FROM ANOMALIES TO FAILURES: CONSTRUCTING CAUSAL ERROR GRAPHS FOR AGENTIC TRACE DIAGNOSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-driven agents are increasingly deployed in complex applications, where long agentic traces make failures difficult to diagnose. Existing trace diagnosis methods often conflate anomalies, errors, and failures, making diagnostic targets ambiguous; they also lack structured modeling of how causally relevant errors propagate and amplify into final task failures, resulting in unreliable failure attribution. To address these problems, we propose **CEG-Agent**, a tool-augmented agentic framework for causal diagnosis of agentic traces. Specifically, CEG-Agent introduces an explicit taxonomy of anomalies, errors, and failures, and constructs **Causal Error Graphs** (CEGs), a unified typed representation that links execution events, diagnostic nodes, and failure outcomes through causal relations. To evaluate causal trace diagnosis, we further construct **CEG-Bench**, a fully agent-annotated benchmark with high-confidence, consensus-derived CEG annotations obtained through an **Adversarial Agentic Adjudication Protocol** (AAAP). We validate the resulting annotations against an expert-curated human gold set, which shows close agreement with the automatic annotations. Experiments on CEG-Bench demonstrate that CEG-Agent achieves state-of-the-art performance under both semantically relaxed and structurally exact evaluation criteria. Our code is publicly available [1].

1 INTRODUCTION

In recent years, LLM-driven agents have been increasingly deployed in complex applications such as deep research (Wei et al., 2025), function calling (Barres et al., 2025), and code generation (Jimenez et al., 2024), where executions often unfold into long agentic traces with multiple intermediate steps and interactions. While effective, these long traces are prone to diverse failures (Yang et al., 2026). Such failures often arise from causally relevant process-level errors that gradually propagate and amplify across the trace, rather than from surface anomalies or isolated local errors. This highlights the need for automatically diagnosing failures in agentic traces and their underlying causal errors to support agent debugging and improvement.

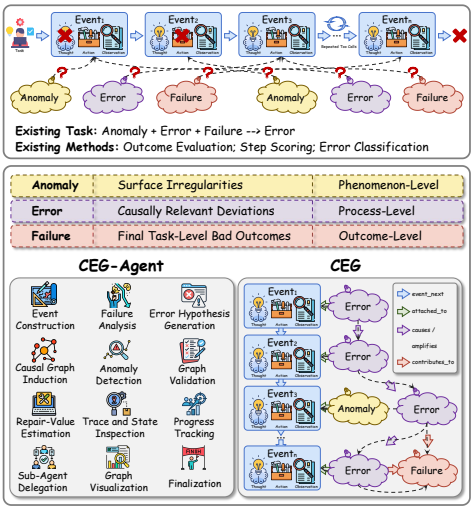


Figure 1: Comparison between prior paradigm and our method for agentic trace diagnosis.

However, existing methods for agentic trace diagnosis still suffer from two fundamental limitations. First, they often collapse heterogeneous diagnostic signals in agentic traces under the generic label of **error**. As shown in Figure 1, a failed trace may exhibit (1) surface irregularities, such as *repeated tool calls* or *unnecessarily verbose reasoning*; (2) causally relevant deviations, such as *missing*

[1] https://anonymous.4open.science/r/CEG-Agent-8888/

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.

constraints or *improper tool use*; and (3) final task-level bad outcomes, such as *incorrect answers* or *incomplete tasks*. These heterogeneous signals naturally call for a distinction among **anomalies**, **errors**, and **failures**, which correspond to phenomenon-level irregularities, process-level deviations, and outcome-level breakdowns, respectively. Conflating these notions as generic errors obscures their distinct diagnostic roles and optimization targets, making it difficult to formulate trace diagnosis as a well-defined problem and to provide consistent and effective supervision signals. Second, existing approaches rarely capture the causal structure of errors within a trace. They mostly rely on flat error classification (Li et al., 2026) or step scoring (Yang et al., 2025), lacking a structured account of how errors propagate, amplify, and eventually lead to failures. As a result, diagnosis often struggles to identify the key causes that truly drive the failure and the propagation paths where they take effect. This in turn weakens failure attribution and hinders debugging and optimization, since developers cannot confidently determine where to intervene first or which deviations are most worth correcting.

