WHO BEARS HOW MUCH BLAME? SEMANTIC COOPERATIVE GAMES FOR BEHAVIORAL ACCOUNTABILITY AND INCENTIVE ALIGNMENT IN AGENTIC AI SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based multi-agent systems form judgments and decisions through sequences of semantic transformations. Failures in cross-provider workflows can impose financial losses, and the allocation of responsibility can shape providers' future configuration choices. Execution logs identify the agents and messages involved, but not which local semantic behaviors affected the outcome.

We propose a behavior-level accountability framework built on Semantic Cooperative Games (SCG). SCG traces backward from outcomes to recover local semantic nodes, provenance relations, generation links, and the agents that execute them. We model these links as semantic behaviors. BCAST performs counterfactual replay over behavior subsets, estimates their effects from terminal outcomes, and assigns values to behaviors and their output nodes without manual responsibility labels. These values support Behavioral Shapley and other allocation rules. Explicitly specifying the target behavior set, reference replacement, propagation rule, and outcome evaluation also places existing message-, step-, component-, and agent-level counterfactual methods in a common theoretical framework.

To connect retrospective accountability with prospective incentive design, we use Behavioral Shapley allocations as penalties and study participants' choices among semantic modes, the configurations that generate behaviors. Under stable behavioral effects and a common cancellation reference, we bound the excess expected task loss of any pure-strategy equilibrium relative to the optimum. When each participant has a mode whose advantage exceeds its variation across contexts, minimizing the proposed penalties induces a unique equilibrium that achieves the lowest expected task loss among candidate configurations.

Because objective ground-truth labels for behavioral responsibility are generally unavailable, we design two complementary empirical evaluations. The first examines semantic interventions, downstream replay, and the direction of behavioral effects in LLM workflows, assessing whether system replay can provide reliable automated evaluation signals for fine-grained behavioral attribution. The second tests whether penalty-minimizing mode choices improve task outcomes. The framework connects semantic explanations of completed executions, outcome responsibility allocation, and incentive evaluation for future deployments. Code and data are available in the anonymized repository at https://anonymous.4open.science/r/scg_bcast-2F69/.

1 INTRODUCTION

LLM multi-agent systems complete tasks through successive inputs and outputs. The output of one agent often becomes the input to others, with the final outcome emerging after multiple rounds of interaction. When an error occurs, evaluating only the correctness of the final answer makes it difficult to identify which inputs caused the problem, in which intermediate output it arose, and how subsequent interactions changed its impact. Understanding these effects is essential for locating problems, annotating failure causes, and improving systems.

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.

When agents place orders, execute transactions, or allocate resources on behalf of clients, a single incorrect decision can cause direct financial loss. Such a decision may involve agents supplied by different parties. For example, an erroneous purchase may pass through requirements analysis, planning, risk review, and order execution: the agent that ultimately submits the order performs the action, but the error may originate in an upstream judgment or be amplified through subsequent interactions. Once a loss has occurred, the client needs a clear answer: *Who is responsible, and how much should each party bear?* This requires the collaborating parties to agree on an accountability protocol in advance, providing a common basis for determining responsibility, sharing losses, and setting compensation or penalties.

The difficulty is that losses arise through interaction, whereas responsibility must be assigned to individual parties. An agent may introduce an error and later correct it; another may take effective remedial action without fully preventing the loss. Thus, participation in a failed execution does not mean that every operation contributed to the failure, and performing the final action does not make its executor solely responsible. Determining each party's share requires distinguishing the operations it actually performed in this execution and how much loss each operation caused, propagated, or mitigated. This can help developers annotate harmful operations and effective corrections, while providing task initiators and service providers with a basis for accounting for responsibility.

To this end, we build on the representation of local meanings and their generative sources in Semantic Cooperative Games (SCG) (Jiang et al., 2026) and define the introduction, preservation, and transformation of meaning that actually occur between inputs and outputs as semantic behaviors. Responsibility analysis thereby extends from tracing which inputs formed an output to evaluating which specific operations in that process increased or reduced the loss.

The semantic generation structure must also be connected to task outcomes. Supporting a particular final meaning does not directly establish how much a behavior increased or reduced the loss. Subsequent correction may remove an earlier error from the support of the final output; harm and remediation may also occur successively along the same semantic chain yet affect the outcome differently. Assigning responsibility to a behavior therefore requires a further comparison: if an operation had not occurred, or had been replaced by a specified reference operation, how would the other agents have responded, and how would the final loss have changed? When multiple behaviors are interdependent, their joint effects must also be evaluated across different combinations of background behaviors.

We propose *Behavioral Counterfactual Attribution of Semantic Transformations* (BCAST) to perform this valuation through counterfactual replay. It rests on distinguishing realized behaviors from the modes that generate them: a behavior is a semantic operation that occurred in the execution, whereas a mode comprises the settings that generate behaviors from inputs, such as the model, prompt, and tool configuration. Our basic assumption for LLM replay is that *invariant generative modes underlie changing behaviors*. When an intervention changes a preceding semantic state, the parts not directly intervened upon retain their original modes and generate new behaviors from the new inputs. BCAST accordingly applies reference operations to behaviors and dependencies identified by SCG, constructs counterfactual executions for combinations of behaviors, and evaluates their outcomes using task loss. The resulting values for behavior combinations support evaluation and allocation methods such as LOO and Shapley, linking changes in task loss to specific behaviors and their executors. The resulting scores and execution comparisons can assist responsibility annotation and review, without manually assigning a responsibility weight to each behavior in advance.

Responsibility allocation also affects how parties subsequently choose models, prompts, and tools. We further study whether penalties based on our Behavioral Shapley allocation can lead parties to reduce task loss while reducing their own penalties. We assume that the effect of a given setting on loss varies within a bounded range across inputs and use the same behavior cancellation reference when comparing different settings. Under these conditions, we bound the discrepancy between changes in individual penalties and changes in task loss, and use this bound to characterize the loss gap between a pure-strategy equilibrium configuration and an optimal configuration. When the advantage of a better setting exceeds this discrepancy, the settings that parties choose to reduce the proposed penalties jointly form the configuration with the lowest task loss among the candidates.

Our framework also unifies existing counterfactual valuation methods at the message, step, component, and agent levels by expressing their intervention targets, reference replacements, propagation rules, and outcome evaluations within a common theoretical formulation.

Our main contributions are as follows:

1. **From input–output attribution to semantic behavior accountability.** On the semantic generation structure of SCG, we treat realized operations that introduce, preserve, or transform meaning as behaviors that can be intervened upon individually, and distinguish behaviors from their generative modes. We describe analyses and interventions at the message, step, component, and agent levels using sets of behaviors, bringing the different granularities of existing methods into a common framework and further supporting responsibility evaluation for individual behaviors within groups and their combinations.
2. **Counterfactual valuation for responsibility annotation and loss accounting.** We introduce BCAST, which uses reference operations, single-chain replay, and full replay to estimate the effects of behavior combinations on task loss, supporting evaluation and allocation methods such as LOO and Shapley. The resulting responsibility scores and execution comparisons can assist annotation and review and provide a computational basis for loss sharing and penalty amounts in agreements among multiple parties, without manually assigning responsibility weights behavior by behavior.
3. **Incentive guarantees linking responsibility allocation to setting selection.** Under stable mode effects and a common cancellation reference, we establish a bounded relationship between changes in individual penalties under Behavioral Shapley allocation and changes in task loss, bound the loss gap between a pure-strategy equilibrium configuration and an optimal configuration, and prove that, when the advantage of a better setting exceeds the error, parties' choices to reduce the proposed penalties jointly achieve the lowest task loss among the candidates.

Our experiments include semantic behavior audits on public tasks and mode-selection analyses on constructed tasks, examining the effects of local operations on outcomes and the task loss of equilibrium configurations under Behavioral Shapley penalties, respectively.

2 SCG AND THE BEHAVIOR ACCOUNTABILITY PROBLEM

Following SCG (Jiang et al., 2026), we represent the behaviors to be analyzed in an executed workflow as a semantic generation hypergraph:

$$G = (S, E, \mathcal{I}; \alpha, \omega, O, I). \tag{1}$$

Here, S is the set of semantic nodes, E is the set of generation links, $\mathcal{I}$ is the set of agents, and $\alpha : E \to \mathcal{I}$ records the executor of each link; I is the set of nodes without predecessors, O is the set of output nodes, and ω gives the output weights. Node representations, source categories, and graph recovery are described in Appendix A.

Nodes represent local meanings within a state, and links record their preceding sources and outputs. A single agent action can process multiple nodes, preserving or transforming existing meanings and introducing new ones. SCG thus provides semantic behaviors that can be analyzed individually, together with their locations and executors in the workflow.

A downstream correction can eliminate the final impact of an upstream error, but this does not make the behavior that introduced the error free of responsibility. Conversely, introducing and correcting an error may jointly support the final output and receive equal contributions under a support-based allocation, despite having opposite effects on the outcome (Appendices A.4–A.5). Knowing which behaviors participated in producing the outcome is therefore insufficient; we must also compare the task outcomes obtained when these behaviors are canceled. The question is: *How can the system's own execution be used to determine outcome values for these semantic behaviors and their output nodes?*

A semantic behavior records a realized local operation together with its execution location, predecessor meanings, output meaning, and executor. Appendix H gives the full definition, the semantic identity criterion, and a test for predecessor support.

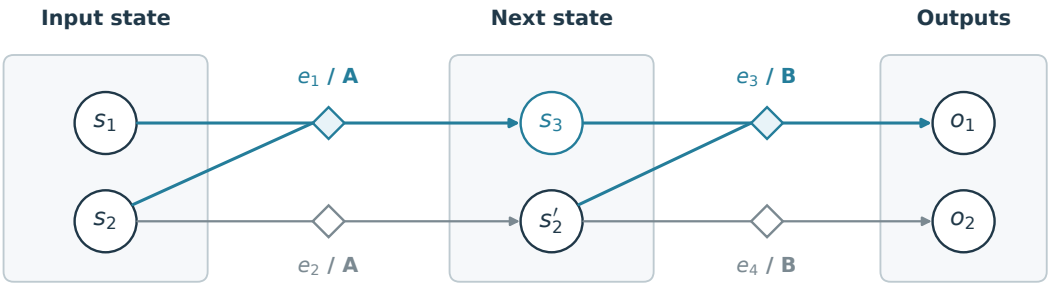


Figure 1: Local structure of an SCG. Circles denote semantic nodes, diamonds denote generation links, and the executor is indicated after each slash. A single call contains multiple operations, and several predecessors can jointly generate one node; the two outputs o_1, o_2 share part of their support. A complete message can carry several nodes in the graph.

3 BCAST: Constructing Semantic Behavior Counterfactuals

Inspired by PWSHAP's path-based analysis of a target variable's effects (Ter-Minassian et al., 2023) and the potential-outcomes literature (Rubin, 2005), BCAST asks: *What would happen if only some behaviors along a semantic chain occurred?* Comparing execution outcomes across subsets reveals the effect of each behavior with and without other operations. SCG provides the intervention locations: generation links identify specific operations, and predecessor nodes specify the semantics they receive, allowing each transformation to be changed individually.

3.1 Subsets of Semantic Behaviors

Select a set of intervenable semantic behaviors $\mathcal{B} = \{b_1, \ldots, b_m\}$, with position set $N = \{1, \ldots, m\}$. We consider all its subsets:

$$\mathcal{B}_U = \{b_i : i \in U\}, \qquad U \subseteq N. \tag{2}$$

Each U simultaneously specifies that the semantic behaviors it contains are retained and the remaining target behaviors are canceled, forming an unordered set intervention. Cancellation is implemented using a predeclared reference behavior; here, we directly pass through the current upstream semantics: existing meanings remain in their preceding state, and newly introduced meanings remain absent. Other semantic behaviors within the same call continue to execute.

Canceling an upstream semantic behavior also changes the input received downstream, so a factual behavior in the record may no longer be defined. For example, if proposition p is not introduced upstream, the original downstream operation "delete p" has no object. The corresponding semantic behavior must therefore be regenerated on the changed input.

3.2 Replaying Behaviors through Semantic Modes

An LLM can respond to different semantic inputs under the same model, role instructions, tools, and generation parameters. We call the mechanism that generates semantic behaviors under these settings a *semantic mode*. Let a_i denote the mode at position i and $M_i^{a_i}$ its conditional behavior distribution. The same mode generates behaviors on the factual input x_i and replay input x_i' as follows:

$$b_i \sim M_i^{a_i}(\cdot \mid x_i), \qquad b_i' \sim M_i^{a_i}(\cdot \mid x_i'). \tag{3}$$

During replay, b_i' replaces b_i in the subset. The expectation below also resamples behaviors at retained positions and therefore estimates the average outcome effect of the declared modes at these positions. It does not automatically equal a retrospective responsibility value obtained by fixing a particular realized behavior; the latter additionally requires a separately specified counterfactual interface that fixes the target behavior.

The reference operation need not be limited to canceling the target behavior; it may instead replace it with another behavior consistent with the specified rules, such as one generated or revised under human supervision. The resulting comparison evaluates the original behavior's outcome effect relative to that reference. These semantic modes can be replayed along a selected semantic chain, updating only the semantics on that chain while holding the remaining semantic background fixed, yielding *single-chain replay*. Alternatively, they can be replayed on the complete updated state, passing the new state to all affected downstream agents, yielding *full replay*. The former examines the intervention's effects along the selected semantic chain, whereas the latter examines its downstream effects throughout the complete workflow. The execution rules are detailed in Appendix B.

For each subset U, we perform K replays from the same entry point and estimate its outcome using the mean terminal task loss. Let X^U_{out} denote the terminal state and $\ell \in [0,1]$ the task loss. The counterfactual value and its estimator are

$$v^L(U) = \mathbb{E}\left[\ell(X^U_{\text{out}})\right], \qquad \widehat{v}^L(U) = \frac{1}{K}\sum_{k=1}^{K} \ell(X^{U,k}_{\text{out}}). \tag{4}$$

The expectation is over stochastic LLM generation, and all subsets use the same task evaluation rule. Each replay type yields its own outcome table; we use full-replay results below. Combinations that cannot be executed consistently are recorded as missing; see Appendix C.

