

Math 6-Role Prompt Pool

M1 Problem Modeler. Extract known quantities, unknowns, target, and basic constraints; classify problem type and recommend a strategy with brief rationale. Do not compute or output a final answer. Produce a compact structured problem model for downstream planning.

M2 Constraint Analyzer. Identify explicit/implicit constraints, domains, units, ranges, rounding requirements, and exceptional cases. Do not solve the problem or output the final answer. Return a concise constraint and risk list.

M3 Planner. Convert upstream problem model and constraints into an executable sequence of dependent solution steps; specify where each key constraint applies. Do not perform calculations or output the final answer.

M4 Solver. Execute the supplied plan carefully; show necessary derivation and intermediate results; output a clearly labeled *Candidate Result*. Do not present it as an authoritative final answer.

M5 Checker. Audit the candidate for arithmetic errors, invalid assumptions, missed constraints, boundary cases, unit issues, and suspicious steps. Output a risk report with location, issue type, evidence, and recheck suggestion. Do not replace with a new final answer.

M6 Verifier-Answerer. Globally verify candidate reasoning and checker report, repair errors if needed, then solve the authoritative task. Follow task output requirements exactly, and place the final mathematical answer in `\boxed{}` for extraction.

VQA 6-Role Prompt Pool

V1 Scene Describer. Scan the original image globally; record visible objects, attributes, layout, readable text, occlusion, and image-quality limitations in compact structured form. Report only direct observations and material uncertainty; do not answer the question.

V2 Question Modeler. Model the question without answering it: identify question type, target entity, answer type, relevant options, and required visual/knowledge evidence. Flag negation/comparison/counting/spatial/OCR/causal requirements and output an evidence checklist.

V3 Grounded Detail Extractor. Reinspect the image against the evidence checklist; localize relevant regions and extract exact attributes, counts, visible text, actions, and local relations. Separate OBSERVED from UNCERTAIN evidence; do not choose an answer.

V4 Reasoner. Use grounded visual evidence for task-required reasoning (spatial, directional, counting, comparison, action, causal, domain, commonsense). Prefer direct image evidence; output a candidate answer with key supporting evidence and uncertainty/conflict notes.

V5 Domain-CommonSense Expert. Provide only necessary scientific/cultural/functional/commonsense knowledge for interpreting grounded visual evidence. Distinguish image observations from background knowledge; assess plausibility; output concise candidate judgment (not final authoritative answer).

V6 Verifier-Answerer. Integrate grounded evidence, candidate reasoning, and domain-commonsense analysis. Verify target alignment, visual support, counts, OCR, relations, and reasoning consistency; resolve conflicts via reinspection; then output final answer in benchmark-required format.

Code 6-Role Prompt Pool

R1 Problem Parser. Parse the programming task into a structured specification: function signature, inputs/outputs, required behavior, and interface constraints. Do not propose algorithms or write code.

R2 Constraint Analyzer. Identify explicit/implicit constraints, boundary inputs, invalid assumptions, and language/library restrictions. Return concise constraint and edge-case list; do not write code.

R3 Strategy Designer. Choose suitable algorithms/data structures; provide concise approach, complexity expectations, and rationale. Do not write implementation code.

R4 Architecture Planner. Synthesize specification, strategy, and constraints into concrete code structure and precise pseudocode, including control flow and edge cases; do not output final implementation.

R5 Code Implementer. Implement the plan as complete runnable candidate code that obeys requested signature and interface. Mark output as candidate implementation, not authoritative final code.

R6 Verifier-Answerer. Audit candidate code against original specification: syntax, signature/interface, functional logic, algorithmic soundness, complexity, edge cases (empty/minimal/duplicates/negative/extremes), overflow, and output format. Repair confirmed defects without changing required interface, then output the complete verified final code in authoritative task format.

Prompt and Protocol Templates. Beyond role prompts, NegoLat uses explicit templates for probe generation, receiver audit, and regeneration. We list the canonical structure below (with placeholders in braces).

