Autonomous Trust: Self-Gating Evaluation as a Prerequisite for Agent-to-Agent Communication at Scale

Nehal Sangoi, Kris Feldmann, Deven Yadav, Sandeep Loi, and Karan Gandhi
Independent Researchers
Nehal Sangoi, Santa Monica, CA, USA (e-mail: nehal.sangoi@gmail.com)
Kris Feldmann, Los Angeles, CA, USA (e-mail: kris@feldmannfamily.com)
Deven Yadav, Bakersfield, CA, USA (e-mail: deven.yadav@gmail.com)
Sandeep Loi, Chicago, IL, USA (e-mail: sandeeploi100@gmail.com)
Karan Gandhi, Dallas, TX, USA (e-mail: karanjg@gmail.com)

***Abstract*—As autonomous agents backed by large language models (LLMs) move from single-user assistants toward peer systems that transact directly with one another, the volume of agent-to-agent (A2A) communication is widely expected to grow rapidly enough that, at some point, no human reviewer can vet each outbound action, yet LLM output quality is non-stationary: a single model can produce an excellent decision at one turn and an unsafe or incoherent one at the next, with no guarantee tied to prior behavior. This paper argues that trust in such systems cannot be modeled on human trust, which is accumulated through track record, nor can it be delegated to the LLM itself, since the model is fundamentally an input-output function with no internal mechanism for self-policing. We propose Autonomous Trust, a framework in which trustworthiness is engineered as a property of the agent as a whole (LLM plus an external evaluation layer) rather than of the model in isolation. The framework rests on four design principles: (1) pre-transmission self-gating, in which the sending agent, not only the receiver, is responsible for intercepting its own unsafe outputs before they leave the system; (2) a verifiability taxonomy that routes each action to an appropriate gating strategy, from deterministic checks to LLM-panel adjudication; (3) continuous self-reported quality telemetry, analogous to application health metrics, that exposes an agent's own output degradation to external monitoring in real time; and (4) explicit treatment of the "gameable verifier" failure mode, in which self-evolving agents can degrade their own automated checks (for example, by authoring tests engineered to always pass). We formulate the design space, analyze failure modes including judge-panel correlated blind spots, and report preliminary empirical validation: a Tier-1 self-gating pilot across independent blind code-generation runs, and a Tier-2 panel-adjudication pilot comparing single-judge against multi-judge detection of action/rationale inconsistencies. Building on these pilot results, we outline a larger empirical evaluation plan. This work contributes a concrete engineering framework, not a purely theoretical trust model, toward the near-term infrastructural challenge of building safe, unsupervised, internet-scale agent ecosystems.**

***Index Terms*—autonomous agents, agent-to-agent communication, large language models, AI safety, self-evaluation, multi-agent trust, LLM-as-judge, reward hacking**

I. Introduction

Autonomous software agents built on large language models are increasingly deployed not as isolated assistants responding to a single human, but as participants in pipelines where they consume the output of other agents and produce output consumed in turn: planning agents that delegate to execution agents, retrieval agents that feed synthesis agents, and negotiation agents that exchange proposals with counterpart agents operating on behalf of other systems or organizations. As this pattern matures, agent-to-agent (A2A) communication looks likely to become a communication layer in its own right. We do not have a firm empirical basis for predicting exactly how large this traffic will get, and some published projections comparing it to host-to-host internet traffic should be read as illustrative rather than measured; the more defensible claim, and the one this paper relies on, is structural rather than quantitative: many autonomous endpoints transacting continuously, largely without a human watching any individual exchange, at a volume that scales with the number of agent pairs and their interaction frequency rather than with available human review capacity.

This scale is the crux of the problem this paper addresses. An LLM's output quality is not monotonic and not reliably predicted by its recent history: the same model, on the same class of task, can produce a well-reasoned action at one invocation and a harmful, incoherent, or simply wrong one at the next. When a human is in the loop, this variability is tolerable because a human is there to catch the mistake before it goes anywhere. When the recipient of an action is another autonomous agent, and the volume of actions makes per-action human review infeasible, that check must be automated, and it must be reliable enough that its absence does not simply relocate the risk from "bad LLM output" to "bad automated approval."