To address these limitations, we introduce **CEG-Agent**, a tool-augmented agentic framework for causal diagnosis of agentic traces. CEG-Agent starts from an explicit diagnostic taxonomy that separates **anomalies**, **errors**, and **failures**: anomalies capture phenomenon-level irregularities, errors capture causally relevant process-level deviations, and failures capture outcome-level task breakdowns. Based on this taxonomy, we define the **Causal Error Graph** (CEG) as a unified typed representation over execution events, anomalies, errors, and failures, with causal edges that specify where errors occur, how they propagate or amplify one another, and how they contribute to task-level failures. To automatically construct such graphs, CEG-Agent composes a set of core diagnosis tools for event construction, failure analysis, error hypothesis generation, causal graph induction, anomaly detection, graph validation, and repair-value estimation, together with auxiliary tools for trace and state inspection, progress tracking, sub-agent delegation, graph visualization, and finalization. The resulting system produces visualizable causal error graphs with repair-value estimates, helping developers understand failure formation and prioritize interventions for agent improvement.

To evaluate causal trace diagnosis, we construct **CEG-Bench**, a fully agent-annotated benchmark with high-confidence, consensus-derived CEG annotations. Following TraceSIR (Yang et al., 2026), we build a candidate pool of 150 failed GLM-4.6 traces from BrowseComp (Wei et al., 2025), Tau2Bench (Barres et al., 2025), and SWE-bench (Jimenez et al., 2024), with 50 cases from each benchmark. Given the complexity of agentic traces and the difficulty of manual causal annotation, we design an **Adversarial Agentic Adjudication Protocol** (AAAP) for CEG annotation. Specifically, Claude Code powered by Claude-Opus-4.7 first drafts an initial CEG for each candidate trace. The draft is then revised using diagnostic feedback from the open-source TraceSIR analyzer powered by Claude-Opus-4.6, which serves as an independent error reviewer. Next, two heterogeneous agentic annotators, Claude Code with Claude-Opus-4.7 and Codex with GPT-5.5, iteratively cross-examine and refine each graph until both reach consensus and raise no remaining objections to the CEG. We finally retain the 100 cases whose CEGs are unanimously accepted by these two strong agentic frameworks powered by frontier LLMs, yielding a final set of 100 consensus traces.

We validate these automatically constructed annotations against **CEG-Bench-Gold**, an expert-curated human reference set over the same 100 traces. The AAAP annotations show strong agreement with human judgments, achieving an overall $\mathrm{CEG\text{-}Sim}$ of 71.5 and high agreement on core diagnostic dimensions. This agreement provides empirical support for AAAP as a scalable and reliable annotation protocol for causal trace diagnosis. Furthermore, experiments on **CEG-Bench** show that **CEG-Agent** outperforms strong agentic baselines, including Claude Code and Codex, with average gains of $+9.0$ points under **semantically relaxed** CEG evaluation and $+4.9$ points under **structurally exact** CEG evaluation. Fine-grained analyses show that these gains are concentrated on the core structural capabilities required for causal diagnosis, such as distinguishing anomalies from causally relevant errors and constructing error propagation chains. Cross-backbone results further show that CEG-Agent exhibits different diagnostic profiles when driven by different LLMs, yet consistently provides explicit, structured, and causally grounded diagnoses for debugging and improvement.

2 Related Work

LLM-driven agents are increasingly evaluated on long-horizon tasks involving web browsing, deep research, function calling, tool use, and software engineering. Benchmarks such as BrowseComp (Wei et al., 2025), Tau2Bench (Barres et al., 2025), SWE-bench (Jimenez et al., 2024), and broader

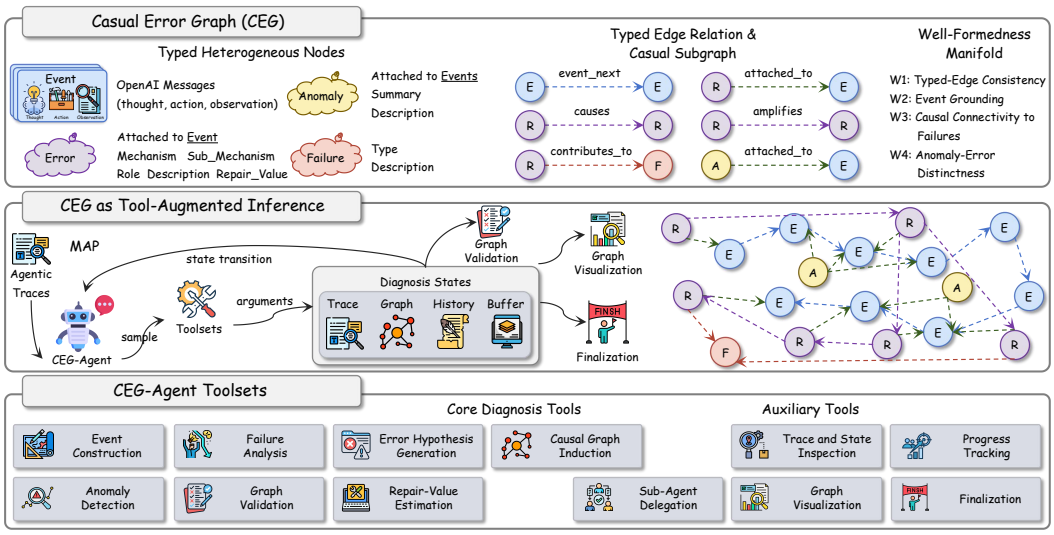


Figure 2: The overall design of CEG-Agent.