Template A: Probe Prompt (Sender → Probe)

```
You are {role_spec.name} (Role {role_spec.role_id}).
Goal: {role_spec.goal}
Restrictions:
- {restriction_1}
- {restriction_2}
```

```
[omit this block if no restrictions]
Your output should cover these semantic types:
- {output_sem_1}
- {output_sem_2}
- {output_sem_3}
- {output_sem_4}
- {output_sem_5}
- ...  and {N_more} more
[show top-5 only; omit block if empty]
TASK:
{task}
UPSTREAM CONTEXT:
{upstream_context}
Provide a concise structured semantic probe.
```

Template B: Audit Prompt (Receiver → ACK/REQUEST)

```
You are the downstream receiver in a receiver-negotiated
communication protocol.
Determine whether the sender's semantic probe is sufficient for YOUR
role.
RECEIVER ROLE: {receiver_role_spec.name} (ID:
{receiver_role_spec.role_id})
RECEIVER GOAL: {receiver_role_spec.goal}
RECEIVER NEEDS: {first_10_input_semantics} {...  if more}
SENDER ROLE: {sender_role}
TASK: {task}
SENDER PROBE: {probe_text or [empty probe]}
ALLOWED SEMANTIC TYPES FOR THIS EDGE:
{edge_ontology_joined}
ALLOWED OPERATIONS: {operations_joined}
RULES: JSON only; ACK if sufficient; else REQUEST with at most
{max_requests};
semantic_type must be from allowed list; do not solve sender task;
priority in {1,2,3}.
REQUEST FORMAT:
{
"status":  "REQUEST",
"preserve":  ["already sufficient content"],
"requests":  [
{"operation":  "VERIFY",
"semantic_type":  "plan.reasoning_step",
"target":  "specific gap",
"priority":  3}
]
}
Your response (JSON only):
```

Template C: Regeneration Prompt (Sender → Regenerated Probe)

```
You are the sender in a receiver-negotiated communication protocol.
The receiver returned REQUEST. Regenerate a COMPLETE and COHERENT
message.
SENDER ROLE: {sender_role_spec.name} (ID: {sender_role_spec.role_id})
SENDER GOAL: {sender_role_spec.goal}
RECEIVER ROLE: {receiver_role_spec.name} (ID:
{receiver_role_spec.role_id})
RECEIVER GOAL: {receiver_role_spec.goal}
TASK: {task}
YOUR PREVIOUS PROBE: {original_probe}
ALREADY SUFFICIENT SEMANTICS (preserve):
```

```
- {preserve_1}
- {preserve_2}
[(none) if empty]
RECEIVER REQUESTS (address all):
1.  [{op_1}] {sem_type_1}:  {target_1} (priority:  {priority_1})
2.  [{op_2}] {sem_type_2}:  {target_2} (priority:  {priority_2})
[(none) if empty]
REQUIREMENTS: preserve sufficient claims; address every request;
output one coherent message;
do not merely append patches; do not violate role boundary; do not
output forbidden final answers.
```

Canonical ACK/REQUEST JSON Templates

ACK template

```
{"status":"ACK", "preserve":[], "requests":[]}
```

REQUEST template

```
{ "status":"REQUEST",
  "preserve":["problem goal","known quantities"],
  "requests":[
    {"operation":"VERIFY",
     "semantic_type":"solution.intermediate_result",
     "target":"Step 3 substitution","priority":3},
    {"operation":"ADD",
     "semantic_type":"constraint.range",
     "target":"valid range of x","priority":2}]
}
```

Schema constraint. Each request clause is a quadruple $q_m = (o_m, t_m, z_m, \pi_m)$, where $o_m \in \{\text{ADD}, \text{REFINE}, \text{VERIFY}, \text{DISAMBIGUATE}, \text{RESOLVE}\}$, $t_m \in \mathcal{T}_\tau^{i \to j}$, and $\pi_m \in \{1, 2, 3\}$.

C Theoretical Analysis

This appendix provides formal proofs for the two theorems presented in Section 3.2 (Theorem 1 and Theorem 2). For clarity, we use a sequential topology (i.e., $|\text{Pa}(j)| = |\text{Ch}(i)| = 1$) as a running example; the results generalize to arbitrary DAGs by applying the analysis independently to each edge $(i, j) \in E$.

C.1 Proof of Theorem 1

Theorem 1 (When Receiver-Guided Regeneration Is Beneficial). *On edge (i, j), the expected downstream performance change induced by receiver-guided regeneration is (Eq. 7)*

$$G_{i \to j} = pt\delta - (1 - p)f\varepsilon.$$

Therefore, receiver-guided regeneration improves expected downstream performance if and only if $pt\delta > (1 - p)f\varepsilon$.