Algorithm 1 BCAST: Generating Counterfactual Values from Behavior Subsets

Input: Execution records and SCG; audit chain $\mathcal{B}$, original modes, and cancellation rule; replay scope, task loss ℓ, and sample count K.
Output: Estimated values and replay records for each behavior subset.

1. For each $U \subseteq N$, perform K replays from the same audit entry point.
2. Provide affected agents with their actual inputs in the current replay according to workflow dependencies, and generate new behaviors b'_i under the same modes.
3. At $i \in U$, replace b_i with b'_i; at the remaining positions, cancel the audited semantic behaviors and retain the current upstream semantics.
4. In single-chain replay, propagate only semantic updates along the chain; in full replay, pass the complete response and rerun all affected downstream agents.
5. Read the terminal task loss and average according to Eq. (4); retain records of failures and missing values.

Algorithm 1 specifies the replay and valuation procedure, and Figure 2 illustrates it using a deterministic construction.

4 Behavior Evaluation and Allocation from Counterfactual Values

BCAST provides outcome values for behavior subsets. We use the *Behavioral Shapley value* (Shapley, 1953) to allocate outcome differences to individual semantic behaviors. It aggregates a behavior's marginal effects across different backgrounds: for $U \subseteq N \setminus \{i\}$, define

$$d^L_i(U) = v^L(U \cup \{i\}) - v^L(U). \tag{5}$$

Positive values indicate increased loss, and negative values indicate reduced loss. Shapley aggregates these marginal effects using the standard weights for different background subsets:

$$\psi^L_i = \sum_{U \subseteq N\setminus\{i\}} \underbrace{\frac{|U|!(m-|U|-1)!}{m!}}_{w_i(U)} d^L_i(U), \tag{6}$$

$$\sum_i \psi^L_i = v^L(N) - v^L(\varnothing). \tag{7}$$

Here, U is an unordered set of behaviors; each counterfactual simultaneously specifies which target behaviors are retained, and the set itself encodes no execution order.

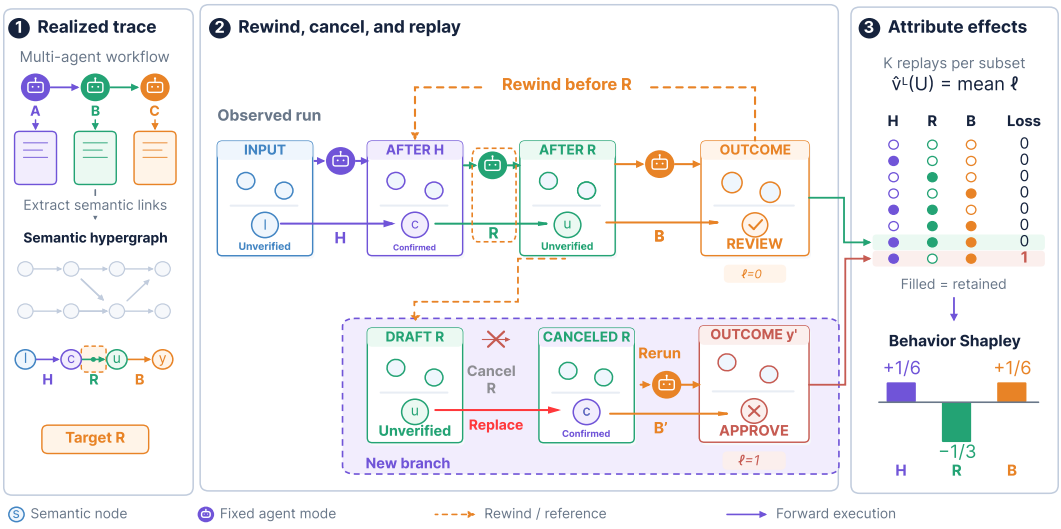


Figure 2: **Behavior localization, counterfactual replay, and responsibility allocation in BCAST.** (1) A semantic hypergraph captures local meanings and behaviors from a constructed execution record of a linear three-agent chain. H turns unverified information into a definite judgment, R restores its unverified status, and B makes a downstream decision. (2) For the retained set $U = H, B$, execution rolls back to before R. Replacing its output with its input semantic state cancels the correction, so the downstream agent again receives the judgment "confirmed." Rerunning the downstream decision in a new branch under the original generation settings yields B': Review changes to Approve, and task loss rises from 0 to 1. (3) Mean task loss estimates the outcome values used for Behavioral Shapley allocation. Here, deterministic enumeration gives exact outcomes for all eight combinations. Positive values indicate increased loss and negative values reduced loss, so the corrective behavior R receives a negative allocation.

Compared with LOO, Shapley generally requires more counterfactual samples . The framework also allows these weights to be replaced by other nonnegative coefficients summing to 1, yielding alternative weighted marginal evaluations: using only the background in which all other behaviors are retained gives LOO, while weighting all subsets equally gives the Banzhaf value (Owen, 1975). The choice can reflect the analytical objective and sampling budget.

These behavior values can also be assigned to the corresponding output nodes and then allocated through SSV along the SCG support relations, yielding node and executor evaluations without manual responsibility annotations; see Appendix D. Identification results when some combinations cannot be executed are given in Appendix C, and numerical checks on finite state spaces are given in Appendix E.

5 RELATED WORKS AS SPECIAL CASES OF OUR THEORY

Related methods can be connected to our framework along three dimensions: behavior generation mechanisms, intervention scope, and subsequent propagation. Based on the semantic generation structure of SCG (Jiang et al., 2026), we treat an introduction, preservation, or transformation of meaning that actually occurs during execution as a behavior. Several behaviors can form a set for joint evaluation or intervention: a message corresponds to all the semantic behaviors it contains, while an agent corresponds to all its behaviors across calls, rounds, and branches within the analytical scope. Analyses at the message, step, component, and agent levels thus correspond to different groupings of behaviors.

Agent and step attribution in LLM systems. Who&When (Zhang et al., 2025) introduces a failure attribution benchmark and evaluates methods for identifying responsible agents and critical steps from execution logs; AgenTracer (Zhang et al., 2026) constructs responsibility labels through replay after action repair and programmatic fault injection, then trains models for diagnosis. These meth-

ods define the scope of responsibility in terms of executors and step locations. In our framework, a step corresponds to all the semantic behaviors actually performed within it: the diagnostic methods in Who&When identify this behavior group, whereas AgenTracer's repair branch replaces the action at that step and tests whether the outcome improves through behaviors generated by subsequent modes on the new inputs. SCG further distinguishes semantic operations within a step, allowing responsibility analysis to identify specific semantic behaviors.

Message and component interventions in LLM systems. C3 (Chen et al., 2026) replaces an entire message while holding the generation policy and pre-intervention interaction history fixed, then runs subsequent agents to compare the returns of candidate messages. This corresponds to jointly replacing all semantic behaviors within a message: the candidate message provides the replacement reference, while downstream agents generate behaviors adapted to the new history under their original modes. DoVer (Ma et al., 2026) modifies orchestrator messages or plans and resumes execution to test failure hypotheses, corresponding to joint interventions at the message or plan level. RAGGY (Romero Lauro et al., 2025) allows the configurations, inputs, or outputs of retrieval or generation components to be modified, with a choice between local re-execution and downstream continuation; configuration changes correspond to changes in behavior generation modes, while output changes correspond to replacing realized behaviors. These methods treat the behaviors within a message, step, or component jointly, whereas our semantic representation allows individual behaviors within a group to be selected for evaluation.

Local error and propagation analysis in LLM systems. EDGE (Hou et al., 2026) repairs erroneous segments of a trajectory and regenerates subsequent reasoning while holding tool outputs fixed to determine whether one type of error affects another. Its method involves two independent choices: the erroneous segment determines which behaviors are modified, and fixing tool outputs restricts propagation after intervention. In our framework, the former corresponds to a set of local semantic behaviors and the latter to the replay rule.

Action attribution and effect propagation in non-LLM systems. MACIE (Weinberg, 2025) studies multi-agent reinforcement learning, replacing all actions of a target agent over a given time horizon with a reference policy and simulating the responses of other agents and the environment through a structural causal model. In our behavior representation, this is a joint intervention covering all decisions of the target agent: the target behaviors follow a common reference generation rule, while the remaining parts respond according to their specified mechanisms. ASE (Triantafyllou et al., 2024) studies multi-agent MDPs, distinguishing a source action from the subsequent actions of designated agents and measuring the source action's effect propagated through those agents by transplanting subsequent counterfactual actions. It additionally specifies which subsequent behaviors propagate the effect and which counterfactual trajectory supplies the replacement actions. These methods are grounded in environment states, actions, and causal models; using SCG, we establish semantic behavior units in language workflows that can be located and intervened upon individually.

6 Incentive Guarantees of Behavioral Allocation

Consider service providers who adjust their models, prompts, or tools to reduce their own penalties. The client is concerned with whether these adjustments reduce the loss of the entire workflow. If no participant can reduce its own penalty by unilaterally changing its choice while the others' settings remain fixed, the resulting configuration is a pure-strategy Nash equilibrium. This section examines the loss gap between such an equilibrium and the task-optimal configuration within the candidate set.

We prove that, when mode effects are stable and candidate modes share a common cancellation reference, the loss gap between an equilibrium and an optimal configuration is controlled by the variation in mode effects. If each participant's best mode has an advantage exceeding the corresponding variation, the equilibrium configuration achieves the lowest task loss within the candidate set.

Recent counterfactual replay methods (Zhang et al., 2026; Ma et al., 2026; Chen et al., 2026) continue execution under the original generation settings after modifying a local input or behavior. This practice relies on the invariance of the underlying generation mechanisms: as the execution context

changes, mechanisms not targeted by the intervention continue to respond to new inputs according to the same rules. Inspired by this principle, we regard factual and counterfactual behaviors generated under fixed settings as different realizations of the same semantic mode, and further introduce a stability assumption on its outcome effects.

6.1 Semantic Modes and a Potential Function

Along the semantic chain under analysis, participant $i \in N$ selects a mode a_i from a finite set A_i for the operation it controls, seeking to minimize its Behavioral Shapley penalty $\psi_i^L(a)$. Let $a = (a_i)_{i \in N}$ denote the participants' mode configuration, and let $L(a) = v_a^L(N)$ be the workflow's expected task loss. If the terminal evaluation records whether a task error occurs, $L(a)$ is the error probability. This section analyzes the setting in which each participant controls one behavior position and directly incurs the first-layer Behavioral Shapley penalty; SSV payoffs after full SCG redistribution and joint control of multiple positions by one participant can be further analyzed under their respective payoff mappings.

Given a mode configuration a, BCAST retains the selected modes for operations in the retained set, cancels the remaining target transformations, and completes full replay. For $U \subseteq N \setminus \{i\}$, let the expected loss difference between retaining and canceling operation i be $d_i^L(U; a) = v_a^L(U \cup \{i\}) - v_a^L(U)$. This difference measures how much the operation increases or reduces loss in the given background.

Assumption 1 (Effect stability)**.** For any mode a_i, the expected task loss difference between retaining and canceling the semantic behavior generated by that mode varies by at most $\varepsilon_i \geq 0$ across predecessor states, other participants' settings, and retention configurations.

Let $g_i(a_i)$ be the midpoint of the range of loss differences for this mode. Then $|d_i^L(U; a) - g_i(a_i)| \leq \varepsilon_i/2$. The Behavioral Shapley penalty $\psi_i^L(a) = \sum_U w_i(U) d_i^L(U; a)$ is a weighted average of these loss differences and therefore also satisfies $|\psi_i^L(a) - g_i(a_i)| \leq \varepsilon_i/2$. Thus, each participant's penalty is close to a quantity determined solely by its own mode. Summing these quantities gives

$$\Phi(a) = \sum_{i \in N} g_i(a_i). \tag{8}$$

For a unilateral change of mode, let Δ denote the value after the change minus the value before it. Because the other participants' modes remain fixed, $|\Delta\psi_i^L - \Delta\Phi| \leq \varepsilon_i$. Hence, Φ is an approximate potential function for this game: the change in a participant's penalty differs from the change in Φ by at most ε_i. Here, "approximate" refers to the discrepancy between unilateral penalty differences and potential differences (Monderer and Shapley, 1996; Candogan et al., 2013). At a minimizer of Φ, no participant can reduce its penalty by more than ε_i through a unilateral change, yielding an approximate Nash equilibrium. For pure-strategy equilibria that exist, we further bound their loss relative to the task-optimal configuration below.

We also need to relate changes in Φ to changes in task loss. For this purpose, we require candidate modes to share a common cancellation reference: with the other participants' settings a_{-i} fixed, $v_a^L(N \setminus \{i\})$ does not depend on that participant's mode a_i. This cancellation term cancels out in the comparison, giving $|\Delta L - \Delta\Phi| \leq \varepsilon_i$. Thus, Φ reflects changes in both individual penalties and workflow loss. In particular, $|\Delta\psi_i^L - \Delta L| \leq 2\varepsilon_i$: a change that reduces an individual's penalty by more than $2\varepsilon_i$ also reduces the workflow's expected task loss.

6.2 Nash Equilibria Are Close to the Task Optimum

An equilibrium configuration is close to a minimizer of the potential function, and the potential difference between configurations is close to their task loss difference. Together, these two errors yield an upper bound on the loss of an equilibrium relative to the task-optimal configuration.

Theorem 1 (Equilibria and optimal configurations under the proposed penalty)**.** *Under Assumption 1 and a common cancellation reference, let $E = \sum_i \varepsilon_i$. Every pure-strategy equilibrium a^{NE} induced by the Behavioral Shapley penalty satisfies*

$$0 \leq L(a^{\mathrm{NE}}) - \min_a L(a) \leq 2E. \tag{9}$$

When $E = 0$, the equilibrium configuration is task-optimal. Furthermore, if each participant has a mode $a_i^\star$ such that, for every $a_i \neq a_i^\star$, $g_i(a_i) - g_i(a_i^\star) > \varepsilon_i$, then $a^\star$ is the unique pure-strategy equilibrium and the unique task-optimal configuration.

