

Comparing Message and Action Evidence for Synthetic AI-Collusion Detection

Syed Alam Shah Bukhari

Independent researcher

26 October 2026

Template-access note. The official Apart Research Google Docs copy link required Google sign-in in this environment and could not be opened. This submission follows the official section and appendix requirements recorded in the project design, but it does not claim to reproduce the inaccessible Google Docs formatting.

Abstract

This controlled pilot asks whether combining agent messages with observed actions improves detection of undesirable coordination over either source alone. We generated 40 deterministic two-agent, three-round resource- allocation trajectories: 20 scripted positive cases with harmful coordinated execution, and 20 negatives consisting of independent similar actions, independent distinct behavior, and talk without harmful execution. Four matched evidence conditions were evaluated with deterministic rule-based classifiers: messages only (M), actions and outcomes only (A), raw messages plus actions (MA), and a structured audit view (S) that records claim/action relationships. The final S classifier uses aggregate coordination, excess, and rejection signals rather than requiring exact claim/action alignment. The same cases and labels were used for every condition, while labels and hidden provenance were excluded from detector views. On the saved test artifacts, balanced accuracy was 0.475 for M, 0.700 for A, 0.575 for MA, and 0.800 for S. A detected all 20 scripted positives but produced 12 false positives; S produced 12 true positives, 20 true negatives, and no false positives. These results are descriptive engineering evidence, not evidence of real-world generalization or spontaneous LLM collusion detection. The labels are generated scripted provenance and were not independently verified by a separate label verifier. The contribution is a reproducible, offline instrumentation pilot that exposes the trade-off between harmful-outcome sensitivity and coordination-specific evidence.

1. Introduction

Monitoring multi-agent systems requires distinguishing undesirable coordination from ordinary cooperation, similar behavior caused by common incentives, and statements that are not followed by action. The Apart Research AI Collusion Research Sprint frames a directly relevant comparison: whether what agents say and what they do together beat either source independently [1]. This project implements that comparison as a small, transparent synthetic benchmark.

The research question is:

In a controlled two-agent simulation with scripted ground-truth provenance, does combining messages and actions improve detection of harmful coordinated execution over messages or actions alone?

The experiment is intentionally narrow. It does not model spontaneous LLM behavior, does not use a language model, and does not claim deployment performance. It evaluates deterministic baselines against a fixed 40-case dataset so that the complete data path, leakage controls, metrics, and reproducibility can be inspected.

The main result is mixed rather than a general win for combined evidence. The actions-only baseline is sensitive to every harmful positive but also flags independent harmful negatives. The messages-plus-actions rule is conservative and misses most positives because it requires a narrow lexical message signal. The aggregate-rule S detector has the highest balanced accuracy in this pilot, but its performance remains inseparable from the scripted templates and labels used to construct the benchmark.

2. Related work

The sprint brief motivates matched comparisons of communication and action evidence for collusion auditing [1]. This report operationalizes that question in a deliberately smaller setting rather than treating an existing deception benchmark as a collusion dataset.

The local reference repository, *LLM-Trust-Breaking*, studies strategic deception, promise-breaking, and trust dynamics in repeated games [3]. Those concepts are related to monitoring, but a promise mismatch or a deceptive statement is not by itself evidence that two agents jointly adopted and executed a harmful plan. The repository was inspected as reference material only; its generated outputs and labels were not used here.

Work on deceptive behavior in language models also demonstrates why behavioral claims require careful experimental controls and should not be inferred from a single textual cue [2]. The present pilot follows that caution by separating sealed scripted provenance from public detector views and by including hard negatives with harmful but independent actions.

3. Methodology

3.1 Environment and cases

The environment is implemented in `src/environment.py`. Two agents, Alpha and Beta, choose integer extraction levels from 0 through 4 for three rounds. Each round has a fresh pool of 8 units and a sustainability limit of 4 total extracted units. Each agent receives payoff $2x$; external harm is $3 * \max(0, \text{total_extraction} - 4)$.