agent benchmarks (Liu et al., 2025; Zhou et al., 2024; Deng et al., 2023) have advanced evaluation of task completion and tool-use capabilities, while process-oriented evaluations and LLM-based judges further assess intermediate reasoning, actions, and step-level quality (Zheng et al., 2023; Lightman et al., 2023). Recent trace-level diagnosis methods, especially TraceSIR (Yang et al., 2026), begin to analyze agentic execution traces through structured reports, trajectory feedback, or error detection, and are related to work on hallucination detection, reasoning errors, tool-use failures, and software fault diagnosis (Schick et al., 2023; Li et al., 2026; Yang et al., 2025). However, these methods often focus on final outcomes, local step quality, or flat error categories, and provide limited support for distinguishing surface anomalies, causally relevant process-level errors, and outcome-level failures, or for explaining how errors propagate and amplify into failures. Our work is also connected to causal reasoning, graph-based diagnosis, provenance modeling, and root cause analysis, where causal graphs and dependency paths are used to explain failures (Qian et al., 2025; Ghosh et al., 2024; Hou et al., 2025; Yamin et al., 2025). CEG adapts these ideas to LLM-driven agents by grounding process-level errors in execution events and linking them to failures through typed causal relations. Finally, CEG-Agent builds on tool-augmented and critic-based LLM frameworks, including ReAct-style tool use, self-refinement, reflection, and multi-agent collaboration (Yao et al., 2023; Madaan et al., 2023; Shinn et al., 2023), but specializes them for structured causal trace diagnosis, producing visualizable CEGs with repair-value estimates for debugging and agent improvement.

3 METHOD

We present **CEG-Agent**, a tool-augmented agentic framework that formulates agentic trace diagnosis as constrained inference over typed heterogeneous causal graphs, as shown in Figure 2. Specifically, we first formalize causal trace diagnosis as a constrained *maximum a posteriori* (MAP) problem, where the target is a schema-valid graph explaining task-level failures and their underlying causally relevant errors from an agentic trace (§3.1). We then define the **Causal Error Graph** (CEG) as a heterogeneous multi-relational graph that separates anomalies, errors, and failures, grounds process-level deviations in execution events, and encodes error propagation and failure attribution through typed edges (§3.2). Finally, we instantiate CEG construction as a tool-augmented inference procedure, where a diagnostic agent incrementally constructs, validates, and revises the graph through schema-respecting state transitions (§3.3). Semantic refinement, graph validation, and counterfactual repair-value estimation are also exposed as registered tools in this process. This section focuses on the core modeling and inference principles, while implementation details are given in Appendix A.

3.1 PROBLEM FORMULATION

An *agentic trace* is a pair $T = (\tau, \mathcal{M})$ where τ is the user task specification and $\mathcal{M} = (m_1, \ldots, m_N)$ is a sequence of OpenAI-style messages spanning assistant reasoning, tool calls, and observations.

Trace diagnosis is the problem of inferring a *causal error graph* $G \in \mathcal{G}^\star(T)$, where $\mathcal{G}^\star(T)$ denotes the schema-valid set of graphs grounded in T. We model this problem as constrained MAP estimation:

$$G^\star \;=\; \arg \max_{G \in \mathcal{G}^\star(T)} \; \log p_\theta(G \mid T), \tag{1}$$

where p_θ is implicitly defined by a tool-augmented agentic policy (§3.3). The constraint $G \in \mathcal{G}^\star(T)$ enforces three diagnostic requirements that distinguish CEG inference from flat error classification: (R1) typed separation of phenomenon-level anomalies, process-level errors, and outcome-level failures; (R2) grounding of each diagnostic node in a concrete execution event; (R3) explicit causal structure linking errors to the failure they explain.

3.2 Causal Error Graph

We formalize a CEG as a typed heterogeneous multi-relational graph $G = \big(V,\, E,\, \tau_V,\, \tau_E,\, \mathbf{x}\big)$, with a node-type map $\tau_V : V \to \mathcal{T}_V$, an edge-type map $\tau_E : E \to \mathcal{L}$, and an attribute map $\mathbf{x}$ that assigns each node a type-specific attribute tuple.