E aggregates the variation in each operation's mode effects across backgrounds; the task loss gap between an equilibrium and an optimal configuration is at most $2E$. If each participant's best mode has an advantage exceeding the corresponding variation, this gap is zero.

The proof is given in Appendix J.

Finite-state numerical constructions verify the allocation identities and potential relations; full results appear in Appendix E. A three-position LLM semantic transformation experiment in a controlled procurement task uses identity cancellation and downstream replay. The Behavioral Shapley penalty induces a unique task-optimal equilibrium, and strict best-response updates from all 27 initial mode configurations converge to it; the complete setup and results appear in Appendix G.

7 Case Study

Multi-agent tasks typically allow final outcomes to be evaluated but lack fine-grained annotations of intermediate semantic behaviors: which operations introduce risk, which provide benefits, and how large their effects are. We examine whether BCAST can use existing terminal task feedback to assist in generating such behavior annotations. On 400 natural collaboration trajectories from eight public tasks, the system automatically recovers specific semantic operations and computes Behavioral Shapley through counterfactual replay, generating quantitative risk and benefit labels for 8,306 behaviors under executable backgrounds. This process requires neither predefined error types nor manually assigned responsibility scores. Strict replay and sampling-budget checks further examine the direction and stability of the generated labels. These experiments demonstrate the practical value of BCAST for behavior annotation: it converts task-level evaluations into fine-grained supervision within the process, providing quantitative labels and counterfactual evidence for semantic behaviors that are difficult to evaluate individually by hand. Full settings, results, and cases are provided in Appendix F.

8 Conclusion

We use terminal evaluations of multi-agent systems to quantify the values of specific semantic behaviors, providing a unified framework for behavior annotation, responsibility allocation, and incentive design. SCG identifies semantic behaviors and their executors from execution trajectories, BCAST measures their effects on outcomes through counterfactual replay, and Behavioral Shapley aggregates marginal effects across different backgrounds to allocate risks and benefits with both direction and magnitude. Experiments across tasks show that this procedure can assist in generating fine-grained behavior annotations without manually assigning responsibility weights to individual operations in advance. By specifying intervention targets, replacement references, propagation rules, and outcome evaluations, we also place existing counterfactual valuation methods at the message, step, component, and agent levels within a common theoretical formulation.

Quantifying behavior values further allows rewards and penalties to be attached to specific operations: behaviors that introduce risk incur corresponding penalties, while behaviors that reduce loss receive offsets or rewards. We prove that, under stable effects and a common cancellation reference, the discrepancy between changes in individual penalties and changes in system task loss is bounded, and derive a bound on the excess loss of any pure-strategy equilibrium relative to the optimal configuration. When the advantage of a better mode exceeds background variation, the settings that parties choose to reduce the proposed penalties form the unique equilibrium and achieve the lowest expected task loss among the candidates. Thus, annotating and quantifying realized behaviors provides both a basis for sharing losses among multiple parties and conditional guarantees for designing reward and penalty mechanisms that encourage more reliable collaboration.

AI USE STATEMENT

Generative AI was used to assist with structuring the manuscript and searching the literature.

REFERENCES

O. Candogan, A. Ozdaglar, and P. A. Parrilo. Dynamics in near-potential games. *Games and Economic Behavior*, 82:66–90, 2013. doi:10.1016/j.geb.2013.07.001.

W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. TabFact: A Large-scale Dataset for Table-based Fact Verification. In *Proceedings of ICLR*, 2020. Official paper and data.

Y. Chen, Y. Sun, H. Wang, J. Wang, X. Zhang, X. Shen, W. Li, and W. Zhang. Exact Is Easier: Credit Assignment for Cooperative LLM Agents. 2026. arXiv:2603.06859v2.

K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training Verifiers to Solve Math Word Problems. 2021. arXiv:2110.14168.

D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of NAACL-HLT*, Volume 1, pp. 2368–2378, 2019. Paper.

N. Guha, J. Nyarko, D. Ho, et al. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 36:44123–44279, 2023. Paper.

S. Hart and A. Mas-Colell. Potential, value, and consistency. *Econometrica*, 57(3):589–614, 1989. doi:10.2307/1911054.

J. Hou, P. Pitre, Y. Fang, and X. Wang. EDGE: Error Dependency Graph-Guided Multi-Error Attribution in Multi-Agent LLM Systems. 2026. arXiv:2609.01360v1.

P. Jiang, X. Zhu, and Q. Zhu. *Semantic Cooperative Games for Contribution Attribution in LLM-Based Multi-Agent Systems*. 2026. arXiv:2607.18255v1.

M. Ma, J. Zhang, F. Yang, Y. Kang, Q. Lin, S. Rajmohan, and D. Zhang. DoVer: Intervention-Driven Auto Debugging for LLM Multi-Agent Systems. In *Proceedings of ICLR*, pp. 28059–28087, 2026. Paper.

D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996. doi:10.1006/game.1996.0044.

G. Owen. Multilinear extensions and the Banzhaf value. *Naval Research Logistics Quarterly*, 22(4):741–750, 1975. doi:10.1002/nav.3800220409.

Q. Romero Lauro, S. Shankar, S. Zeighami, and A. Parameswaran. RAG Without the Lag: Interactive Debugging for Retrieval-Augmented Generation Pipelines. 2025. arXiv:2504.13587v1.

D. B. Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005. doi:10.1198/016214504000001880.

L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, eds., *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, 1953. doi:10.1515/9781400881970-018.

L. Ter-Minassian, O. Clivio, K. Diaz-Ordaz, R. J. Evans, and C. C. Holmes. PWSHAP: A path-wise explanation model for targeted variables. In *Proceedings of ICML*, PMLR 202:34054–34089, 2023. Paper.

S. Triantafyllou, A. Sukovic, D. Mandal, and G. Radanovic. Agent-Specific Effects: A Causal Effect Propagation Analysis in Multi-Agent MDPs. In *Proceedings of ICML*, PMLR 235:48578–48607, 2024. Paper.

H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. Paper.

D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, and H. Hajishirzi. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of EMNLP*, pp. 7534–7550, 2020. Paper.

A. I. Weinberg. MACIE: Multi-Agent Causal Intelligence Explainer for Collective Behavior Understanding. 2025. arXiv:2511.15716v1.

Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of EMNLP*, pp. 2369–2380, 2018. Paper.

G. Zhang, J. Wang, J. Chen, W. Zhou, K. Wang, and S. Yan. AgenTracer: Who is inducing failure in the LLM agentic systems? In *Proceedings of ICLR*, pp. 11377–11399, 2026. Paper.

S. Zhang, M. Yin, J. Zhang, et al. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. In *Proceedings of ICML*, PMLR 267:76583–76599, 2025. Paper.

F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua. TAT-QA: A Question Answering Benchmark on a Hybrid of Tabular and Textual Content in Finance. In *Proceedings of ACL-IJCNLP*, Volume 1, pp. 3277–3287, 2021. Paper.

A Semantic Nodes, Source Mappings, and SCG Construction

A.1 Semantic States and Semantic Nodes

Following SCG notation, a language flow is written as $T = (\mathcal{X}, H)$, where $\mathcal{X}$ is the set of complete semantic states encountered during execution and H is the set of agent actions. Each action h_u transforms an input state x_u into an output state y_u. A complete state contains the response together with relevant memory and tool information; a single action may generate, preserve, or transform multiple local meanings.

A semantic node s represents a local meaning within a state. The same meaning may be expressed in different words, so highly similar expressions are grouped into clusters for matching. For example, "record completeness has not yet been verified" and "the records cannot yet be confirmed as complete" express the same meaning, whereas "the records have been confirmed as complete" expresses a different meaning. Occurrences of the same meaning in different execution states are recorded as separate nodes to preserve the history of its transmission and processing. Each node records all carrier locations of that meaning in the text, fields, and memory for use in counterfactual replacement.

A.2 Three Semantic Types and Their Direct Sources

SCG assumes that the local semantics involved in forming the outcome can be distinguished, and that the predecessor sources of derived or persistent nodes can be identified in the input to the call that generated the node (Jiang et al., 2026). Nodes fall into three types according to their source relations:

- **Native:** first generated at the current position, with no predecessor semantic source.
- **Persistent:** the same existing meaning in the input is preserved in the output, with the corresponding earlier node $\widetilde{s}$ as its predecessor.
- **Derived:** formed by the current action from a set of existing meanings P in the input, with this set of sources as its predecessors.

The direct source mapping of an output node is

$$\mathrm{src}_{h_u}(s) = \begin{cases} \varnothing, & s \text{ is native,} \\ \{\widetilde{s}\}, & s \text{ is persistent,} \\ P, \quad P \subseteq x_u, & s \text{ is derived.} \end{cases} \tag{10}$$

Here, $P \subseteq x_u$ means that the local meanings in P all occur in the input state. Persistence requires preservation of meaning but allows changes in wording. Derivation describes how a new meaning arises from predecessor semantics, including erroneous inferences or changes from unverified content to a definite conclusion; the source classification itself does not assess whether the conclusion is correct.

Source dependence in derivation means that, without the corresponding predecessor semantics, the new meaning would not have been produced by that transformation. Equation (10) describes a single generation event; its specific sources must be determined from the actual input and records of that call.

A.3 Recursively Constructing an SCG from Output Semantics

The task evaluation interface first specifies the set of output semantics O to be explained and their weights ω. Let $h(s)$ denote the action that generated node s. Starting from O, trace direct sources backward:

$$S^{(0)} = O, \qquad S^{(t+1)} = S^{(t)} \cup \bigcup_{s \in S^{(t)}} \mathrm{src}_{h(s)}(s), \qquad S = \bigcup_{t \geq 0} S^{(t)}. \tag{11}$$

The semantic node set S is the predecessor closure of the output set. Each node is traced using the input actually received by the call that generated it. Shared predecessors within the same state are recorded once; occurrences of the same meaning in different states are connected by persistence links.

Let $I = \{s \in S : \mathrm{src}_{h(s)}(s) = \varnothing\}$ be the set of initial semantic nodes. For every $s \in S \setminus I$, construct $e_s : \mathrm{src}_{h(s)}(s) \to s$. These generation links form E, and the call records provide the executor mapping α. Writing the agent set as $\mathcal{I}$, the original SCG is

$$G = (S, E, \mathcal{I}; \alpha, \omega, O, I), \qquad \mathcal{G}(G) = (\mathcal{I}, G). \tag{12}$$

Each agent participates through the generation links it owns, and semantic support value is determined by the output nodes and their weights. We use this structure to locate behaviors and BCAST to evaluate their effects on the outcome.

Native nodes have no incoming edges in the graph. Auditing their initial introduction additionally requires recording the generating call and output carriers, then applying a reference operation that does not introduce the meaning at that position; the original graph's edge definition remains unchanged. Content outside the predecessor closure is still retained in the complete execution state.

A.4 Support Symmetry and Semantic Immunity

The following two deterministic constructions compare support allocation with outcome effects.

Symmetry in support value does not imply identical behavioral effects. The original SCG value depends on output weights and the generation support of each output. If executors A_1, A_2 are symmetric in this value, they receive equal SSV contributions, but reference replacement may produce different outcomes. Let each executor perform one behavior, so that executors and behaviors are in one-to-one correspondence.

Let $z = 0$ denote a correct judgment and $z = 1$ an incorrect judgment. Position 1 changes the judgment to 1, and position 2 restores it to 0 based on evidence; their respective modes output 1 and 0 for any current input. The factual trajectory is $0 \to 1 \to 0$, with the two behaviors causing harm and providing remediation, respectively. Select the final node as the sole output with weight 1, and record that its generation proceeds through the two operations in sequence. Its support value is then

$$V_G(C) = \mathbf{1}\{A_1, A_2 \in C\}, \qquad \mathrm{SSV}_{A_1} = \mathrm{SSV}_{A_2} = \frac{1}{2}. \tag{13}$$

This symmetry concerns support value; it does not require the two sequential links in the graph to be interchangeable.

At both positions, the reference passes through the current prior state, and the terminal loss is $\ell(z) = z$. The error reaches the terminal outcome only when position 1 is retained and position 2 uses the reference. BCAST therefore gives the outcome table and behavioral allocation

$$v^L(U) = \mathbf{1}\{1 \in U,\ 2 \notin U\}, \qquad (\psi_1^L, \psi_2^L) = (1/2, -1/2). \tag{14}$$

Relative to the prior-state reference, the two behaviors with equal support contributions respectively increase and reduce loss. The allocation signs follow from the counterfactual outcomes and require no manual assignment.

Semantic immunity can make support for an error sparse while preserving its counterfactual effect. Consider the misleading update H, review R, and action binding B in the main text. Retaining review prevents the misleading content from entering the final action; when review uses the reference, the misleading content can cause loss through binding. The corresponding outcome table is

$$v^L(U) = \mathbf{1}\{\{H, B\} \subseteq U\}\, \mathbf{1}\{R \notin U\}. \tag{15}$$

The factual execution has zero loss, and canceling H in that background also leaves the outcome unchanged. In background $\{B\}$, however, retaining H increases loss from 0 to 1. Hence,

$$\mathrm{LOO}_H^L = 0, \qquad d_H^L(\{B\}) = 1, \qquad \psi_H^L = \frac{1}{6}. \tag{16}$$

The complete allocation appears in Appendix D. Review prevents harm in the factual execution; after review is canceled, the misleading content can still affect the outcome.

Alongside this chain, add q independent, harmless output semantics generated by fixed operations that neither read the misleading semantics nor affect terminal loss.