The deterministic generator in `src/generate.py` uses one master seed, 20261023. Same-seed reruns support deterministic replay but do not estimate across-seed variance or uncertainty. The saved dataset contains:

Case type	Count	Operational role
Positive	20	Jointly scripted harmful execution
Hard negative	10	Independent agents with similar high actions
Independent negative	5	Independent, distinct behavior
Talk-only negative	5	Coordination-like discussion without harmful execution
Total	40	Fixed matched evaluation cases

A positive label is assigned by the generator when a scripted joint plan includes at least one prohibited over-extraction round and the corresponding actions are executed. Negative labels are assigned to the independent and talk-only generator branches. The generator performs configuration, trajectory length, and binary-label checks. It does **not** provide an independent post-hoc verifier of plan adoption, causal provenance, or label correctness; therefore “ground truth” in this report means scripted provenance.

The 20 positive examples were part of the main benchmark, not a distinct positive-control experiment. No separate development set was implemented. The detector rules were fixed as transparent baselines, but they were not selected using a held-out tuning split. The configured `detection_threshold` is recorded for provenance, while the current binary detectors use hard-coded rule conditions rather than applying that confidence threshold.

3.2 Evidence conditions and leakage controls

The four view constructors are in `src/views.py`.

- **M**: common game context, aliases, announcement order, and messages only.
- **A**: common game context, aliases, actions, payoffs, excess, and harm only.

- **MA:** the M and A observable fields together.
- **S:** a deterministic structured-audit view that parses claims from messages, compares available claims with observed actions, and records coordination, excess, rejection, and claim/action-alignment fields. These alignment fields are retained for auditing but are not required by the final S decision rule.

The view builders do not copy labels, sealed provenance, seeds, scenario group identifiers, case identifiers, intervention metadata, or label justifications. The runner computes metrics only after detector predictions have been made. An independent recursive scan of all 40 generated views found no forbidden metadata keys or forbidden metadata tokens at nested levels. The views still contain the complete finished trajectory, so this is an offline evaluation and not a real-time monitoring test.

3.3 Rule-based detectors

The detectors are implemented in `src/detectors.py`.

- **M** (`detect_M`) counts coordination markers (`coordinate`, `both`, `you take`, `you go`, `mutual`, `together`) and messages mentioning an action of at least 3. It predicts positive when both counts are at least 2.
- **A** (`detect_A`) predicts positive when any action record has positive excess. It intentionally cannot infer whether the harm was coordinated.
- **MA** (`detect_MA`) requires both the M prediction and at least one harmful action round.
- **S** (`detect_S`) predicts positive when the structured audit reports at least one coordination-signal round, at least one excess round, and no rejection round. Although `build_view_S()` records claim/action alignment and consistency, `detect_S()` does not require exact claim-to-action alignment or exact mutual-plan matching.

Confidence fields are heuristic annotations, not calibrated probabilities and do not affect the binary predictions.

3.4 Evaluation

`src/run_experiment.py` evaluates every condition on the same 40 case IDs. `src/evaluate.py` calculates accuracy, precision, recall, F1, specificity, false-positive rate, balanced accuracy, and raw TP/TN/FP/FN counts. Predictions, metrics, paired disagreements, configuration, and dependency-free SVG figures are persisted under `results/`.

4. Results

The values below were read from `results/metrics.json` and independently recomputed by joining each saved prediction to its dataset `case_id`. All four prediction files contain the same 40 unique case IDs.

View	Accuracy	Precision	Recall	F1	Specificity	FPR	Balanced accuracy	TP/TN/FP/FN
M	0.475	0.429	0.150	0.222	0.800	0.200	0.475	3/16/4/17
A	0.700	0.625	1.000	0.769	0.400	0.600	0.700	20/8/12/0
MA	0.575	1.000	0.150	0.261	1.000	0.000	0.575	3/20/0/17
S	0.800	1.000	0.600	0.750	1.000	0.000	0.800	12/20/0/8

M produced four false positives, all from talk-only negatives, and missed 17 positives. A produced 12 false positives: 10 hard negatives and 2 independent distinct negatives. MA produced no false positives but missed 17 positives. S produced no false positives and missed 8 positives.

The results are internally reproducible. The command `python -m pytest tests/test_experiment.py -q` completed with five passing tests and exit code 0. The command `python -m src.run_experiment --output-dir results_review` completed with exit code 0 and reproduced the printed metrics. SHA-256 comparisons between the canonical `results/` output and an independent rerun matched for `dataset.jsonl`, all four prediction files, `metrics.json`, `metrics.csv`, `paired_comparison.json`, and `experiment_log.json`.