The central claim of this paper is that trust in such a system cannot be modeled the way trust between humans is modeled, earned incrementally through a track record, because a single LLM invocation carries no memory-linked guarantee that the next invocation will resemble the last. Nor can responsibility for safety be assigned to the LLM itself, since a model is, architecturally, an input-in/output-out function with no native mechanism to inspect or veto its own proposals.

Trustworthiness, therefore, has to be engineered as a property of the *agent as a system*: the LLM together with an evaluation mechanism external to it, capable of intercepting a proposed action before it is transmitted.

This paper makes the following contributions:

1) A problem formulation for trust in unsupervised, LLM-backed A2A communication at scale, distinguishing it from prior human-in-the-loop and long-run-reputation trust models.
2) **Autonomous Trust**, a framework built around pre-transmission self-gating: the principle that the sending agent, not only the receiver, is responsible for catching and stopping its own unsafe or low-quality outputs.
3) A verifiability taxonomy that assigns each class of agent action to an appropriate automated gating strategy, from deterministic checks through LLM-panel adjudication.
4) A self-reported quality telemetry model through which an agent exposes its own output degradation to external monitoring, analogous to application performance metrics.
5) An analysis of the "gameable verifier" failure mode, the risk that a self-evolving agent degrades the very checks meant to constrain it, together with candidate mitigations.
6) A preliminary empirical validation of the framework's Tier-1 and Tier-2 gating mechanisms, using a small original code-generation benchmark and a panel-adjudication pilot, reported alongside their limitations.

The remainder of this paper is organized as follows. Section II reviews related work in multi-agent LLM systems, computational trust, and LLM-as-judge evaluation. Section III formalizes the problem. Section IV presents the Autonomous Trust framework. Section V analyzes the gameable verifier problem in depth. Section VI walks through an illustrative A2A scenario. Section VII discusses limitations and threats to validity. Section VIII reports preliminary empirical validation. Section IX outlines remaining future empirical work, and Section X concludes.

II. Related Work

Multi-agent LLM systems. Frameworks such as AutoGen [1] and agent architectures built on the ReAct paradigm [2] demonstrate that LLM-backed agents can coordinate, delegate, and exchange intermediate results to solve tasks beyond the reach of a single model invocation. Generative agent simulations [3] further show that populations of LLM-backed agents can sustain extended interaction with emergent, believable behavior. This body of work establishes the feasibility of agent-to-agent coordination but generally assumes cooperative, sandboxed settings rather than treating the reliability of each outbound message as a first-class design constraint.

Computational trust and reputation models. Trust and reputation in multi-agent systems is a long-studied problem in distributed systems, predating the LLM era [4]. Classical models typically compute trust from accumulated interaction history: a track record of past behavior used to predict future reliability. This paper argues that such models transfer poorly to LLM-backed agents, whose output quality can vary sharply between consecutive invocations regardless of historical performance, making single-action risk, rather than long-run reputation, the more relevant unit of analysis.

LLM-as-judge evaluation. Using one LLM to evaluate the output of another has emerged as a practical evaluation strategy, shown to correlate reasonably well with human judgment on open-ended tasks [5]. However, this approach is self-referential: the judge model is subject to the same classes of failure as the model it evaluates. Ensemble or panel-based judging has been explored as a way to reduce single-judge variance, though correlated errors across judges remain an open problem. This paper treats that tension directly as part of the Autonomous Trust design space, rather than assuming it away.

Reward hacking and specification gaming. The risk that an optimized system satisfies the letter of an automated check while violating its intent is well documented in the AI safety literature under the terms reward hacking and specification gaming [6], [7]. This paper extends that concern to a specific and underexplored instance: a self-evolving coding or task agent authoring or maintaining its own verification logic (tests, linters, acceptance checks), creating an incentive gradient toward checks that are easy to pass rather than checks that are correct.

Self-critique and self-refinement. Techniques such as Self-Refine [8] and Constitutional AI [9] show that models can be prompted to critique and revise their own output, improving quality without additional training. These approaches operate *within* the generative process, whereas Autonomous Trust treats self-critique as necessary but insufficient, and requires an evaluation stage architecturally external to the generating model as a precondition for outbound transmission.