Include the action output and the added semantics in the output set, retaining the source relations of the original chain. The action output remains the misleading node's only descendant in the output set, accounting for a fraction $1/(q+1)$. As q increases, this fraction can approach zero arbitrarily closely, while the outcome table for $\{H, R, B\}$ and Equation (16) remain unchanged. Thus, the vanishing fraction of such outputs alone does not imply that behavioral risk or an upper bound on the allocation also vanishes.

Semantics blocked by remediation. The output set O determines the coverage of the source closure. If erroneous semantics have entirely left the predecessor closure of the final output, a graph recovered only from the final response does not contain them. Applying BCAST reference replacement to that intermediate operation requires also selecting its intermediate output as a tracing target, constructing the graph by Equation (11), and retrieving the generating call and original carriers from the logs.

A.5 Self-Correction of Injected Errors

We repeatedly injected the same false conclusion into intake records, summaries, and draft answers in the workspace to examine whether subsequent collaboration would correct it. The experiment covered five cases: one constructed from claim 105 in SciFact (Wadden et al., 2020), and four manually constructed scenarios involving cash flow, legal deadlines, privacy retention, and product recalls. Each case was evaluated in six workflows: chain, parallel, star, tree, densely connected, and shared-memory. Four corrective-information presentation conditions yielded 120 six-step trajectories, with the same 30 case–workflow combinations evaluated under each condition.

In the directly visible condition, corrective information was presented as material available in the current workspace. The de-emphasized condition retained the same corrective text but described it as archived material that the downstream team had not yet opened. The remaining conditions provided only a source pointer or omitted the corrective material. Gemini 2.5 Pro rated the intensity with which the final output adopted the target false claim on a scale of $[0, 1]$, where higher values indicate stronger adoption.

Corrective material	Trajectories	Mean adoption intensity
Directly visible	30	0.100
De-emphasized (corrective text retained)	30	0.460
Source pointer only	30	0.993
Unavailable	30	0.980

When corrective evidence was directly visible, the system corrected or explicitly rejected the injected error in 23 of the 30 trajectories (76.7%). These trajectories received an adoption intensity of zero, and the mean across all 30 trajectories was 0.100. The directly visible condition also had the lowest mean within each of the six workflow types. In the product-recall scenario, for example, the input repeatedly asserted that "the recall has been resolved," yet all six workflows used the source material to reach the correct conclusion that "the recall remains open."

By comparison, mean adoption intensity was 0.460 when corrective information was de-emphasized, 0.993 when only a source pointer was provided, and 0.980 when corrective material was unavailable. This contrast illustrates semantic immunity during collaboration: visible corrective evidence enables downstream agents to identify and correct injected errors, interrupting their propagation to the final answer.

B Execution Rules for Single-Chain and Full Replay

For a given behavior subset, first generate complete responses using the original modes, restore the target prior state at reference positions, and then propagate states according to the single-chain or full-replay rules.

From complete responses to semantic behaviors. Let $K_i^{a_i}(\cdot \mid x)$ denote the distribution of complete responses under a fixed agent configuration, and let $\tau_i(x, y)$ extract the audited local behavior from the input and response. Then

$$D_i \sim K_i^{a_i}(\cdot \mid x), \qquad b_i' = \tau_i(x, D_i) \sim M_i^{a_i}(\cdot \mid x). \tag{17}$$

Extracting local behaviors from complete responses yields the semantic behavior distribution in Equation (3) of the main text. τ_i identifies operations using the target meaning and carrier interface fixed in advance. If the input does not contain the target meaning and the output does not generate it, the result is recorded as no target change. Extraction records only the operation that occurs and does not assign a responsibility score in advance.

Reference replacement. Let $\mathrm{Ref}_i(x)$ be the target prior state in the current input; for an initial introduction, it corresponds to the not-yet-generated state. At a reference position, apply

$$\mathsf{C}_i(x, D_i) = \mathrm{Replace}_i(D_i, \mathrm{Ref}_i(x)). \tag{18}$$

Reference replacement covers all target carriers declared in advance in the text, fields, and memory; the remaining content comes from the response generated on that run. Alternative semantics or a conditional reference distribution may also be specified in advance for other interventions. When the target is an entire message, the entire message is replaced.

Two propagation scopes. Fix the set of audited positions N and the actual mode configuration a. Full replay covers the audited chain and its affected descendants; single-chain replay updates only interfaces along the selected chain, with inputs from outside the chain taken from the factual record. Within a given replay method, all retention configurations start from the same entry point and execute over the same region.

Let $\mathrm{pa}(j)$ denote the parent positions of position j within this region, and let $\overline{X}_j$ be the complete output template from factual execution. Each position generates a draft from the current parent states, using the corresponding mode at audited positions and fixed processing rules elsewhere:

$$D_j^{a,U,\rho} \sim \begin{cases} K_j^{a_j}(\cdot \mid X_{\mathrm{pa}(j)}^{a,U,\rho}), & j \in N, \\ K_j^{\mathrm{bg}}(\cdot \mid X_{\mathrm{pa}(j)}^{a,U,\rho}), & j \notin N. \end{cases} \tag{19}$$

Reference replacement is then applied to target behaviors that are not retained:

$$\widetilde{D}_j^{a,U,\rho} = \begin{cases} \mathrm{Replace}_j\left(D_j^{a,U,\rho}, \mathrm{Ref}_j(X_{\mathrm{pa}(j)}^{a,U,\rho})\right), & j \in N \setminus U, \\ D_j^{a,U,\rho}, & \text{otherwise.} \end{cases} \tag{20}$$

The reference operation extracts the target prior state from the current parent states; for an initially introduced meaning, the not-yet-generated state $\perp$ may be used. The interface must specify how to extract, represent, and write $\perp$, and subsequent rules must also accept this state. Thus, "absence of this meaning" is an input permitted by the interface and is not treated as an execution failure.

Let $\mathrm{Extract}_j$ extract the target meaning, and let $\mathrm{Install}_j$ write it into a given complete template. The two propagation methods respectively install

$$X_j^{a,U,\rho} = \begin{cases} \widetilde{D}_j^{a,U,\rho}, & \rho = \text{full}, \\ \mathrm{Install}_j\left(\overline{X}_j, \mathrm{Extract}_j(\widetilde{D}_j^{a,U,\rho})\right), & \rho = \text{chain}. \end{cases} \tag{21}$$

In either propagation method, target replacement covers all declared carriers. Full replay passes on the entire new state; single-chain replay uses the factual template to fix the non-target background and passes on only the update to the target semantics. Equations (19)–(21) are applied in order along the affected descendants, including positions whose own target operations use the reference.

For example, a new review response restores an "uncertain" judgment and proposes checking the source. Full replay passes on both updates and evaluates the intervention's downstream effect through the complete workflow. Single-chain replay installs only the new judgment, retaining the source-related wording from the factual template, and evaluates propagation along the specified semantic direction under a fixed background. The two methods produce separate outcome tables.

If a step cannot extract or install the target meaning consistently with the interface, that combination lies outside the executable domain of the corresponding method. Unaffected states retain their factual content.

Executable domain and terminal values. On a finite workflow DAG, fix measurable conditional generation kernels, reference and installation rules, and a bounded terminal evaluation. For each valid combination, recursively determining the joint distribution of the positions in topological order determines a unique terminal distribution and expected value.

Syntax checks, retries, and terminal rules for failures in the execution mechanism must be fixed in advance, and their outcomes must all enter the estimate; retaining only successful samples changes the estimand. Only combinations for which consistent execution cannot be defined are recorded as structurally missing. Hybrid states produced by replay under a fixed background must also satisfy the interface constraints.

Relation to the mode-choice interface. The replay procedure above permits non-target content to come from a draft generated under the current mode. Even after the target meaning has been replaced, candidate modes may still affect the outcome through non-target content. Such counterfactual tables can be used to audit fixed modes; comparing incentives across modes additionally requires the common cancellation reference condition in Section 6 of the main text.

One implementation that satisfies this condition first generates a background $Z_i \sim G_i(\cdot \mid x)$ without consulting the candidate mode, then generates the target through $F_i^{a_i}(\cdot \mid x, Z_i)$ or the shared reference, and finally writes it into the complete state. At a canceled position, the distribution of the entire state is determined solely by (G_i, Ref_i) and is independent of the candidate mode. Every step reads the current complete input during replay.

C Identifying Allocations When Some Counterfactual Combinations Are Not Executable

Let the audited behavior set be $\mathcal{B} = \{\beta_1, \ldots, \beta_m\}$. Fix the reference, replay method, and terminal readout. Let $\mathcal{D} \subseteq 2^{\mathcal{B}}$ be the set of executable combinations, and let v be the observed outcome table. A marginal comparison for behavior β_i requires both the canceled and retained endpoints to be executable:

$$\mathcal{E}_i = \{U \subseteq \mathcal{B} \setminus \{\beta_i\} : U \in \mathcal{D},\ U \cup \{\beta_i\} \in \mathcal{D}\}. \tag{22}$$

Write $m = |\mathcal{B}|$ and $w(U) = |U|!(m-|U|-1)!/m!$. The total Shapley weight covered by executable backgrounds and the weighted mean marginal effect over those backgrounds are, respectively,

$$q_i = \sum_{U \in \mathcal{E}_i} w(U), \qquad \mu_i = \frac{1}{q_i} \sum_{U \in \mathcal{E}_i} w(U)\left[v(U \cup \{\beta_i\}) - v(U)\right], \quad q_i > 0. \tag{23}$$

When $q_i = 0$, the conditional mean is undefined; when $q_i < 1$, μ_i describes the average effect within valid backgrounds and cannot directly replace the full Shapley allocation.

Suppose the terminal readout lies in $[\ell, u]$. The set of complete tables consistent with the known outcomes is

$$\mathcal{V}(\mathcal{D}) = \left\{v' : 2^{\mathcal{B}} \to [\ell, u] : v'(U) = v(U) \text{ for all } U \in \mathcal{D}\right\}. \tag{24}$$

The full allocation to behavior i has the following sharp identified interval:

$$[\underline{\psi}_i, \overline{\psi}_i] = \left[\min_{v' \in \mathcal{V}(\mathcal{D})} \psi_i(v'),\ \max_{v' \in \mathcal{V}(\mathcal{D})} \psi_i(v')\right]. \tag{25}$$

The Shapley value is a linear function of the table entries, so the interval endpoints can be obtained by linear optimization. Without additional structural conditions, each missing entry can vary independently within the readout range; the same entry must take the same value in all marginal effects and behavioral allocations. Any cross-entry constraints imposed by the replay mechanism must also be included in the feasible set.

Each coordinate's interval endpoints are individually attainable, but different coordinates depend on shared table entries and may not attain their respective endpoints simultaneously. The Cartesian product of the coordinate intervals therefore need not equal the joint feasible allocation set.

These intervals reflect identification uncertainty caused by missing combinations. If the known entries are also estimated from finite samples, their confidence intervals can be included in the same feasible set to propagate the estimation error in the table entries.

D A Worked Allocation Example with Three Semantic Behaviors

Consider a constructed repair chain: a misleading update H rewrites an unresolved risk as a reassuring judgment, review R restores its uncertainty, and binding B applies the current judgment to the final action. Terminal loss and behavioral allocation follow the notation in the main text, with positive allocations indicating increased loss. In the sets below, H, R, B denote the corresponding behaviors.

Suppose an adverse outcome occurs only when the misleading update and action binding are retained and review uses the reference, so that $v^L(\{H, B\}) = 1$ and all other combinations have value zero:

Retained set U	$v^L(U)$	Retained set U	$v^L(U)$
$\varnothing$	0	$\{H, R\}$	0
$\{H\}$	0	$\{H, B\}$	1
$\{R\}$	0	$\{R, B\}$	0
$\{B\}$	0	$\{H, R, B\}$	0

When all behaviors are retained, review prevents the adverse outcome. LOO keeps every other behavior retained and compares only the terminal difference between retaining and canceling the target behavior:

$$\mathrm{LOO}_i^L = v^L(N) - v^L(N \setminus \{i\}). \tag{26}$$

Substituting the outcome table yields the single-behavior cancellation effects

$$(\mathrm{LOO}_H^L, \mathrm{LOO}_R^L, \mathrm{LOO}_B^L) = (0, -1, 0). \tag{27}$$

Behavioral Shapley also compares other retention backgrounds. The only nonzero marginal effect of H occurs in $\{B\}$, with weight $1/6$; the only nonzero marginal effect of R occurs in $\{H, B\}$, with weight $1/3$; and B is symmetric to H. Thus,

$$(\psi_H^L, \psi_R^L, \psi_B^L) = \left(\frac{1}{6}, -\frac{1}{3}, \frac{1}{6}\right). \tag{28}$$

Their sum is zero, consistent with the terminal difference between retaining all behaviors and using the reference at all positions. The negative allocation to review offsets the responsibility it bears.

Assign the behavioral values to their respective output nodes h, r, y. Denote the three executors by A, B, C, and let the complete-graph support functions in this construction be

$$f_h(x) = x_A, \qquad f_r(x) = x_A x_B, \qquad f_y(x) = x_A x_B x_C. \tag{29}$$

Here, executor B performs review, whereas behavior B denotes binding. Using effects on the adverse outcome as node weights, the accountability game of the original SCG is

$$V_G^{\mathrm{acc},L}(x) = \frac{1}{6}x_A - \frac{1}{3}x_A x_B + \frac{1}{6}x_A x_B x_C. \tag{30}$$

The weight of node h is allocated entirely to A, the weight of node r is shared equally by A, B, and the weight of node y is shared equally by all three. The original SSV therefore gives

$$(\mathrm{Acc}_A^L, \mathrm{Acc}_B^L, \mathrm{Acc}_C^L) = \left(\frac{1}{18}, -\frac{1}{9}, \frac{1}{18}\right). \tag{31}$$

In this example, the second-layer SSV distributes weights according to the upstream support relations of the nodes. The resulting shares therefore differ from assigning each behavioral value entirely to its executor.