Proof. Consider a communication edge $(i, j) \in E$. We define two binary random variables:

- $D_{i \to j} \in \{0, 1\}$: indicates whether the initial semantic probe Y_i^P is *deficient* with respect to the information needs of receiver A_j. A probe is deficient when it omits high-importance semantic items from $\mathcal{U}_j$.
- $I_{j \to i} \in \{0, 1\}$: indicates whether receiver A_j issues a REQUEST to sender A_i. Specifically, $I_{j \to i} = 1$ if and only if $F_{j \to i} = \text{REQUEST}(\mathcal{Q}_{j \to i})$.

We further define:

$$p = \Pr(D_{i\to j} = 1), \tag{11}$$
$$t = \Pr(I_{j\to i} = 1 \mid D_{i\to j} = 1), \tag{12}$$
$$f = \Pr(I_{j\to i} = 1 \mid D_{i\to j} = 0), \tag{13}$$

where p is the *deficiency prevalence*, t is the *true-request rate* (correctly requesting regeneration when a deficiency exists), and f is the *false-request rate* (incorrectly requesting regeneration when the probe is already sufficient).

Let $\delta > 0$ denote the expected downstream improvement from regenerating a genuinely deficient probe, and let $\varepsilon \geq 0$ denote the expected degradation from unnecessarily regenerating an already sufficient probe.

The joint distribution over $(D_{i\to j}, I_{j\to i})$ can be decomposed into four mutually exclusive cases:

Case	Probability	Performance Change
$D = 1, I = 1$ (true positive)	$p \cdot t$	$+\delta$
$D = 1, I = 0$ (false negative)	$p \cdot (1 - t)$	0
$D = 0, I = 1$ (false positive)	$(1 - p) \cdot f$	$-\varepsilon$
$D = 0, I = 0$ (true negative)	$(1 - p) \cdot (1 - f)$	0

The expected downstream performance change is obtained by summing over all cases:

$$\begin{aligned} G_{i\to j} &= \Pr(D = 1, I = 1) \cdot \delta + \Pr(D = 1, I = 0) \cdot 0 \\ &\quad + \Pr(D = 0, I = 1) \cdot (-\varepsilon) + \Pr(D = 0, I = 0) \cdot 0 \\ &= pt\delta - (1 - p)f\varepsilon. \end{aligned} \tag{14}$$

Receiver-guided regeneration improves expected downstream performance when $G_{i\to j} > 0$, which holds if and only if:

$$pt\delta > (1 - p)f\varepsilon. \tag{15}$$

This condition can be rewritten as:

$$\frac{t}{f} > \frac{(1 - p)\varepsilon}{p\delta}, \tag{16}$$

which shows that effective negotiation requires the ratio of true-request rate to false-request rate to exceed a threshold determined by the deficiency prevalence and the benefit-to-harm ratio. □

Interpretation. Theorem 1 reveals that the effectiveness of receiver-guided regeneration depends on the *precision* of the receiver's deficiency detection. Simply increasing the request rate (higher t and f) is not sufficient; what matters is the ability to selectively trigger regeneration when deficiencies genuinely exist while avoiding unnecessary regeneration when probes are already adequate. This justifies the design of structured semantic requests with bounded budgets, which encourage targeted rather than indiscriminate feedback.

C.2 Controlled Study for Theorem 1

We conduct a controlled study on the AQuA mathematical reasoning benchmark to empirically validate Theorem 1 (see also Fig. 3 in Section 4). The study focuses on the **Planner→Solver** edge within a hierarchical MAS topology (Fig. 6), where the Planner generates step-by-step reasoning plans and the Solver executes them to select the correct answer from $\{A, B, C, D, E\}$.

Data Construction. We construct 200 paired (clean, corrupted) samples following a systematic corruption protocol:

1. **Clean source extraction.** We extract Planner outputs from TextMAS Sequential-topology predictions on AQuA, filtering for well-structured plans with ≥300 characters and ≥4 explicit reasoning steps.

2. **Ontology-level corruption.** Each corrupted sample is injected with 3 defects (total 600 defects across 200 samples), targeting Planner→Solver semantic fields. The corruption types are designed to be detectable by the Solver's semantic audit.

Table 8 summarizes the five corruption types used in this study.