Gap. No existing framework we are aware of combines (a) pre-transmission, sender-side self-gating as an architectural requirement for A2A communication, (b) a verifiability-based taxonomy routing actions to tiered automated checks, and (c) continuous self-reported quality telemetry as an observability primitive for autonomous agents operating without per-action human supervision. Autonomous Trust is proposed to fill this gap.

III. Problem Formulation

We consider a setting in which an autonomous agent A, backed by an LLM, produces a sequence of candidate actions $a_1, a_2, \ldots, a_n$, each of which, if transmitted, is consumed by a receiving agent B (which may itself be LLM-backed) or by a downstream system. An action may be a message, a data artifact, a code change, an API call, or a task-completion signal.

We adopt the following constraints, drawn directly from the operating conditions of A2A systems at scale:

- **C1: No per-action human review.** At the traffic volumes anticipated for A2A communication, a human cannot inspect each a_i before transmission.
- **C2: Non-stationary output quality.** The probability that a_i is safe, sane, and correct is not reliably predicted by the

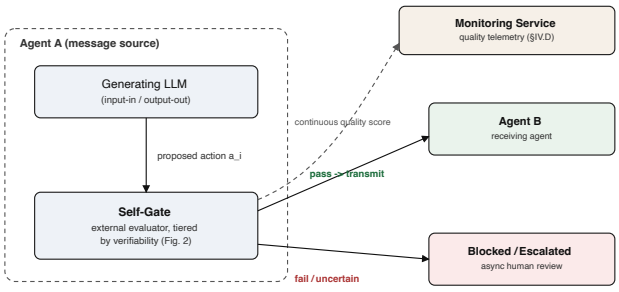

Fig. 1. Autonomous Trust architecture: pre-transmission self-gating with external evaluation and quality telemetry.

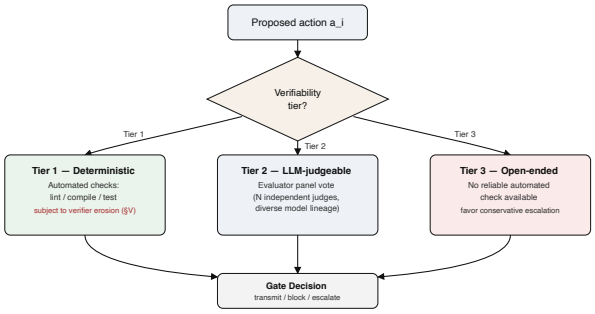

Fig. 2. Verifiability taxonomy: each action is routed to a gating strategy suited to its tier.

quality of $a_1, \ldots, a_{i-1}$; a single invocation can diverge sharply from recent history.

- **C3: Verifier erosion.** Where automated checks exist to validate a_i (e.g., tests, linters), those checks are themselves artifacts that may be authored, maintained, or modified by the same agent population over time, creating a channel through which check reliability can degrade without an explicit adversary.
- **C4: Incomplete verifiability.** Not all action classes admit a deterministic or ground-truth check; many require qualitative judgment of coherence, appropriateness, or safety.

The problem, then, is to define a function $G(a_i) \rightarrow \{\text{transmit}, \text{block}, \text{escalate}\}$, evaluated by agent A itself prior to transmission, that holds a low false-negative rate (unsafe actions transmitted), a low false-positive rate (safe actions blocked), and reasonable computational overhead, under constraints C1–C4, without assuming G is trustworthy simply because it is automated.

IV. Proposed Framework: Autonomous Trust

Fig. 1 shows the overall architecture: the sending agent's evaluation layer sits outside its generating model, intercepts every proposed action before transmission, and continuously reports a quality signal to an external monitoring service independent of any single gate decision.

A. Architectural Principle

Autonomous Trust treats trustworthiness as a property of the pairing (LLM, evaluator), not of the LLM alone. The evaluator is architecturally external to the generating model: it does not share the generating model's context, weights, or failure correlation by default, and it has unilateral authority to prevent transmission. This mirrors the observation that a purely input-in/output-out generative function has no internal channel by which to reliably audit itself; the audit must be a separate computational step.