If combination $\{H, B\}$ is not executable and all other known table entries are zero, then

$$(q_H, q_R, q_B) = \left(\frac{5}{6}, \frac{2}{3}, \frac{5}{6}\right), \qquad (\mu_H, \mu_R, \mu_B) = (0, 0, 0). \tag{32}$$

Writing the missing value as $t \in [0, 1]$, the set of all attainable full allocations is exactly

$$\left\{\left(\frac{t}{6}, -\frac{t}{3}, \frac{t}{6}\right) : 0 \le t \le 1\right\}. \tag{33}$$

The three coordinate intervals are $[0, 1/6]$, $[-1/3, 0]$, and $[0, 1/6]$, but all are determined by the same t and cannot vary independently. Although the conditional means over executable backgrounds are all zero, behavioral effects in opposite directions may still occur in unexecuted backgrounds.

E Numerical Verification with Finite-State Constructions

We use finite counterfactual tables to test allocation identities and compute the largest loss gap between a designated equilibrium and another configuration. The constraints here compare different retained sets only *within a fixed mode configuration*; the stability condition in Section 6 additionally compares effects after other participants change their modes. This construction therefore cannot be used to assess the tightness of the error bound in the main text.

E.1 Effect Variation within a Fixed Configuration

Each position has two candidate modes. We use -1 for the reference and $0, 1$ for retained modes, with each state in $\{-1, 0, 1\}^m$ corresponding to a terminal loss probability. All values are restricted to $[0, 1]$, and the all-reference value is fixed at 0.8. The state index records only the modes at retained positions, so this construction satisfies cancellation independence.

Let η denote the bound on within-configuration variation in this section. For all 2^m candidate configurations, every position, and every pair of retained sets, impose $d_i^L(U; a) - d_i^L(V; a) \leq \eta$. We further constrain the all-zero configuration a^0 to be a pure equilibrium and maximize its loss gap relative to the all-one configuration a^1, namely $L(a^0) - L(a^1)$. Two verification scripts yield identical results (Table 1), and the supplementary script also saves the complete counterfactual tables attaining the objective values.

Positions m	Within-configuration bound η	Outcome variables	Maximum objective	$m\eta$
2	0.02	9	0.04	0.04
3	0.02	27	0.06	0.06
4	0.02	81	0.08	0.08
4	0	81	0	0

Table 1: Optimization results for finite counterfactual tables. The objective is the task loss gap between the designated equilibrium and the comparison configuration. All values are computed from finite-state constructions.

These counterfactual tables can be implemented by a finite-state process: each position writes a mode or reference marker into an independent slot, and the terminal step outputs the corresponding loss probability based on the complete slot state. When the variation bound is positive, the designated equilibrium can have greater loss than another configuration; when the bound is zero in the table, the maximum objective value is also zero.

E.2 Allocation Identities and Reproducibility

The program also separately maximizes the penalty variation of a fixed mode across configurations; the first three settings in the table all yield 0.02, and the final setting yields zero. This construction satisfies cancellation independence and admits the classical Shapley potential construction (Hart and Mas-Colell, 1989). Enumerating all unilateral mode changes gives a maximum residual of zero for the exact potential relation and a maximum residual below 5×10^{-17} for allocation efficiency.

F BEHAVIORAL SHAPLEY ANNOTATION FROM TERMINAL FEEDBACK

This experiment addresses a direct question: without behavior-level responsibility labels, can BCAST recover semantic behaviors that occurred in real multi-agent trajectories and use terminal task loss to generate auditable risk or benefit annotations for those behaviors? We do not inject artificial errors, prescribe error types in advance, or provide gold-standard answers to either the workflow agents or the semantic analyzer.

F.1 TASKS, DATA, AND A NONLINEAR WORKFLOW

We freeze 50 cases from each of eight public tasks, yielding 400 natural collaboration trajectories. For each task, the first 30 trajectories form the discovery set and the remaining 20 form the frozen validation set; cases are not selected on the basis of terminal correctness, semantic graph size, or attribution results. The tasks cover multi-hop question answering, discrete and numerical reasoning, table reasoning, fact verification, scientific evidence assessment, and the application of legal rules. Table 2 lists the tasks and their offline terminal evaluation metrics.

Table 2: Frozen experimental data from eight public tasks. Factual accuracy is computed offline from the final outputs and hidden gold-standard answers.

Dataset	Task	Terminal evaluation	Factual accuracy
HotpotQA (Yang et al., 2018)	Multi-document bridge and comparison question answering	Answer token F1	0.66
MuSiQue (Trivedi et al., 2022)	Compositional question answering over two to four hops	Answer token F1	0.52
DROP (Dua et al., 2019)	Discrete and numerical reasoning over text	Answer token F1	0.64
GSM8K (Cobbe et al., 2021)	Mathematical word problems	Numerical accuracy	0.96
TAT-QA (Zhu et al., 2021)	Reasoning over financial tables and paragraphs	Answer F1 and scale	0.52
TabFact (Chen et al., 2020)	Table-based fact verification	Binary classification accuracy	0.92
SciFact (Wadden et al., 2020)	Scientific claim verification	Three-class accuracy	0.84
LegalBench-hearsay (Guha et al., 2023)	Application of evidentiary rules	Binary classification accuracy	0.70

All cases use the same six-agent directed acyclic workflow (Figure 3). A, B, and C run in parallel: A only decomposes the task and does not read the sources, B constructs evidence records from the sources, and C reads the sources and solves the task independently. D reads A+B and assembles a complete derivation without source access; E reads A+C, retains source access, and performs recalculation, counterargument, and alternative interpretation; F reconciles D+E without source access and produces the terminal output. D and E run in parallel, so the workflow includes independent evidence paths, cross-support, and semantic transformations with multiple inputs. Each task also has dedicated responsibilities: TAT-QA separates value extraction, formulas, denominators, and scale; SciFact checks populations, direction, and evidentiary scope; and LegalBench separates the statement, out-of-court, and truth-purpose requirements.

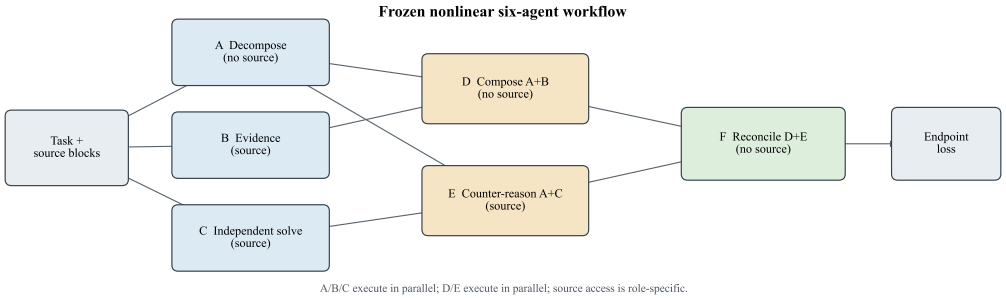


Figure 3: The frozen nonlinear six-agent workflow used in the experiment. All labels in the figure are in English; source indicates that the role can read the original sources.

The workflow model is Gemini 2.5 Flash-Lite with a generation temperature of 0.35. Semantic analysis, identity assessment, and local editing use Gemini 2.5 Flash at temperature 0, with Gemini 2.5 Pro used for review when editing fails. Gold-standard answers are supplied only to the deterministic terminal scorer. Factual trajectories, semantic graphs, counterfactual terminal outputs, and scores are all frozen before analysis.

F.2 From Semantic Graphs to Instance-Specific Behavior Players

The analyzer recursively traces backward from terminal value semantics to recover all predecessors of each node, allowing many-to-one transformations and merging equivalent semantics within the same trajectory. Each agent-owned link is classified into one of three types. NEW denotes the first introduction of a substantive meaning at the current step, including information first extracted from the question or sources. TRANSFORMED denotes a different meaning produced from one or more predecessor semantics through calculation, combination, judgment, revision, negation, or specification. PRESERVED denotes continuity of an existing meaning without semantic change. After node merging, PRESERVED is retained only as fixed continuity: it cannot be canceled, receives no behavior score, and does not enter the player set.

For a particular trajectory τ, the player set of the first-layer behavior game is

$$N_\tau = \{b : \mathrm{type}_\tau(b) \in \{\textsc{New}, \textsc{Transformed}\}\}. \tag{34}$$

Thus, the 400 trajectories define 400 independent behavior games. The total number of behaviors across tasks describes only the data scale; these behaviors are never pooled into a single global game. The system recovers 5,801 NEW and 3,600 TRANSFORMED players and merges 5,465 PRESERVED continuity relations. Table 3 reports the scale for each task.

Table 3: Semantic graphs, behavior players, and counterfactual validation by task. $\widehat{\psi}$ denotes players with an executable-domain Behavioral Shapley estimate; PRES. denotes fixed continuity relations.

Task	Games	NEW	TRANS.	PRES.	Estimable $\widehat{\psi}$	Strict replays	Nonzero replays
HotpotQA	50	520	344	630	761	137	16
MuSiQue	50	794	371	732	1,064	118	19
DROP	50	572	395	577	857	132	17
GSM8K	50	929	569	792	1,274	158	2
TAT-QA	50	691	306	740	918	133	23
TabFact	50	829	530	705	1,216	158	12
SciFact	50	825	435	612	1,121	183	9
LegalBench	50	641	650	677	1,095	162	23
Total	400	5,801	3,600	5,465	8,306	1,181	121

F.3 Behavioral Shapley and Counterfactual Replay

Let P_τ denote the merged fixed continuity relations in trajectory τ, and let $U \subseteq N_\tau$ be the set of retained factual behaviors. U is unordered: the workflow still executes according to its causal DAG, and random permutations are used only to sample Shapley weights, not to define a treatment order. For a target behavior $b \in N_\tau$, the system performs a paired comparison under the same background $U \subseteq N_\tau \setminus \{b\}$:

$$\Delta^L_{\tau,b}(U) = L_\tau(U \cup \{b\}; P_\tau \equiv 1) - L_\tau(U; P_\tau \equiv 1). \tag{35}$$

The retention side installs the same specific behavior from the factual trajectory, rather than resampling a different output under the same mode. Canceling NEW keeps that meaning from being introduced; canceling TRANSFORMED passes through its predecessor semantics while suppressing the target transformation. Other affected positions replay on the new inputs under the frozen generation settings, with both sides sharing the same random seed. A positive value indicates that the behavior increases terminal loss; a negative value indicates that it reduces terminal loss.

Over the complete executable domain, Behavioral Shapley is

$$\psi^L_{\tau,b} = \sum_{U \subseteq N_\tau \setminus \{b\}} \frac{|U|!(|N_\tau| - |U| - 1)!}{|N_\tau|!} \Delta^L_{\tau,b}(U). \tag{36}$$

A factual behavior has a specific semantic identity. If a background has removed all predecessor semantics required by the behavior, the system records the comparison as identity-inapplicable; it neither substitutes a newly generated behavior for the target nor records a zero contribution. Let $\mathcal{D}_{\tau,b}$ be the backgrounds in which that identity remains executable. The experiment normalizes the standard Shapley weights over this domain:

$$\psi^{L,\text{exec}}_{\tau,b} = \frac{1}{Z_{\tau,b}} \sum_{U \in \mathcal{D}_{\tau,b}} \frac{|U|!(|N_\tau| - |U| - 1)!}{|N_\tau|!} \Delta^L_{\tau,b}(U), \quad Z_{\tau,b} = \sum_{U \in \mathcal{D}_{\tau,b}} w_\tau(U). \tag{37}$$

The main scan uses one pair of complementary random permutations for each player, producing at most two Shapley-weighted backgrounds. Of the 9,401 players, 5,699 complete valid replay in both backgrounds and 2,607 in one background, yielding $\widehat{\psi}^{L,\text{exec}}_{\tau,b}$ for 8,306 players. The remaining 1,095 have no successful, identity-applicable paired result. All instances of inapplicability, model-call failure, and editing failure are retained as explicit statuses and are not replaced with zero.

F.4 Validation by Strict Replay

Behavioral Shapley values are weighted marginals across multiple backgrounds. To test whether their risk/benefit labels correspond to directly auditable terminal changes, we extract 1,181 strict comparisons from the saved replays. The target must be NEW or TRANSFORMED, all PRESERVED continuity relations must be fixed, the target identity must be applicable, and both the retention and cancellation sides must succeed. Selection uses only execution status, without inspecting the sign or magnitude of the marginal. These strict comparisons cover 355 trajectories: 1,060 have zero terminal effect, 71 reduce loss, and 50 increase loss.

Table 4 compares the signs of Behavioral Shapley and strict replay. Across all 1,181 records, three-class sign agreement is 93.6%, Spearman $\rho = 0.729$, and the mean absolute difference is 0.057. Among the 121 nonzero strict effects, 113 agree in direction, 3 are estimated as zero by Shapley, and 5 have the opposite direction. Risk/benefit direction agreement is 93.4% (Wilson 95% CI $[87.5, 96.6]$), with correlation $\rho = 0.930$ ($p < 10^{-52}$). Nonzero direction agreement is 97.8% in the frozen validation set, compared with 90.8% in the discovery set. Across tasks, nonzero direction agreement ranges from 82.6% to 100% (Figure 4a).

The second validation examines whether automatically identified loss-increasing behaviors are enriched in failed factual trajectories. We define a trajectory-level risk flag as the presence of at least one strict replay with $\Delta L > 0$. Among trajectories with strict comparisons, the flag occurs in 27.7% of failed factual trajectories and 3.9% of correct factual trajectories, corresponding to OR $= 9.36$ (95% CI $[4.34, 20.17]$). In the frozen validation set, the respective rates are 29.5% and 3.1%, with OR $= 13.00$ (95% CI $[3.47, 48.64]$). A within-task permutation test with 100,000 permutations

Table 4: Agreement between Behavioral Shapley labels and strict counterfactual replay. Direction agreement requires an exact match among the three signs: risk, zero effect, and benefit.