Table 8: Corruption types for the Theorem 1 controlled study. Each type targets a specific Planner→Solver semantic field from the Math ontology (Appendix B.1) and has an expected Solver operation for recovery. Counts are over all 600 injected defects (200 samples × 3 defects).

Corruption Type	Semantic Field	Description	Op.	#
`missing_intermediate_step`	`plan.reasoning_step`	Drop a critical calculation step; insert skip-gap	ADD	148
`wrong_step_dependency`	`plan.step_dependency`	Inject cyclic or future-step dependency	VERIFY	158
`missing_constraint_binding`	`plan.constraint_binding`	Remove constraints; inject conflict instruction	REFINE	154
`numeric_calculation_error`	`plan.expected_intermediate`	Linked numeric error chain across steps	VERIFY	140
`numeric_option_error`	`solution.candidate_result`	Assign the computed value to the wrong answer option	VERIFY	0

The 600 defects are distributed as: `missing_intermediate_step`: 148, `wrong_step_dependency`: 158, `missing_constraint_binding`: 154, `numeric_calculation_error`: 140, `numeric_option_error`: 0 (defined in the ontology but not realized in this injection draw).

Measurement Protocol. We measure the four parameters $(p, t, f, \delta, \varepsilon)$ through three controlled experiments:

(a) Measuring t and f (request rates). We run the Solver's semantic audit on both clean and corrupted probes, counting how often it issues a REQUEST:

$$f = \frac{\#\text{REQUEST on clean samples}}{200}, \tag{17}$$

$$t = \frac{\#\text{REQUEST on corrupted samples}}{200}. \tag{18}$$

This measures the Solver's ability to distinguish deficient from sufficient probes.

(b) Measuring δ (benefit of repair). For corrupted samples where the Solver issues REQUEST, we compare:

- **Before repair:** Corrupted probe → Solver → Verifier → Accuracy = $\text{Acc}_{\text{before}}$
- **After repair:** Corrupted probe → REQUEST → Planner regeneration → Solver → Verifier → Accuracy = $\text{Acc}_{\text{after}}$

The improvement is $\delta = \text{Acc}_{\text{after}} - \text{Acc}_{\text{before}}$.

(c) Measuring ε (cost of false repair). For clean samples where the Solver incorrectly issues REQUEST, we compare:

- **No extra REQUEST:** Clean probe → full pipeline → Accuracy = $\text{Acc}_{\text{normal}}$
- **Forced unnecessary repair:** Clean probe → forced regeneration → pipeline → Accuracy = $\text{Acc}_{\text{repaired}}$

The degradation is $\varepsilon = \text{Acc}_{\text{normal}} - \text{Acc}_{\text{repaired}}$.

Results. The empirical measurements (consistent with Fig. 3) are:

- **Request rates:** $t = 0.645$ (corrupted probes) and $f = 0.035$ (clean probes).
- **Repair benefit:** among correctly triggered requests, $\text{Acc}_{\text{before}} = 48.06\%$ $(62/129)$ rises to $\text{Acc}_{\text{after}} = 66.67\%$ $(86/129)$, yielding $\delta = 0.186$ $(+18.6$ percentage points).
- **False repair cost:** only 7 of 200 clean probes are requested by mistake; among these seven, accuracy falls from 71.43% $(5/7)$ to 57.14% $(4/7)$, yielding $\varepsilon = 0.143$ $(-14.3$ percentage points).

Setting $p = 0.5$ (balanced clean/corrupted distribution) and substituting into Eq. 7:

$$G = pt\delta - (1-p)f\varepsilon = +5.75 \text{ pp}.$$

Thus the expected gain is positive: repairing genuinely deficient probes ($pt\delta$) outweighs the rare cost of unnecessary regeneration ($(1-p)f\varepsilon$), validating the prediction of Theorem 1.

C.3 Proof of Theorem 2

Theorem 2 (Diminishing Returns of Semantic Requests). *Let $\mathcal{U}_j$ denote receiver-relevant semantic items with weight $w_u \geq 0$ and recovery probability $s_u \in [0,1]$. Under budget M_j (Eq. 8):*

$$C_j(M_j) = \frac{\sum_{u \in \mathcal{U}_j} w_u \left[1 - (1-s_u)^{M_j}\right]}{\sum_{u \in \mathcal{U}_j} w_u}.$$

Then $C_j(M_j)$ increases with M_j but exhibits diminishing marginal gains.