B. Pre-Transmission Self-Gating

The framework's defining requirement is that the *sending* agent, not solely the receiver, is responsible for evaluating an action before it is transmitted. This differs from receiver-side filtering (where an untrustworthy action is caught only after leaving its origin) in two ways: it reduces the propagation of bad actions across a multi-hop agent network before any downstream cost is incurred, and it allows the evaluation to use context and instrumentation available only at the origin (e.g., the generating model's confidence signals, intermediate reasoning artifacts, or access to the tools that produced the action).

C. Verifiability Taxonomy

We propose routing each action class to a gating strategy appropriate to how it can be checked:

- **Tier 1: Deterministically verifiable.** Actions with an objective, mechanically checkable correctness criterion (e.g., code that can be linted, compiled, or run against a test suite). Gating here is cheapest and most reliable, subject to the verifier-erosion risk discussed in Section V.
- **Tier 2: LLM-judgeable.** Actions without a deterministic check but with enough structure that a separate evaluation model can assess coherence, relevance, or policy compliance (e.g., a natural-language proposal or summary). Gating here uses one or more evaluator LLMs, as discussed below.
- **Tier 3: Weakly verifiable or open-ended.** Actions with no reliable automated check available (e.g., creative or strategic proposals with no ground truth). Gating here is necessarily probabilistic and should favor conservative escalation (block or flag for asynchronous human review) over confident automatic approval.

Fig. 2 summarizes this routing: each proposed action is classified by verifiability tier and directed to the gating strategy appropriate to that tier, with all three paths converging on a single transmit/block/escalate decision.

D. Self-Reported Quality Telemetry

Independent of the pass/fail gating decision for any single action, Autonomous Trust requires each agent to emit a continuous quality signal, a scalar or structured score reflecting its own estimate of recent output reliability, to an external monitoring service, in the same way a conventional application emits latency, error-rate, or resource-utilization metrics. This telemetry serves two purposes distinct from per-action gating:

it allows drift or degradation to be detected in aggregate even when no single action fails its gate, and it gives downstream agents or operators a signal to adjust how much weight to place on a given agent's output, independent of any individual transmitted action.

E. Panel-Based Adjudication for Tiers 2 and 3

For actions without a deterministic check, Autonomous Trust proposes an ensemble of independent evaluator LLMs voting on acceptability, rather than a single evaluator. A single LLM judge is subject to the same class of failure as the generating model; a panel, if its members are sufficiently diverse in model family, prompting, or training lineage, reduces the risk that a single blind spot silently approves an unsafe action. This is treated explicitly as a mitigation, not a solution: correlated failure across panel members remains possible and is discussed further in Section VII.

V. Design Analysis: The Gameable Verifier Problem

A specific risk to Tier 1 gating deserves separate treatment. In a system where agents not only produce actions but also author or evolve the checks used to validate those actions (for example, an LLM-backed coding agent that writes and updates its own test suite as a codebase evolves), an optimization pressure exists toward checks that are easy to satisfy rather than checks that are correct. A test suite that always passes maximizes measured success without improving actual output quality, and can mask its decline. This is a domain-specific instance of the broader specification-gaming problem [6], [7], made more acute in agent systems because the verifier is not a fixed external artifact but part of the same evolving system it is meant to constrain.

Candidate mitigations include:

- **Verifier provenance tracking.** Recording who (which agent, at what time) authored or last modified a given check, so that changes to verification logic are auditable independently of changes to the code or content they verify.
- **Held-out or adversarial verification.** Periodically validating outputs against checks the generating agent did not author and cannot modify, to detect divergence between self-authored and independent verification results.
- **Diversity requirements for panels.** Requiring that Tier 2/3 evaluator panels draw from model families or configurations distinct from the generating agent, reducing the chance that a shared blind spot in the underlying model class silently approves degraded output.
- **Periodic external re-validation.** Scheduling independent, higher-cost review (potentially human) of a sampled subset of gated actions, to calibrate whether the automated gate's pass rate reflects genuine quality or verifier erosion.

None of these mitigations is presented as a complete solution; each trades cost, latency, or architectural complexity for a reduction in a specific failure mode, and their relative effectiveness is an open empirical question (Section IX).