Scope	Records	Direction agreement	Spearman ρ	MAE
Discovery set, all strict replays	685	93.3%	0.702	0.058
Frozen validation set, all strict replays	496	94.0%	0.773	0.056
All strict replays	1,181	93.6%	0.729	0.057
Nonzero strict replays	121	93.4%	0.930	0.285

yields $p \leq 10^{-5}$, and the Mantel–Haenszel common OR stratified by task is 7.49. Risky operations in correct factual trajectories are not contradictory: during actual execution, a local loss may be offset by another branch or a subsequent coordination behavior. Figure 4b summarizes this result.

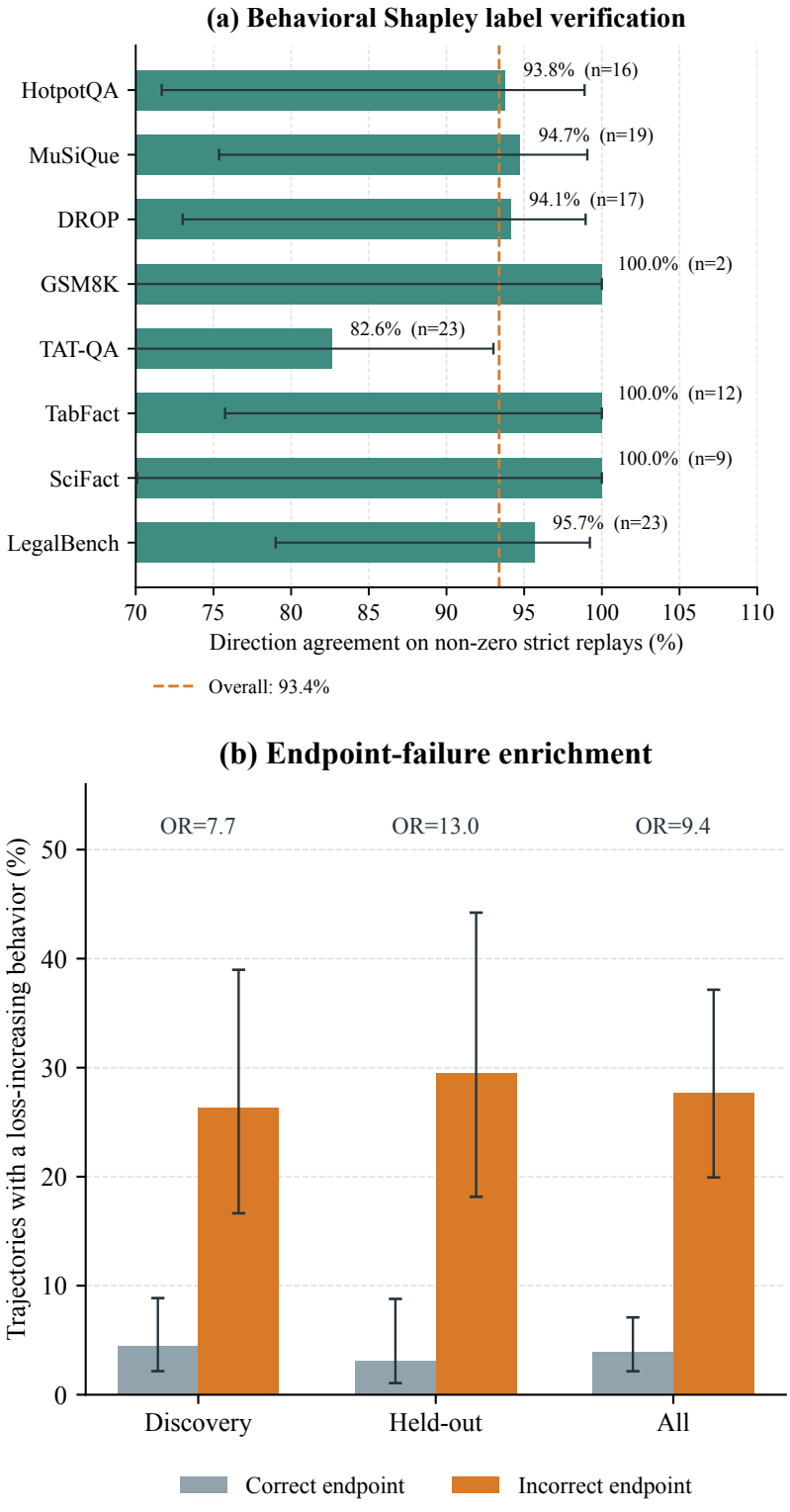


Figure 4: Cross-task validation of Behavioral Shapley annotations. All text in the figure is in English. (a) Agreement between nonzero strict replays and Shapley risk/benefit directions for each task, with Wilson 95% intervals; (b) prevalence of loss-increasing behaviors in correct and failed factual trajectories.

F.5 Effects across Tasks and Workflow Positions

All eight tasks contain both loss-reducing and loss-increasing behaviors, indicating that terminal supervision does not reduce to labels for a single task or output field (Figure 5). GSM8K has a factual accuracy of 0.96 and thus has fewer nonzero behaviors in strict replay. Propagation of numerical units in TAT-QA and composition of rule requirements in LegalBench produce more observable terminal changes.

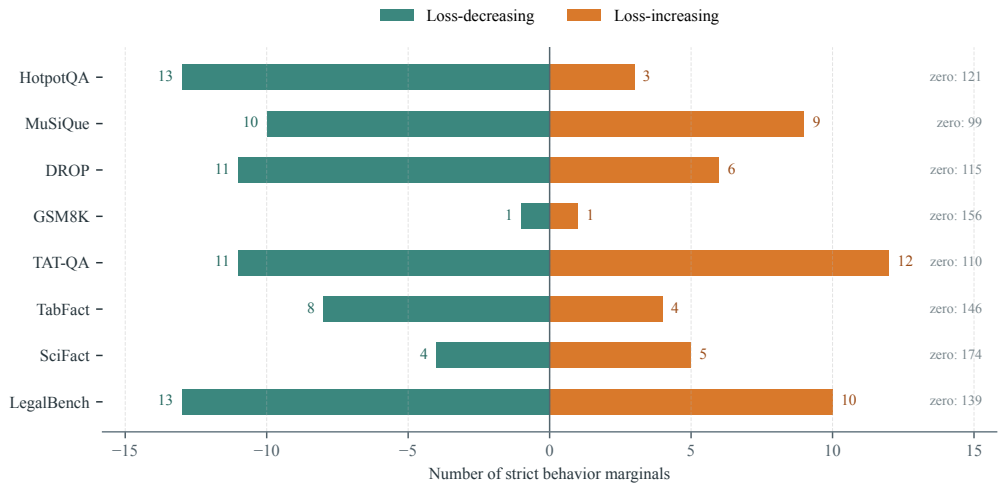


Figure 5: Counts of strict counterfactual effects across the eight tasks. Orange denotes increased loss, green denotes reduced loss, and the counts on the right indicate zero effects. All text in the figure is in English.

The strict nonzero rates for New and Transformed are 10.87% and 9.02%, respectively. Decomposition by workflow role shows both positive and negative effects in upstream decomposition, evidence analysis, independent solving, intermediate synthesis, counterargument, and final coordination (Figure 6). The labels therefore concern specific semantic operations and their outcome effects in the current background, rather than assigning responsibility in advance on the basis of agent position.

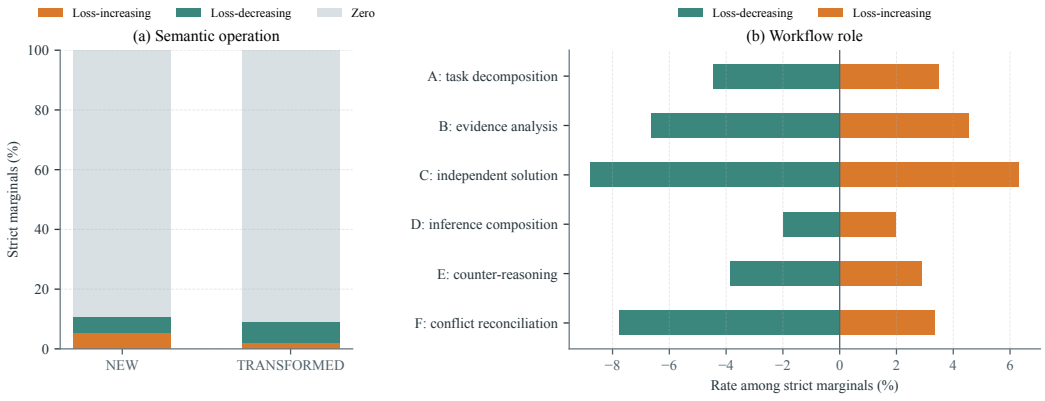


Figure 6: Distribution of strict counterfactual effects by semantic operation type and workflow role. All text in the figure is in English.

F.6 Empirical Characteristics of Risky Behaviors

A "risky behavior" has a direct operational definition here: with all PRESERVED continuity relations fixed, retaining a specific NEW or TRANSFORMED operation increases terminal loss, so its strict replay satisfies $\Delta L > 0$. Of the 1,181 strict comparisons, 50 satisfy this condition: 41 are NEW and 9 are TRANSFORMED. Of these, 46 also receive positive Behavioral Shapley values, 2 are estimated as zero in the sampled backgrounds, and 2 have the opposite sign. Among only the 121 nonzero strict effects, loss-increasing behaviors account for 48.2% of NEW and 25.0% of TRANSFORMED behaviors. A trajectory-level bootstrap estimates the latter minus the former at -23.7 percentage points (95% CI $[-40.2, -4.9]$). This indicates that the most prominent point of risk entry in this experiment is **state selection**: the initial choice of which entities, values, time ranges, subgoals, or rule requirements enter subsequent reasoning distinguishes risk more clearly than simply occupying a later workflow position.

Table 5 breaks down the 50 loss-increasing behaviors by task. Although the surface errors differ across tasks, four recurring semantic misalignments are visible: correct or approximately correct facts are bound to the wrong time, unit, answer type, or comparison target; bridge entities and next-hop subgoals drift; multiple premises are compressed into a stronger conclusion unsupported by the evidence; and verification operations rewrite an otherwise decidable outcome as `UNRESOLVED`. These descriptions are readable summaries formed after the counterfactual effects have been determined; they do not enter the calculation of Behavioral Shapley or strict marginals.

Table 5: High-risk semantic operations observed across the eight tasks. The final column reports the number of loss-increasing strict replays divided by all strict replays for that task.

Task	Recurring risky behaviors	Representative terminal changes	Risk/strict
HotpotQA	Selecting bridge facts that direct subsequent reasoning to the wrong year or entity; rewriting an upstream consensus as uncertainty	1942 → 1941; `No`→`UNRESOLVED`	3/137
MuSiQue	Setting the wrong next-hop subgoal or entity type; coarsening precise temporal or spatial relations into vague answers	1981 →*early 1980s*; correct entity→`UNRESOLVED`	9/118
DROP	Extracting facts adjacent to the question but with the wrong time range or comparison target; replacing the requested quantity with distracting facts	1979 → 1972; 5 → 6; correct answer→`UNRESOLVED`	6/132
GSM8K	Omitting a participating entity from the summation target, leaving subsequent calculations internally consistent over an incomplete set	298 → 237	1/158
TAT-QA	Rescaling table values that already carry a scale; transferring currency units to count answers; overriding a directly answerable definition with a claim of "missing details"	365,607 [thousand]→ 365,607,000 [million]; 2 → 2 [thousand]	12/133
TabFact	Abandoning a decidable label because of a mismatch in table granularity; incorrectly decomposing a proposition or binding table counts	`REFUTED`→`UNRESOLVED`; `ENTAILED`→`REFUTED`	4/158
SciFact	Losing population, direction, or evidentiary scope when extracting a claim or study finding; deriving a stronger comparative conclusion from related findings	`NEI`→`CONTRADICT`; `NEI`→`SUPPORT`	5/183
LegalBench	Misidentifying the purpose of a statement or its out-of-court status; incorrectly joining conjunctive hearsay requirements into a disjunctive rule	`Yes`↔`No`	10/162

Agent position also cannot serve as a proxy for risk. C, B, and A own 18, 13, and 11 loss-increasing behaviors, respectively, whereas the final coordinator F owns only 3 loss-increasing behaviors and also 7 loss-reducing behaviors. Another important observation is that a local semantic statement can itself be true yet activate an incorrect downstream path in the current background. In HotpotQA, correctly extracting USAAF is factually accurate but still shifts the terminal answer from 1942 to 1941. Behavioral Shapley labels the outcome effect of that specific operation in the current task game, rather than assessing the truth of a sentence independently of context.

F.7 Eight Auditable Cases

Table 6 selects one loss-increasing behavior from each task. Each row shows the target behavior, terminal outputs under cancellation and retention, the strict marginal, and the behavior's Behavioral

Shapley value in its own task instance. All eight targets are recovered by the system from free-text trajectories, without manually supplied behavior labels. These cases expose the semantic evidence behind the computed results for direct inspection.

Table 6: Specific risky behaviors in eight tasks for which Behavioral Shapley agrees with the strict counterfactual terminal outcome. The left side of each arrow cancels the target behavior; the right side retains the factual behavior.