Proof. Each request clause provides an independent opportunity to recover item u with probability s_u. The probability that u is not recovered after M_j requests is $(1-s_u)^{M_j}$, so recovery probability is $1-(1-s_u)^{M_j}$. By linearity of expectation, $C_j(M_j) = \frac{\sum_u w_u[1-(1-s_u)^{M_j}]}{\sum_u w_u}$.

Monotonicity. The marginal gain is:

$$\Delta C_j(M_j) = C_j(M_j+1) - C_j(M_j) = \frac{\sum_u w_u (1-s_u)^{M_j} s_u}{\sum_u w_u} \geq 0. \tag{19}$$

Diminishing returns. The change in marginal gain is:

$$\Delta C_j(M_j+1) - \Delta C_j(M_j) = -\frac{\sum_u w_u s_u^2 (1-s_u)^{M_j}}{\sum_u w_u} \leq 0. \tag{20}$$

Thus marginal gains decrease monotonically. As $M_j \to \infty$, $C_j(M_j) \to 1$ for items with $s_u > 0$. □

Interpretation. A small number of well-targeted requests ($M_j \approx 3$) captures most recoverable information, while further requests yield diminishing returns. This justifies bounded request budgets, consistent with Theorem 2.

C.4 Controlled Study for Theorem 2

We empirically validate Theorem 2's prediction of diminishing returns in semantic coverage as the request budget M_j increases (see also Fig. 4(e) in Section 4). Using the same AQuA dataset and Planner→Solver edge as in the Theorem 1 study (Appendix C.2), we vary the Solver's request budget and measure both defect coverage and downstream accuracy.

Experimental Setup. We reuse the same 200 corrupted samples from the Theorem 1 study, each containing exactly 3 injected defects (from the five types in Table 8). Before varying the budget, we run a preliminary audit to classify each sample by its audit status: REQUEST (at least one semantic request issued) or ACK (no request). Of the 200 samples, 94 yield REQUEST and 106 yield ACK. We retain only the 94 REQUEST samples for the budget ablation, since ACK samples have zero coverage under any budget and are uninformative for the diminishing-returns analysis.

Note that the Theorem 1 study (Appendix C.2) reports a different REQUEST count (129/200) because it uses a separate audit prompt without the budget truncation logic; the two experiments share the same corrupted data but differ in audit configuration, leading to different request rates.

For each budget level $M_j \in \{0, 1, 2, 3, 4\}$, we run the Solver's semantic audit on the 94 REQUEST samples with the constraint that it may issue at most M_j request clauses per sample. We then measure:

- **Defect coverage** $C_j(M_j)$: fraction of injected defects that appear in the Solver's REQUEST.
- **Downstream accuracy**: final answer accuracy after regeneration guided by the M_j-bounded request.

Results. Table 9 reports the coverage and accuracy as a function of the request budget.

Table 9: Effect of request budget M_j on defect coverage and downstream accuracy (computed on the 94 REQUEST samples). Diminishing returns are evident: coverage gains decrease as M_j increases.

M_j	Coverage $C_j(M_j)$	ΔC_j (marginal)	Accuracy (%)	Δ Acc (marginal)
0	0.0%	—	52.13	—
1	33.0%	+33.0	75.53	+23.40
2	59.2%	+26.2	76.60	+1.07
3	80.5%	+21.3	80.85	+4.25
4	88.3%	+7.8	77.66	-3.19

Analysis. The empirical results closely match the theoretical predictions of Theorem 2:

1. **Trend with budget.** Coverage increases monotonically with M_j, while accuracy rises from $M_j = 0$ to $M_j = 3$ and then declines at $M_j = 4$.
2. **Diminishing marginal returns.** The marginal coverage gain ΔC_j decreases monotonically: $+33.0 \to +26.2 \to +21.3 \to +7.8$. Accuracy improves quickly at small budgets, peaks at $M_j = 3$, and then declines at $M_j = 4$.
3. **Practical budget recommendation.** At $M_j = 3$, the system reaches 80.5% semantic coverage and the best accuracy (80.85%). Increasing to $M_j = 4$ adds only +7.8% coverage while reducing accuracy by 3.19 points, indicating over-specification noise.