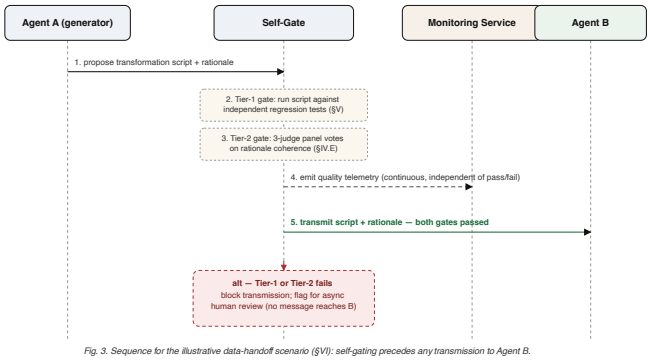

Fig. 3. Sequence for the illustrative data-handoff scenario (§VI): self-gating precedes any transmission to Agent B.

Fig. 3. Sequence for the illustrative data-handoff scenario: self-gating precedes any transmission to Agent B.

VI. Illustrative Scenario

Consider two autonomous agents, A (a data-preparation agent) and B (a downstream analytics agent operated by a different system), exchanging a proposed dataset transformation as part of an automated pipeline. Agent A generates a transformation script (Tier 1: code, checkable via linting and test execution) and an accompanying natural-language rationale for a schema change (Tier 2: LLM-judgeable).

Under Autonomous Trust, A runs its self-gate before transmitting either artifact. The transformation script is compiled and run against an independently maintained regression test set, which mitigates the gameable-verifier risk described in Section V. The rationale is submitted to a three-member evaluator panel drawn from a model family distinct from A's generator, which votes on whether the rationale is coherent and consistent with the script's actual behavior.

Concurrently, A emits a quality-telemetry score reflecting its confidence in this output, contributing to a rolling reliability signal that agent B, or a monitoring operator, can consult independent of this single transaction. Only after both gates pass does A transmit the artifacts to B; a Tier 1 or Tier 2 failure instead routes the action to a blocked-and-flagged state for asynchronous review, rather than reaching B at all. Fig. 3 depicts this exchange as a sequence diagram.

VII. Discussion, Limitations, and Threats to Validity

Autonomous Trust is presented in this paper as a conceptual and architectural framework, not as a validated empirical result, and several limitations should be stated directly.

1) Panel-based adjudication mitigates but does not eliminate correlated judge failure. If evaluator models share training data, architecture, or systematic blind spots with the generating model or with each other, a panel's apparent independence may be illusory.
2) Gating necessarily introduces latency and compute cost, and the framework does not yet specify where a given deployment should sit on that cost/reliability curve; this is an empirical, application-specific question.
3) For Tier 3 actions in particular, "trustworthy" itself may not have a single objective definition, and the

framework's reliance on conservative escalation for this tier is a design default, not a guarantee of correctness.

4) The framework assumes the evaluation layer can be kept architecturally and causally separate from the generating agent. In practice, shared infrastructure, shared training data, or shared operator incentives could erode that separation over time, a variant of the verifier-erosion problem discussed in Section V.

VIII. Preliminary Empirical Validation

The framework as presented so far is architectural. To give its central claims an initial empirical anchor, we ran two small, single-session pilot studies: a Tier-1 self-gating pilot on a code-generation task, and a Tier-2 panel-adjudication pilot on rationale/action consistency checking. Both pilots use models from the Claude family as the generating and evaluating agents. These are pilots, not benchmarks: they are undersized for statistical precision, and their purpose is to test whether the framework's mechanisms behave as described on at least one concrete task, not to produce generalizable performance figures. All problem specifications, hidden tests, eval-set items, judge prompts, and per-run outputs are retained as supplementary material for reproducibility.

A. Tier-1 Pilot: Self-Gating on a Code-Generation Task

We constructed an original 20-function Python benchmark (not drawn from existing suites such as HumanEval or MBPP, to avoid training-data contamination concerns) spanning common bug-prone patterns: empty-input handling, tie-breaking rules, off-by-one boundaries, wraparound arithmetic, and order preservation. Each function has a natural-language specification and a hidden pytest suite (70 assertions total, validated against a hand-written reference implementation, which passed all 70 before the pilot began). A generating agent was given only the specifications, with no visibility into the hidden tests, and asked to implement all 20 functions in a single pass. The Tier-1 self-gate is the hidden test suite itself, run against the submission prior to "transmission."