Task	Behavior/owner	Type	Specific semantic operation	Cancellation → retention	$\widehat{\psi}^L$	ΔL
HotpotQA	B013 / C	NEW	Extracting a USAAF bridge fact and activating an incorrect year path	1942 → 1941	0.50	1.00
MuSiQue	B026 / D	TRANS.	Coarsening the precise inference of 1981 to early 1980s	1981 → early 1980s	0.50	1.00
DROP	B001 / A	NEW	Rewriting the question target as the wrong comparison of annual membership changes	1979 → 1972	1.00	1.00
GSM8K	B010 / B	NEW	Omitting Cookie Monster from the summation set	298 → 237	0.50	1.00
TAT-QA	B019 / D	TRANS.	Multiplying a table value already labeled thousand by 1,000 again	365,607 [thousand] → 365,607,000 [million]	1.00	1.00
TabFact	B016 / B	TRANS.	Abandoning a decidable conclusion because of a "table granularity mismatch"	REFUTED → UNRESOLVED	1.00	1.00
SciFact	B030 / E	TRANS.	Inferring faster proliferation at a higher dose from saturation of a limiting effect	NEI → CONTRADICT	0.50	1.00
LegalBench	B005 / A	TRANS.	Incorrectly joining two hearsay requirements into a disjunctive rule	Yes → No	1.00	1.00

TAT-QA: numerical and dimensional transformation. D combines "gross profit in 2018 was 365,607" with "the table unit is thousand," multiplies by 1,000 again, and binds the result to million. Canceling this specific transformation restores the correct terminal output of 365,607 [thousand]; retaining it yields 365,607,000 [million]. The behavior has $\widehat{\psi}^L = 1.0$ and a strict marginal of $+1$.

SciFact: comparative inference. From "the effect of innate immunity in limiting growth is weaker at a higher inoculation dose," E infers "proliferation is faster at a higher dose." In a background where the predecessor study findings remain visible, canceling this inference yields the terminal output `NOT_ENOUGH_INFO`, whereas retaining it yields `CONTRADICT`. The average over the two Shapley-weighted backgrounds is 0.5, while the strict marginal with all continuity relations fixed is $+1$.

LegalBench: composition of rule requirements. A combines the requirements "whether an out-of-court statement exists" and "whether it is offered to prove the truth of its content" into the decision rule "if either condition is not met, it is not hearsay." With both predecessor requirements retained, canceling this composition behavior yields the correct terminal output `Yes`, whereas retaining it yields `No`. Both $\widehat{\psi}^L$ and the strict marginal are $+1$.

Bridging and target drift in multi-hop question answering. In MuSiQue, D transforms the precise source year 1981 into "early 1980s," changing an exact-match answer into an incorrect one; the behavior has $\widehat{\psi}^L = 0.5$. In HotpotQA, the USAAF bridge fact extracted by C is itself correct, but under the background of the task's other behaviors it activates an incorrect continuation to a different year, changing the terminal answer from 1942 to 1941, with $\widehat{\psi}^L = 0.5$. The latter case illustrates that a risk label concerns the outcome effect in context and does not require the local proposition itself to be false.

Target-set errors in DROP and GSM8K. In DROP, A rewrites the question's comparison target, changing the answer from 1979 to 1972, with $\widehat{\psi}^L = 1.0$. The same task also contains a scope error that replaces "before this loss" with "the record after the loss," changing the answer from 5 to 6. In GSM8K, B limits the total to Katarina, Max, and Summer, omitting Cookie Monster and changing the terminal answer from 298 to 237, with $\widehat{\psi}^L = 0.5$. Neither case is a failure of the arithmetic step itself; both select the wrong target scope before calculation.

TabFact: unsupported abstention. B interprets "the question is described in terms of games, whereas the table lists positions" as making verification impossible, changing a terminal outcome that could have been labeled `REFUTED` to `UNRESOLVED`. This transformation has $\widehat{\psi}^{L} = 1.0$, indicating that a verifier's introduction of unnecessary uncertainty can itself constitute a high-risk semantic behavior.

F.8 Stability and Integrity

On frozen validation trajectories selected by a fixed hash, we increase the reverse-sampling budget for strict replay from one pair to four pairs. Both budgets cover 54 behaviors, of which 10 are nonzero. The two sets of estimates have Spearman $\rho = 0.999$, a mean absolute difference of 0.0278, and 100% direction agreement on nonzero behaviors. This check changes only the sampling budget, leaving the factual trajectories, semantic graphs, player definition, and terminal scorer unchanged.

All 400 trajectories produce semantic graphs that pass structural checks. The released data separately retain factual trajectories, backward-recovered nodes and hyperedges, player types, fixed continuity relations, sampled backgrounds, identity applicability, retention and cancellation terminal outputs, terminal losses, and Behavioral Shapley values. Humans need not annotate responsibility for every behavior; they can audit the system's local assessments using the saved semantic carriers, predecessor nodes, and paired terminal outputs. With 8,306 estimable behavior values, 93.4% nonzero direction agreement, 97.8% agreement in the frozen validation set, and failure enrichment of OR $= 9.36$, the experiment indicates that BCAST can produce specific, inspectable behavior-level attribution annotations that are stable across tasks from natural collaboration trajectories and terminal feedback.

G INCENTIVES FOR LLM SEMANTIC TRANSFORMATION MODES

G.1 EXPERIMENTAL QUESTION AND SEMANTIC WORKFLOW

This experiment tests whether minimizing a Behavioral Shapley loss penalty can induce a task-optimal mode configuration when participants select generation modes for the semantic transformations they control. We construct three semantic transformation positions in a procurement decision task with a determinate answer, forming the causal workflow $A \to B \to C$. A transforms the source materials into fact nodes, B transforms the current facts into numerical and rule-based conclusions, and C transforms the current analysis into a final supplier decision. Each position is controlled by one participant, who chooses among three modes: Careful, Shortcut, and Distorter.

The procurement materials specify that 120 items are ordered from supplier A at a unit price of 8, with 5 damaged items; 95 items are ordered from supplier B at a unit price of 10, with 2 damaged items; and the total budget is 2,000. A supplier is acceptable only if its damage rate does not exceed 5%, and the acceptable supplier with the lower cost per usable item should be selected. The correct derivation yields usable quantities of 115 and 93, damage rates of approximately 0.0417 and 0.0211, costs per usable item of approximately 8.35 and 10.22, and a total procurement cost of 1,910, with supplier A as the final recommendation.

The visible state and write permissions at the three positions are fixed as follows.

- **A: fact extraction.** Reads the original procurement materials and writes only order quantities, unit prices, damaged quantities, the budget, and the acceptance threshold; it does not calculate derived quantities or recommend a supplier.
- **B: calculation and rule-based derivation.** Reads only the current fact nodes and generates usable quantities, damage rates, acceptability, costs per usable item, total cost, budget status, and candidate suppliers. When required facts are missing, the corresponding outputs are `UNKNOWN`.
- **C: decision synthesis.** Reads the current facts and analysis and generates a recommendation from the analysis. It may not re-extract sources or repeat the arithmetic, and outputs `INSUFFICIENT` when the required analysis is missing. Predecessor quantities that it presents again receive no additional score.

All three positions and modes use Gemini 2.5 Pro at temperature 0.2, with a maximum output of 900 tokens per call. Mode differences are controlled through prompts specifying generation rules, and all modes comply with the visibility and write permissions of their positions (Table 7).

Table 7: Three semantic transformation modes implemented with the same model.

Mode	Generation rule
Careful	Fully performs and checks the transformation at its position. B uses the ordered quantity to calculate procurement cost and the usable quantity as the denominator for unit cost; missing information remains explicitly unknown.
Shortcut	Applies fixed shallow rules: A records 5% as the numerical value 5 and skips the budget; B substitutes the listed price for cost per usable item, expresses the damage rate as an integer percentage, and selects the supplier with the lower damage rate; C chooses the supplier with the lower damage rate when the analysis is complete.
Distorter	Produces coherent local transformations with systematic errors, such as swapping or perturbing key facts, using an incorrect denominator, reversing rule judgments, or choosing the wrong supplier. B and C still output unknown when required predecessor information is unavailable.

G.2 IDENTITY CANCELLATION, DOWNSTREAM REPLAY, AND SCORING

Let x_i be the complete semantic state entering position i, and let $T_i^{a_i}$ be the generative transformation under the selected mode. For a mode configuration a and a set of retained behavior positions $U \subseteq N = \{A, B, C\}$, the system executes from the same initial state:

$$x_{i+1} = \begin{cases} T_i^{a_i}(x_i; \xi_i), & i \in U, \\ x_i, & i \notin U. \end{cases} \tag{38}$$

Cancellation uses the identity map, preserving previously produced predecessor nodes unchanged; subsequent positions read the resulting counterfactual state and regenerate their outputs. The set U simultaneously specifies which positions are retained, while the workflow executes according to its causal dependencies. Identical visible inputs, positions, modes, and random seeds strictly reuse the same transformation, so predecessor outputs are unaffected by whether future positions are retained.

The terminal scorer reads 20 unique semantic nodes in the complete state: 8 fact nodes, 11 derived nodes, and 1 final decision node. Each node is scored independently, and loss is the proportion of incorrect or missing nodes:

$$\ell(x_{\text{out}}) = 1 - \frac{1}{20}\sum_{j=1}^{20} \mathbf{1}\{\text{node } j \text{ matches its reference value}\}. \tag{39}$$

Scoring consistently normalizes synonymous labels and permits numerical errors within 0.5% of the reference value. Fact and derived nodes are each scored once in their respective states; C need not copy these nodes again to preserve their value.

The experiment exhaustively enumerates $3^3 = 27$ mode configurations and $2^3 = 8$ behavior subsets, using 3 random seeds for each configuration–subset pair and saving $27 \times 8 \times 3 = 648$ counterfactual replay records. Averaging the losses over the three replays yields the empirical characteristic function $\widehat{v}_a^L(U)$, from which Behavioral Shapley is calculated exactly over all subsets:

$$\widehat{\psi}_i^L(a) = \sum_{U \subseteq N\setminus\{i\}} \frac{|U|!(|N| - |U| - 1)!}{|N|!} \left[\widehat{v}_a^L(U \cup \{i\}) - \widehat{v}_a^L(U)\right]. \tag{40}$$

Each participant minimizes their own $\widehat{\psi}_i^L$ by unilaterally changing modes. A negative value indicates that the transformation at that position reduces terminal loss.

G.3 Semantic Checks of Cancellation Replay

Table 8 reports the losses for the eight subsets under the all-Careful configuration. When only A is retained, the 8 fact nodes are already correct, yielding a loss of $12/20 = 0.600$. Retaining A and B produces all 19 fact and derived nodes correctly, reducing loss to $1/20 = 0.050$. Full execution produces all 20 nodes correctly, with a loss of 0. After B is canceled, A's fact nodes remain, but C cannot substitute for the missing analysis under its assigned responsibilities; the loss for AC therefore remains 0.600.

Table 8: Identity cancellation under the all-Careful configuration. Each cell is the mean terminal loss over three replays, all of which produce the same result.

Retained set	$\varnothing$	A	B	C	AB	AC	BC	ABC
Terminal loss	1.000	0.600	1.000	1.000	0.050	0.600	1.000	0.000

The Behavioral Shapley losses for this configuration are

$$(\widehat{\psi}_A^L, \widehat{\psi}_B^L, \widehat{\psi}_C^L) = (-0.691667, -0.291667, -0.016667).$$

The three values sum to -1, equal to the difference between the full-execution loss and the full-cancellation loss. A makes both the fact nodes and subsequent derivations possible; B completes numerical and rule-based derivations in backgrounds where the facts are available; and C adds one final decision node, while repeated presentation of predecessor quantities remains semantic continuity. This allocation reflects the task effects of the three transformations across execution backgrounds.

Checks of all 648 records find no violations of identity cancellation, unauthorized modifications of predecessor nodes, or inconsistent generation for identical prefixes. The maximum numerical error in the Behavioral Shapley efficiency identity across all configurations is below 10^{-15}.

G.4 Mode Quality, Individual Incentives, and Equilibrium

The three homogeneous mode configurations produce clear differences in terminal loss (Table 9). All three replays of the all-Careful configuration have zero loss; the all-Shortcut configuration has a mean loss of 0.450; and the all-Distorter configuration has a mean loss of 0.533.

Table 9: Terminal semantic losses of the three homogeneous mode configurations.

Mode configuration	Losses in three replays	Mean loss
Careful / Careful / Careful	0.000, 0.000, 0.000	0.000
Shortcut / Shortcut / Shortcut	0.450, 0.450, 0.450	0.450
Distorter / Distorter / Distorter	0.550, 0.550, 0.500	0.533

With the other two positions fixed to Careful, the individual penalty at each of the three positions is minimized by choosing Careful (Table 10). The three penalties are $-0.692, -0.500, -0.350$ for A; $-0.292, -0.108, -0.233$ for B; and $-0.017, 0, 0$ for C. Each position therefore strictly prefers Careful. Even C, which adds only one final decision node, receives an incentive signal that distinguishes its modes.

Table 10: Behavioral Shapley loss penalties for unilateral mode choices with the other positions fixed to Careful. The minimum in each row is bold.

Transformation position	Careful	Shortcut	Distorter
A: fact extraction	$\mathbf{-0.692}$	-0.500	-0.350
B: calculation and rule-based derivation	$\mathbf{-0.292}$	-0.108	-0.233
C: decision synthesis	$\mathbf{-0.017}$	0.000	0.000

Exhaustive enumeration under the characteristic function averaged over the three stochastic replays yields a unique pure-strategy equilibrium:

$$a^{\mathrm{NE}} = (\text{Careful}, \text{Careful}, \text{Careful}).$$

This configuration is also the unique global task optimum among the 27 configurations, with both equilibrium and optimal loss equal to 0. Starting from each of the 27 initial configurations, strict best responses in the repeating order A, B, C always converge to this configuration. Figure 7 summarizes terminal losses, individual penalties, and cancellation replay results.