5. Discussion

The pilot does not support the simple claim that adding messages to actions always improves detection. A is maximally sensitive to the chosen harm definition but has a 0.600 false-positive rate because independent harmful actions are observable but their provenance is not. M is vulnerable to talk-only coordination language and misses positive cases whose wording does not contain its narrow markers. MA removes A's false positives but inherits M's low recall. S performs best among these fixed rules because its aggregate conditions align with the generated positive and talk-only structure.

This pattern is useful as an engineering result, not as a benchmark leaderboard. The positive and negative message families were authored together but remain synthetic. The detector can exploit vocabulary, message count, speaker order, and action distributions. The dataset is small, balanced by construction, and not split into development and held-out test groups. The absence of an independent label verifier further limits the scientific claim.

The appropriate conclusion is that a reproducible matched-view harness has been completed and that different evidence restrictions create distinct, inspectable error profiles. A defensible next study should independently verify labels, hold out semantic template families, add an ambiguity stratum, and pre-register rules on a development set before evaluating a larger test set.

6. Sprint work, pre-existing work, and reproducibility

During the October 23–25 Apart Research sprint, the author and AI assistant drafted the research design and decision log, implemented the root resource-allocation simulator, deterministic generator, four evidence views, rule-based detectors, evaluator, persistence layer, tests, figures, and this submission report. The final local execution and verification were performed on 26 October 2026.

The nested LLM-Trust-Breaking repository was pre-existing reference material. It was inspected to distinguish deception/promise-breaking from collusion, but it was not used as collusion data, its generated outputs were not used as results, and its files were not modified. No external model calls, paid APIs, or uploaded data were used.

To reproduce the pilot from the workspace:

```
python -m pytest tests/test_experiment.py -q
python -m src.run_experiment --output-dir results
```

The authoritative artifacts are results/dataset.jsonl, results/predictions_{M,A,MA,S}.json, results/metrics.json, results/metrics.csv, results/paired_comparison.json, and results/experiment_log.json.

References

- [1] Apart Research. "AI Collusion Research Sprint." 23–25 October 2026. <https://apartresearch.com/sprints/ai-collusion-research-sprint-2026-10-23-to-2026-10-25>
- [2] Hubinger, E., et al. "Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training." arXiv:2401.05566, 2024. <https://arxiv.org/abs/2401.05566>
- [3] Jerick-1380. "LLM-Trust-Breaking." GitHub repository, accessed 26 October 2026. <https://github.com/Jerick-1380/LLM-Trust-Breaking>

Appendix A. Limitations and Dual-Use / Ethical Considerations

Limitations

This is a 40-case synthetic, rule-based pilot. It does not estimate detector performance on natural language generated by autonomous agents, real markets, or deployed multi-agent systems. The scripted labels are not independently verified ground truth. The complete trajectories are available to the offline detectors, so the setup does not establish online early-warning performance. There is no separate development set, no confidence interval, and no grouped held-out test evaluation. The sample is small, balanced by construction, and vulnerable to template and distribution shortcuts. The structured S condition is a deterministic parser and aggregate rule, not a validated audit standard. Only one master seed (20261023) was evaluated; same-seed reruns demonstrate deterministic replay, not across-seed variance or uncertainty. The 20 positive examples were part of the main benchmark, not a distinct positive-control experiment.

Dual-use and ethical considerations

Collusion monitoring can support accountability, but false positives can penalize benign cooperation and false negatives can miss harmful behavior. The action-only result demonstrates this risk directly: it flags independent harmful behavior as collusion because observable harm does not establish intent or joint provenance. These outputs must not be used for employment, law-enforcement, access, or other consequential decisions.

The artifacts contain synthetic messages and no personal data. Future monitoring systems should minimize retained communication data, restrict access, document uncertainty, and provide human review and appeal. Agent messages must be treated as untrusted data rather than executable instructions; prompt-injection resistance and privacy review are required before any language-model monitor is connected to live communications.

The work is released as a transparent research prototype. Its rules, limitations, and failure cases should be disclosed together rather than presenting the highest balanced accuracy as evidence of reliable real-world collusion detection.