These findings validate the theoretical framework of Theorem 2 and provide practical guidance for setting the request budget: a small number of well-targeted semantic requests ($M_j \approx 3$) captures most of the recoverable information, while additional requests offer diminishing returns.

D Experimental Details

D.1 Benchmarks and Evaluation Protocol

Table 10 summarizes the upstream scorer and metric for each benchmark. All three methods (TextMAS, StateBridge (Peng et al., 2026), NegoLat) share identical evaluation pipelines.

Benchmark Grouping. We organize benchmarks into three task families:

- **Math Reasoning**: AQuA (Ling et al., 2017) (5-way MCQ), AIME24/25 (Maxwell-Jia, 2024; math ai, 2025) (competition math), and HMMT Feb 25/26 (Dekoninck et al., 2026). Qwen2.5-Math extracts answers and performs equivalence judgment; lm-eval and MathArena use exact/symbolic matching.
- **Visual QA**: MMBench EN-dev (Liu et al., 2024), MMStar (Chen et al., 2024a), RealWorldQA (xAI, 2024), and HallusionBench (Guan et al., 2024). MMBench and MMStar aggregate option-level scores; RealWorldQA uses exact match; HallusionBench reports `aAcc`.

Table 10: Evaluation protocol per dataset.

Dataset	Scorer	Metric
AQuA	Qwen2.5-Math	accuracy
AIME24/25	lm-eval	accuracy
HMMT Feb 25/26	MathArena	accuracy
MMBench EN-dev	lmms-eval	gpt_eval_score
MMStar	lmms-eval	average
RealWorldQA	lmms-eval	exact_match
HallusionBench	EvalScope	aAcc
HumanEval, MBPP	EvalPlus	Base/Plus pass@1

- **Code Generation**: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). EvalPlus executes original and augmented test suites, reporting Base and Plus pass@1.

D.2 Implementation Details

Inference Configuration. We follow the default decoding setups of the respective evaluation harnesses for each task family (math, VQA, and code), keeping generation hyperparameters consistent across compared methods within the same benchmark. On Llama-3.2-3B-Instruct, we run AIME25 7 times and AIME24 and HMMT26 4 times each, and report the average.

Latent Transfer. Both StateBridge (Peng et al., 2026) and NegoLat use the StateBridge aligner with max_prefix_tokens $= 256$. The aligned continuous prefix is injected via `inputs_embeds` at the receiver's input sequence front.

Protocol Settings. The per-edge request budget is fixed at $M_j = 3$ across all datasets, consistent with Theorem 2. Receiver feedback follows the JSON schema enforced by `ReceiverFeedback`/`RequestClause` validators.

E Case Studies

E.1 Success Case

We present a representative success case under the Hierarchical topology. This example is the same as Fig. 5 in the main text, and provides the full per-layer negotiation trace. It demonstrates how receiver-negotiated communication identifies semantic deficiencies in upstream probes and guides message refinement to produce the correct answer.

Success Case: NegoLat on MMBench (Hierarchical Topology)

Question: In the picture, which direction is the man wearing a hat facing?
Options: A. facing the camera B. back to the camera C. facing the little boy D. facing the floor
Gold Answer: C **NegoLat Prediction:** C (✓)

Layer 1: Scene Describer (root agent, processes image directly)

Input: Task prompt + Image
Probe Output: `A. facing the camera`
Target: Grounded Detail Extractor
Receiver Feedback: `ACK` — Probe accepted without modification.
Note: The scene describer incorrectly interprets the man's orientation. This error will be corrected downstream through negotiation.

Layer 1: Question Modeler (root agent, analyzes question structure)

Input: Task prompt

Probe Output:

```
question.question_type: directional
question.answer_type: option
question.target_entity: man wearing a hat
question.evidence_requirement: visual orientation of the man's
    face and body relative to the camera and the boy
question.spatial_relation: requires determining if the man's
    face is directed toward the camera, away, or toward the boy
question.inference_required: no - directly observable
```

Target: Grounded Detail Extractor

Receiver Feedback: `ACK` — Question decomposition is complete.

Layer 2: Grounded Detail Extractor (receives from Scene Describer + Question Modeler)

Upstream Context:

- From Scene Describer: `"A. facing the camera"`
- From Question Modeler: question decomposition (target entity, spatial relation, etc.)