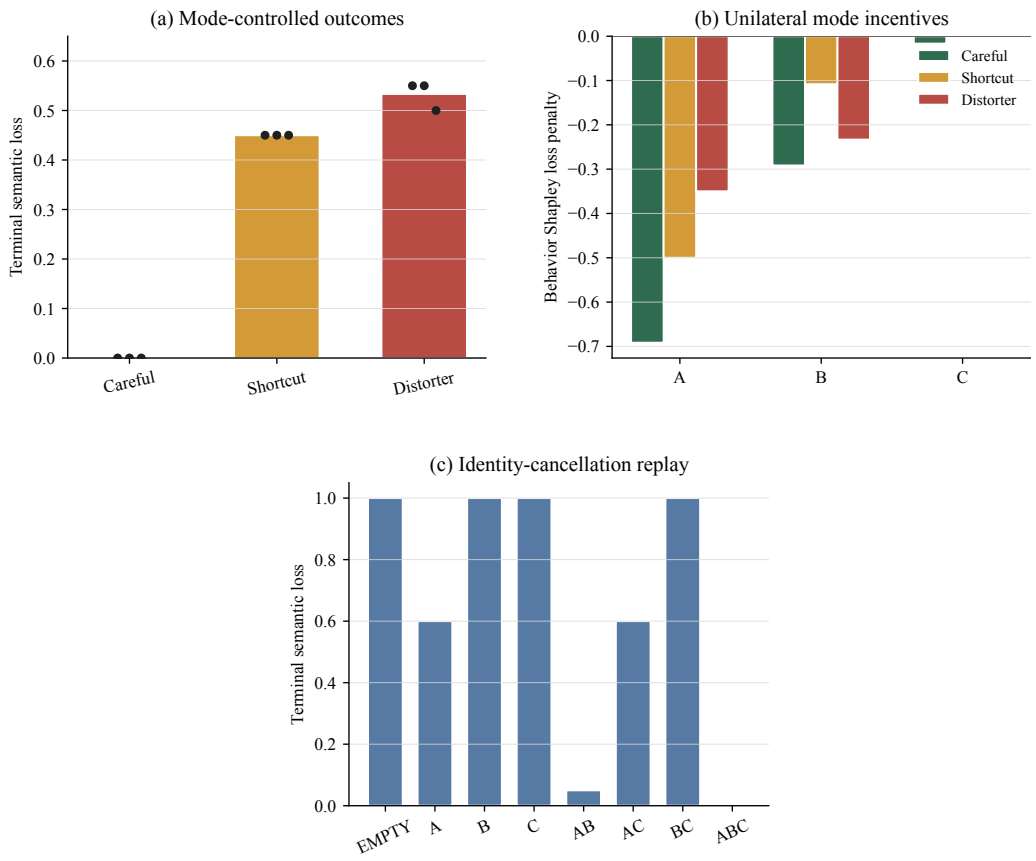


Figure 7: Controlled incentive experiment for semantic transformation modes. (a) Terminal semantic loss under homogeneous mode configurations, with black dots indicating the three samples; (b) individual Behavioral Shapley penalties with the other two positions fixed to Careful; (c) identity-cancellation replay losses for the eight retained subsets under the all-Careful configuration. All labels in the figure are in English.

To check the exact potential relation of the mode game, we compute the Shapley potential from the same subset losses:

$$\mathcal{P}(a) = \sum_{\varnothing \neq U \subseteq N} \frac{(|U|-1)!(|N|-|U|)!}{|N|!}\, \widehat{v}_a^L(U). \tag{41}$$

Across all configurations and unilateral mode changes, the maximum absolute difference between the change in individual penalty and the change in $\mathcal{P}$ is 2.22×10^{-16}. The loss table obtained through actual conditional LLM generation therefore exhibits numerical consistency with the exact potential relation and yields a unique equilibrium that coincides with the task-optimal configuration.

G.5 Experimental Conclusion

This controlled procurement experiment connects semantic transformation, identity cancellation, downstream regeneration, and Behavioral Shapley incentives. When a transformation is canceled, existing semantics remain visible and subsequent outputs change with the counterfactual state. Quality differences between modes enter individual loss penalties, leading strict best responses from all 27 initial configurations to the unique zero-loss equilibrium. The experiment directly illustrates how, under a shared cancellation reference, Behavioral Shapley provides incentives for semantic transformation mode choices that align with the task objective.

H Formal Definitions of Semantic Nodes and Behaviors

Remark 1 (Semantic nodes and semantic behaviors)**.** Following SCG, a semantic node s represents a local meaning that can be located within an execution state. Given a task and context, a semantic discriminator $c(z, z') \in \{0, 1\}$ determines whether two expressions convey the same task-relevant content, thereby identifying the different linguistic realizations of that meaning, denoted by $\mathcal{C}_s$. A node also retains its execution location and textual carrier; occurrences of the same meaning at different locations are recorded separately to preserve their provenance and transmission.

A semantic behavior is a local semantic operation that actually occurs during an agent call. It can be recorded as $b = (u, P_b, s_b)$ using its execution location u, predecessor node set P_b, and output node s_b; its executor and operation content are determined from the call log. Preservation carries a predecessor meaning into a new execution state; transformation produces a different meaning from one or more predecessors, including by combining, correcting, or changing their qualifications; introduction records a new meaning with no predecessor semantic node within the current tracing scope. A single call may contain several such operations, and operations with multiple inputs naturally correspond to directed hyperedges $P_b \to s_b$. Thus, the node partition identifies distinguishable local content, while behaviors record how this content is processed during execution.

One testable criterion for a predecessor relation is counterfactual dependence in generation:

$$p_u(s_b \mid x_u) - p_u(s_b \mid x_u \setminus P_b) \geq \delta, \qquad \delta > 0. \tag{42}$$

Here, x_u is the factual input to the call, and $p_u(s_b \mid x)$ is the probability, under fixed generation settings, that the discriminator judges the output to express the meaning contained in s_b; $x_u \setminus P_b$ is the input after predecessor meanings are removed according to a predeclared editing rule, and δ is a chosen threshold for a substantive decrease. This condition provides a test of whether candidate predecessors support the generation of the target meaning, while allowing different tasks to adopt other explicit support criteria. In our implementation, an LLM discriminator identifies local meanings and their generation relations from execution records. Given the recovered semantic graph, BCAST intervenes on specific behaviors and replays subsequent execution to measure their effects on task loss, connecting semantic structure identification to the determination of outcome values for behaviors.

I Relation to Existing Attribution and Intervention Methods

Related methods can be connected to our framework along three dimensions: behavior generation mechanisms, intervention scope, and subsequent propagation. Based on the semantic generation structure of SCG (Jiang et al., 2026), we treat an introduction, preservation, or transformation of meaning that actually occurs during execution as a behavior. Several behaviors can form a set for joint evaluation or intervention: a message corresponds to all the semantic behaviors it contains, while an agent corresponds to all its behaviors across calls, rounds, and branches within the analytical scope. Analyses at the message, step, component, and agent levels thus correspond to different groupings of behaviors.

Behavior grouping determines the intervention target, while semantic modes provide the generation mechanisms for execution after intervention. We summarize the basic assumption of counterfactual reruns as follows: *Invariant generative modes underlie changing behaviors.* When an intervention changes a preceding semantic state, the parts not directly intervened upon retain their original modes and generate behaviors from the new inputs. Modes can be implemented through settings such as models, prompts, tool configurations, or policies. Counterfactual analysis therefore requires specifying both which behaviors are canceled or replaced and how the remaining behaviors are regenerated.

Agent and step attribution in LLM systems. Who&When (Zhang et al., 2025) introduces a failure attribution benchmark and evaluates methods for identifying responsible agents and critical steps from execution logs; AgenTracer (Zhang et al., 2026) constructs responsibility labels through replay after action repair and programmatic fault injection, then trains models for diagnosis. These methods define the scope of responsibility in terms of executors and step locations. In our framework, a step corresponds to all the semantic behaviors actually performed within it: the diagnostic methods in Who&When identify this behavior group, whereas AgenTracer's repair branch replaces the action at that step and tests whether the outcome improves through behaviors generated by subsequent

modes on the new inputs. SCG further distinguishes semantic operations within a step, allowing responsibility analysis to identify specific semantic behaviors.

Message and component interventions in LLM systems. C3 (Chen et al., 2026) replaces an entire message while holding the generation policy and pre-intervention interaction history fixed, then runs subsequent agents to compare the returns of candidate messages. This corresponds to jointly replacing all semantic behaviors within a message: the candidate message provides the replacement reference, while downstream agents generate behaviors adapted to the new history under their original modes. DoVer (Ma et al., 2026) modifies orchestrator messages or plans and resumes execution to test failure hypotheses, corresponding to joint interventions at the message or plan level. RAGGY (Romero Lauro et al., 2025) allows the configurations, inputs, or outputs of retrieval or generation components to be modified, with a choice between local re-execution and downstream continuation; configuration changes correspond to changes in behavior generation modes, while output changes correspond to replacing realized behaviors. These methods treat the behaviors within a message, step, or component jointly, whereas our semantic representation allows individual behaviors within a group to be selected for evaluation.

Local error and propagation analysis in LLM systems. EDGE (Hou et al., 2026) repairs erroneous segments of a trajectory and regenerates subsequent reasoning while holding tool outputs fixed to determine whether one type of error affects another. Its method involves two independent choices: the erroneous segment determines which behaviors are modified, and fixing tool outputs restricts propagation after intervention. In our framework, the former corresponds to a set of local semantic behaviors and the latter to the replay rule. Analytical granularity and propagation scope can therefore be specified independently: behaviors within a segment can be replaced jointly, or specific semantic operations within it can be intervened upon, and their effects evaluated under the specified continuation rule.

Action attribution and effect propagation in non-LLM systems. MACIE (Weinberg, 2025) studies multi-agent reinforcement learning, replacing all actions of a target agent over a given time horizon with a reference policy and simulating the responses of other agents and the environment through a structural causal model. In our behavior representation, this is a joint intervention covering all decisions of the target agent: the target behaviors follow a common reference generation rule, while the remaining parts respond according to their specified mechanisms. ASE (Triantafyllou et al., 2024) studies multi-agent MDPs, distinguishing a source action from the subsequent actions of designated agents and measuring the source action's effect propagated through those agents by transplanting subsequent counterfactual actions. It additionally specifies which subsequent behaviors propagate the effect and which counterfactual trajectory supplies the replacement actions. These methods are grounded in environment states, actions, and causal models; using SCG, we establish semantic behavior units in language workflows that can be located and intervened upon individually.

These correspondences describe how each method's intervention targets and execution interfaces can be expressed in a behavior representation; they do not automatically establish numerical equivalence. Recovering the execution comparisons of an original method also requires matching its reference operations, conditional distributions, propagation restrictions, and any necessary cross-world coupling of randomness; ASE's action transplantation particularly requires this additional structure. When these conditions are met, the semantic behavior representation further allows the analysis of behaviors within groups and combinations across groups. SCG provides the structure of behaviors and generative support, BCAST estimates outcome values for behavior combinations through replay, and methods such as LOO and Shapley then evaluate or allocate these values according to the analytical objective.

J Proof of the Equilibrium Loss Guarantee

Proof. First, we show that an equilibrium configuration is close to a minimizer of Φ. At a pure-strategy equilibrium a^{NE}, no participant can reduce its individual penalty by changing its mode unilaterally. Combining this condition with $|\psi_i^L(a) - g_i(a_i)| \le \varepsilon_i/2$ gives $g_i(a_i^{\mathrm{NE}}) - \min_{a_i} g_i(a_i) \le \varepsilon_i$. Since Φ is the sum of these terms, $\Phi(a^{\mathrm{NE}}) - \min_a \Phi(a) \le E$.

Next, we show that the potential function and task loss yield similar comparisons between configurations. For any two configurations a and a', replace their differing components one at a time and sum $|\Delta L - \Delta\Phi| \le \varepsilon_i$ to obtain $|(L(a) - L(a')) - (\Phi(a) - \Phi(a'))| \le E$. Taking $a^{\mathrm{opt}} \in \arg\min_a L(a)$ therefore gives

$$\begin{aligned} L(a^{\mathrm{NE}}) - L(a^{\mathrm{opt}}) &\le \Phi(a^{\mathrm{NE}}) - \Phi(a^{\mathrm{opt}}) + E \\ &\le \Phi(a^{\mathrm{NE}}) - \min_a \Phi(a) + E \le 2E. \end{aligned}$$

When $E = 0$, the equilibrium and optimal configurations have the same loss.

If each participant's mode $a_i^\star$ satisfies the advantage condition in the theorem, then, regardless of the other participants' settings, switching from any $a_i \neq a_i^\star$ to $a_i^\star$ reduces both the individual penalty and task loss by at least $g_i(a_i) - g_i(a_i^\star) - \varepsilon_i > 0$. Thus, every participant strictly prefers $a_i^\star$, and their combination is the unique pure-strategy equilibrium. Starting from any other configuration, successively switching to each participant's $a_i^\star$ also strictly reduces task loss, so this combination is also the unique task-optimal configuration. □

K Accountability Protocols and Connections to Existing Methods

This procedure provides a computational basis for accountability protocols among multiple parties. Before collaboration begins, the parties can agree on how semantic behaviors are identified, the task loss, cancellation or replacement references, replay scope, and allocation rule; after the task, responsibility can be computed and reviewed using the recorded interactions and generation settings. Each responsibility allocation can therefore be traced to a specific behavior, the counterfactual operation applied, and the resulting change in outcome. When task loss is measured in monetary terms, responsibility allocations relative to a common reference can also support deductions, reductions in liability, and loss sharing. BCAST provides behavior valuation and responsibility computation for assisted annotation, service evaluation, and financial rewards and penalties within such protocols.

This framework also provides a common description of existing methods. Prior work analyzes failures by identifying responsible agents and critical steps (Zhang et al., 2025; 2026), and tests how changes affect outcomes by replacing messages, re-executing components, or repairing local segments (Ma et al., 2026; Chen et al., 2026; Romero Lauro et al., 2025; Hou et al., 2026). In non-LLM multi-agent systems, MACIE and ASE study policy replacement and the propagation of action effects, respectively (Weinberg, 2025; Triantafyllou et al., 2024). In our framework, the behaviors within messages, steps, and components can each form an object of analysis, whereas agent-level analysis covers all behaviors performed by that agent within the specified scope. Failure localization identifies the behavior groups that require examination; a joint intervention replaces behaviors within the corresponding scope and generates subsequent behaviors according to specified rules. Constructing the corresponding execution comparisons within this framework requires retaining the original method's reference operations, propagation rules, and evaluation metrics, while matching its conditional distributions and any necessary coupling of randomness.