We ran four independent blind generations, each a freshly instantiated agent with no memory of prior runs. In the first run, the generated module used Python 3.10-only return-type syntax (`tuple[int, int] | None`), which raised a `TypeError` at import time in this pilot's Python 3.9 environment. This is a whole-module failure: the gate blocked the entire deliverable, all 20 functions, before any of it could be transmitted, since the code could not even be loaded in the target environment. Applying one semantically neutral compatibility shim (`from __future__ import annotations`, which defers annotation evaluation and changes no runtime behavior) let us also test the same submission at the function level: all 20 functions then passed all 70 hidden tests. After observing this failure, we added one clarifying line to the shared specification (target Python 3.9, avoid the newer union-type syntax) and ran three further independent blind generations against the revised specification. All three produced 3.9-compatible code on the first attempt, and all 20 functions in each run passed all 70 hidden tests, 60 of 60 function-level results correct across these three runs with zero gate failures.

Aggregating across all four runs (80 function-implementation attempts), logic-level correctness was 80 of 80 once environment compatibility was controlled for, but one of four runs (25%), taken as originally generated, would have shipped a completely non-functional deliverable to a downstream consumer had it not been gated, a defect invisible on a read-through of the code (it is idiomatic and would run correctly in a newer Python environment) and detectable only by actually executing the check. Gate overhead was negligible: the full 70-assertion suite ran in well under 0.1 seconds per run.

This result is consistent with the framework's non-stationarity claim (C2) in a narrow sense: one of four nominally identical invocations of the same generation process introduced a defect the other three did not. The observed defect class, an environment/version incompatibility, is narrower than the unsafe-or-incoherent outputs the framework is chiefly concerned with, and the pilot is too small to estimate a defect rate with any precision. We observed no pure logic bugs (0 of 80 function implementations were behaviorally incorrect once executable), which we read as a ceiling effect of a moderate-difficulty, well-specified benchmark and a capable generating model rather than evidence that gating's logic-level value is generally small; RQ1 below calls for testing this across harder and more ambiguous task classes.

B. Tier-2 Pilot: Single-Judge vs. Panel Adjudication

We constructed 20 items in the format of the illustrative scenario in Section VI: a short description of a concrete data-pipeline transformation and an accompanying natural-language rationale, as a sending agent might produce them together prior to transmission. Ten items have an accurate, internally consistent rationale; ten have a rationale containing one subtle factual or logical inconsistency with the stated transformation (for example, an inverted unit-conversion factor, an inner join described as a left join, the wrong partition granularity, or drop-versus-retain behavior stated backwards). Evaluator agents saw only the transformation/rationale pairs, not the ground-truth labels, and independently accepted or rejected each item.

We compared a single evaluator (Claude Sonnet) against a three-member panel (Claude Haiku, Sonnet, and Opus, majority vote): a within-vendor approximation of the model-family diversity that Sections IV-E and V call for, since we did not have access to models from other vendors for this pilot. The single Sonnet judge achieved 10 of 10 detections of the flawed items with zero false positives (20 of 20 correct). Each individual panel member, including the smallest model tier, Haiku, independently also scored 20 of 20 correct, with zero disagreement among the three judges on any item, so the panel's majority-vote result was identical to the single judge's.

This pilot does not show a measurable advantage for panel adjudication, because the eval set produced a ceiling effect: even the lightest-weight judge caught every inconsistency we constructed. That is itself informative for the cost/benefit

question raised in Section VII: for Tier-2 items in this difficulty range, a single lightweight judge may be sufficient, and panel overhead may not be justified. It does not test the framework's central concern about panel adjudication, correlated blind spots on cases genuinely hard enough to fool at least one competent judge, since none of our constructed inconsistencies were subtle enough to do so. RQ3 below is revised accordingly to call for an adversarially constructed eval set, calibrated against a single-judge baseline's actual failure cases, so that any panel value-add can be measured rather than assumed.