Agent Output: `C. facing the little boy`

Probe to Reasoner:

```
grounding.region: man wearing a hat
grounding.localization: behind the little boy, positioned
    in the background
grounding.localization: facing the camera [repeated entries]
```

Receiver Audit: `REQUEST`

```
{
  "status": "REQUEST",
  "requests": [
    {"operation": "VERIFY",
     "semantic_type": "evidence.visual_evidence",
     "target": "man wearing a hat's orientation", "priority": 3},
    {"operation": "VERIFY",
     "semantic_type": "evidence.direction",
     "target": "man wearing a hat's facing direction", "priority": 3}
  ]
}
```

The receiver detects inconsistency: probe says "behind the little boy" but also "facing the camera". It requests verification of visual evidence.

Regenerated Message: After regeneration, the sender refines localization to emphasize the man's position relative to the boy, providing clearer spatial grounding for downstream reasoning.

Layer 3: Domain-CommonSense Expert (receives from Grounded Detail Extractor)

Upstream Context: Refined grounding information from Layer 2
Agent Output: `C. facing the boy`

Probe to Verifier-Answerer:

```
knowledge.domain_knowledge: The man is positioned behind the
    child. His head is turned slightly toward the child, and
    his gaze is directed toward the child's position,
    indicating orientation toward the child.
knowledge.commonsense_knowledge: In typical interpersonal
    interactions, when one person is positioned behind another
    and their gaze is directed toward the other person, it
    implies they are facing that person.
knowledge.functional_knowledge: The man's posture and gaze are
    consistent with someone engaging with the child in a
    shared activity (e.g., playing a game).
```

Receiver Audit: `REQUEST`

```
{
  "status": "REQUEST",
  "requests": [
    {"operation": "ADD",
     "semantic_type": "reasoning.candidate_answer",
     "target": "Output exactly one candidate letter {A,B,C,D}
               as 'Candidate Answer: X'", "priority": 3}
  ]
}
```

The receiver requests a properly formatted candidate answer for final aggregation.

Regenerated Message: `Candidate Answer: C`

Layer 3: Reasoner (receives from Grounded Detail Extractor)

Upstream Context: Refined grounding information from Layer 2
Agent Output: `Candidate Answer: C`

Initial Probe: `reason` (incomplete output)
Receiver Audit: `REQUEST` — Requests formatted candidate answer.
Regenerated Message: `Candidate Answer: C`

Layer 4: Verifier-Answerer (final aggregator)

Upstream Context:

- From Reasoner: `Candidate Answer: C`
- From Domain-CommonSense Expert: `Candidate Answer: C`

Final Output: **C**

Both upstream agents agree on option C after negotiation-guided refinement. The verifier outputs the correct answer.

Analysis: Why Negotiation Succeeded

1. **Error Detection:** The Scene Describer initially misidentified the man's orientation as "facing the camera" (option A). This error propagated to the Grounded Detail Extractor's probe.
2. **Semantic Deficiency Identification:** The Reasoner detected an inconsistency in the probe: the spatial description ("behind the little boy") contradicted the orientation claim ("facing the camera").
3. **Targeted Feedback:** The receiver issued `VERIFY` requests for `evidence.visual_evidence` and `evidence.direction`, prompting the sender to re-examine the visual grounding.
4. **Corrective Regeneration:** After regeneration, the refined message provided clearer evidence that the man's gaze and body orientation are directed toward the child.
5. **Consensus Formation:** Both Layer-3 agents independently concluded "C. facing the little boy" based on the refined upstream information, and the final verifier aggregated their consensus to produce the correct answer.

Key Insight: Without negotiation, the initial error from the Scene Describer would likely propagate through the pipeline. NegoLat's receiver-driven feedback mechanism caught and corrected this semantic deficiency before final answer generation.

E.2 Case Study on Theorem 1

To complement the VQA success case above, we present two cases from the Theorem 1 controlled study (Appendix C.2) that illustrate both the benefit and the risk of receiver-negotiated regeneration. Both cases use the AQuA dataset under a two-agent Sequential topology (Planner $\rightarrow$ Solver).