C. Limitations of This Validation

Both pilots are single-session and small-N, and use only within-vendor (Claude-family) models for generation and judging, which cannot speak to the cross-vendor correlated-failure risk the framework treats as a first-class concern (Sections IV-E, VII). The Tier-1 benchmark is self-authored rather than drawn from an established suite, and its difficulty was calibrated to produce a legible signal rather than to match the distribution of real-world A2A code-generation tasks. The Tier-2 eval set produced a ceiling effect and so does not yet test panel adjudication under genuinely contested cases. We report these results as an initial anchor for the framework's central claims, not as a benchmark result to be compared against other systems; Section IX outlines the larger-scale, harder, and cross-vendor evaluation this pilot motivates but does not substitute for.

IX. Future Work

Building on the pilots in Section VIII, we outline an empirical evaluation plan to test the framework's remaining central claims:

1) **RQ1.** Does a self-gating evaluator, applied before transmission, measurably reduce the rate of unsafe or incoherent outbound actions relative to an ungated baseline, across Tier 1–3 task classes and across task difficulty levels harder than the pilot benchmark in Section VIII-A, where logic-level (not only environment-level) defects are more likely to occur?
2) **RQ2.** In a self-evolving agent system where the agent also authors its own Tier 1 verification logic, how quickly and how far does effective verifier reliability degrade, and which mitigations from Section V most effectively slow that degradation?
3) **RQ3.** Does panel-based adjudication measurably outperform single-judge gating on Tier 2/3 tasks that are adversarially constructed to be difficult enough that at least a single judge fails, in terms of unsafe-action detection rate and false-positive rate, and at what marginal compute and latency cost per panel member added? Does the answer change when panel members are drawn from genuinely distinct model vendors rather than one vendor's model family, as in the Section VIII-B pilot?
4) **RQ4.** What is a minimal, domain-general schema for self-reported quality telemetry that a receiving agent or monitoring system can consume without per-domain customization, and does aggregate telemetry drift correlate with, and precede, individually gated failures?

Proposed methodology includes constructing larger benchmark task suites spanning the three verifiability tiers at multiple difficulty levels, instrumenting gate decisions and telemetry against ground-truth or human-adjudicated labels of actual output quality, and measuring detection rate, false-positive rate, and cost/latency overhead as the primary comparison metrics across gating configurations (no gate, single judge, panel of varying size, vendor diversity, and configuration).

X. Conclusion

As LLM-backed autonomous agents increasingly communicate directly with one another at a scale that precludes per-action human review, trust can no longer be modeled on human-style accumulated reputation, nor can it be delegated to the generating model itself. This paper has proposed Autonomous Trust, a framework in which trustworthiness is engineered as a property of the agent as a system rather than of the model alone. At its center is a pre-transmission self-gate, external to the generating model, that routes each action to a gating strategy appropriate to its verifiability, exposes continuous self-reported quality telemetry analogous to application health metrics, and accounts explicitly for the risk that automated verifiers can be degraded by the same agents they are meant to constrain. Along the way, we formulated the problem, laid out the framework's design principles, analyzed the gameable-verifier failure mode, reported preliminary empirical pilots on Tier-1 self-gating and Tier-2 panel adjudication, and outlined the larger empirical plan those pilots motivate. As agent-to-agent communication moves from research prototype toward infrastructure, we argue that self-gating evaluation is not an optional safety feature but a precondition for deploying such a system without a human reviewing every exchange.

Data and Code Availability

The problem specifications, hidden test suites, reference implementation, per-run generated code, panel eval-set items and labels, judge prompts, and raw judge verdicts for both pilots in Section VIII are retained alongside this manuscript's source materials and are available as supplementary material upon request or in an accompanying repository.

References

[1] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[2] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

[3] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *Proc. 36th Annual ACM Symp. on User Interface Software and Technology (UIST)*, 2023.

[4] J. Sabater and C. Sierra, "Review on Computational Trust and Reputation Models," *Artificial Intelligence Review*, vol. 24, no. 1, pp. 33–60, 2005.

[5] L. Zheng *et al.*, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[6] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," *arXiv preprint arXiv:1606.06565*, 2016.

[7] J. Skalse, N. H. R. Howe, D. Krasheninnikov, and D. Krueger, "Defining and Characterizing Reward Hacking," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
[8] A. Madaan *et al.*, "Self-Refine: Iterative Refinement with Self-Feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
[9] Y. Bai *et al.*, "Constitutional AI: Harmlessness from AI Feedback," *arXiv preprint arXiv:2212.08073*, 2022.