Case A: Corrupted Sample — Successful Recovery via REQUEST

Question: A trader sold an article at a profit of 20% for Rs.360. What is the cost price of the article?
Options: (A) 270 (B) 300 (C) 280 (D) 320 (E) 315
Gold Answer: B **Before Regen:** A (×) **After Regen:** B (✓)

Injected Defects (3 total)

1. `numeric_calculation_error`: Changed key multiplier from 1.20 to 3.4
2. `missing_constraint_binding`: Removed constraint lines; injected "Ignore domain constraints"
3. `wrong_step_dependency`: Injected cyclic dependency Step 2 $\leftrightarrow$ Step 5

Receiver Audit Output

```
{
  "status": "REQUEST",
  "requests": [
    {"operation": "VERIFY", "semantic_type": "plan.step_dependency",
     "target": "Step 2 depends on Step 5", "priority": 3},
    {"operation": "REFINE", "semantic_type": "plan.constraint_binding",
     "target": "Ignore positivity/integrality/domain constraints", "priority": 3},
    {"operation": "VERIFY", "semantic_type": "plan.expected_intermediate",
     "target": "use 3.4 as the key intermediate result", "priority": 3}
  ]
}
```

Why This Works: The receiver precisely identifies all three injected defects—cyclic dependency, missing constraints, and wrong intermediate value—and issues targeted `VERIFY`/`REFINE` requests. After regeneration, the Planner produces a corrected plan using SP $= 1.20 \times$ CP, leading to the correct answer CP $= 360/1.20 = 300$ (option B).

Case B: Clean Sample — False Positive REQUEST Causes Regression

Question: What is the largest integral value of k for which the quadratic equation $x^2 - 5x + k = 0$ will have two real and distinct roots?
Options: (A) 9 (B) 7 (C) 3 (D) 8 (E) 12
Gold Answer: C **Before Regen:** C (✓) **After Regen:** E (×)

Original Clean Planner Output (Correct)

The clean Planner correctly derives: discriminant $D = 25 - 4k > 0 \Rightarrow k < 6.25$. The largest integer is $k = 6$, but 6 is not among the options. Among the given choices, only (C) $k = 3$ satisfies $k < 6.25$; larger options (7, 8, 9, 12) all violate the constraint. The Planner notes this discrepancy but correctly selects (C).

Receiver Audit Output (False Positive)


```
{
  "status": "REQUEST",
  "requests": [
    {"operation": "ADD", "semantic_type": "plan.reasoning_step",
     "target": "Verify that k = 6 satisfies D > 0", "priority": 3},
    {"operation": "REFINE", "semantic_type": "plan.constraint_binding",
     "target": "Ensure k must be integer and k < 6.25", "priority": 3},
    {"operation": "VERIFY", "semantic_type": "plan.expected_intermediate",
     "target": "discriminant when k = 6 is 1", "priority": 3}
  ]
}
```


Why This Fails: The receiver misinterprets the Planner's nuanced reasoning (acknowledging 6 is correct but absent from options) as a deficiency. The `ADD`/`REFINE` requests prompt regeneration, but the regenerated plan over-focuses on $k = 6$ and loses track of the answer-choice constraint, ultimately selecting (E) 12—a value that violates $k < 6.25$.

F Limitations and Future Work

While NegoLat demonstrates consistent improvements across tasks and topologies, several design choices leave room for future extensions.

Text-Based Control Plane. NegoLat uses textual semantic probes and structured ACK/REQUEST messages for negotiation. This design improves interpretability and makes the negotiation process explicit and auditable, but introduces additional decoding overhead and restricts receiver auditing to the semantics exposed in the probe. A hybrid control plane that retains textual probes for interpretability while augmenting them with lightweight latent signals could reduce this cost.

Manually Curated Semantic Ontology. The task-family ontologies $\mathcal{T}_\tau$ for Math, VQA, and Code are manually designed. While this provides a clear and controllable negotiation interface, extending NegoLat to new domains may require additional effort to define appropriate semantic types. Automatically inducing or adapting the ontology from agent interaction traces is an important direction for future work.

Homogeneous Backbone Assumption. Our current experiments assume all agents share the same frozen backbone f_θ, which simplifies latent alignment via a shared embedding matrix. The semantic control plane itself is model-agnostic, so extending the framework to heterogeneous agents—where models differ in scale, architecture, or vendor—requires only replacing the latent bridge Φ_{SB} with a cross-model alignment module. Validating negotiation across heterogeneous backbones is a natural next step.