Heterogeneous node types. The node-type alphabet is $\mathcal{T}_V = \{\mathrm{E}, \mathrm{A}, \mathrm{R}, \mathrm{F}\}$, inducing disjoint heterogeneous subsets $V = V_E \,\dot\cup\, V_A \,\dot\cup\, V_R \,\dot\cup\, V_F$, where V_E contains *event* nodes, V_A contains *anomaly* nodes, V_R contains *error* nodes, and V_F contains *failure* nodes. Attribute tuples are $\mathbf{x}(e) = (\textit{thought}, \textit{action}, \textit{observation})$ for $e \in V_E$; $\mathbf{x}(r) = (\varepsilon(r), \mu(r), \nu(r), \rho(r), d(r))$ for $r \in V_R$ with grounding event $\varepsilon(r) \in V_E$, coarse mechanism $\mu(r) \in \mathrm{Mech}$, fine sub-mechanism $\nu(r)$, structural role $\rho(r) \in \{\textsc{root}, \textsc{propagated}, \textsc{amplification}\}$, and a trace-grounded description $d(r)$; and $\mathbf{x}(f) = (\theta(f), d(f))$ for $f \in V_F$ with failure type $\theta(f)$. The mechanism (8 values), role (3 values), and failure-type (6 values) vocabularies are *grounded* in large-scale human error analyses, and *extensible*, with an explicit *other* bucket as a completeness escape hatch; empirically all mechanisms and all failure types are populated in CEG-Bench, with *other* absorbing only 3 residual failures (B.2). V_A may be grounded in one or more events, denoted by $\mathbf{x}(a) = (\boldsymbol{\varepsilon}(a) \subseteq V_E, \nu(a), d(a))$.

Typed edge relation. The edge-type alphabet is $\mathcal{L} = \{\textsc{event-next}, \textsc{attached-to}, \textsc{causes}, \textsc{amplifies}, \textsc{contributes-to}\}$. To enforce heterogeneity, every edge respects a legal endpoint relation $\Omega \subset \mathcal{T}_V \times \mathcal{L} \times \mathcal{T}_V$:

$$\forall (u, \ell, v) \in E : \; \big(\tau_V(u),\, \ell,\, \tau_V(v)\big) \in \Omega, \tag{2}$$

where Ω is the valid type tuples: $(\mathrm{E}, \textsc{event-next}, \mathrm{E})$, $(\mathrm{R}, \textsc{attached-to}, \mathrm{E})$, $(\mathrm{R}, \textsc{causes}, \mathrm{R})$, $(\mathrm{R}, \textsc{amplifies}, \mathrm{R})$, and $(\mathrm{R}, \textsc{contributes-to}, \mathrm{F})$. Anomalies in V_A may also be grounded in one or more events via ATTACHED-TO edges. Since they are surfaced for diagnostic completeness but excluded from causal attribution, we do not emphasize these edges in the following discussion.

Causal subgraph and reachability. Let $\mathcal{L}_C = \{\textsc{causes}, \textsc{amplifies}, \textsc{contributes-to}\}$ denote the causal sub-alphabet, and let $G_C = (V_R \cup V_F, E_C)$, where $E_C = \{(u, \ell, v) \in E \mid \ell \in \mathcal{L}_C\}$, be the *causal subgraph* obtained by projecting away temporal and grounding edges. For an error $r \in V_R$, we define the failure reachability set under G_C as $\mathrm{Reach}_G(r) = \{f \in V_F \mid r \xrightarrow{\mathcal{L}_C^*} f\}$, where $\xrightarrow{\mathcal{L}_C^*}$ denotes any directed path using only causal edge types.

Well-formedness manifold. The manifold $\mathcal{G}^\star(T)$, which acts as the feasible region for the diagnostic optimization problem in Eq. (1), is the set of graphs satisfying four predicates. W_1: typed-edge consistency, i.e., Eq. (2); W_2: event grounding, requiring $\forall r \in V_R, \exists e \in V_E : (r, \textsc{attached-to}, e) \in E$; W_3: causal connectivity to failures, ensuring $\forall r \in V_R, \mathrm{Reach}_G(r) \neq \varnothing$; and W_4: anomaly–error distinctness, $\forall a \in V_A, \forall r \in V_R : a \not\equiv r$, forbidding textually equivalent diagnostic content.

3.3 CEG as Tool-Augmented Inference

We approximate the MAP objective in Eq. (1) by a constrained tool-augmented inference procedure. A top-level LLM policy maintains a diagnosis state and incrementally constructs G by invoking specialized tools, each of which writes to one component of the state under the schema constraints of $\mathcal{G}^\star(T)$. Let $S_t = (T,\, G_t,\, H_t,\, \mathcal{Q}_t)$ denote the diagnosis state at iteration t, comprising the trace T, the current partial graph G_t, the conversation history H_t, and an auxiliary work-tracking buffer

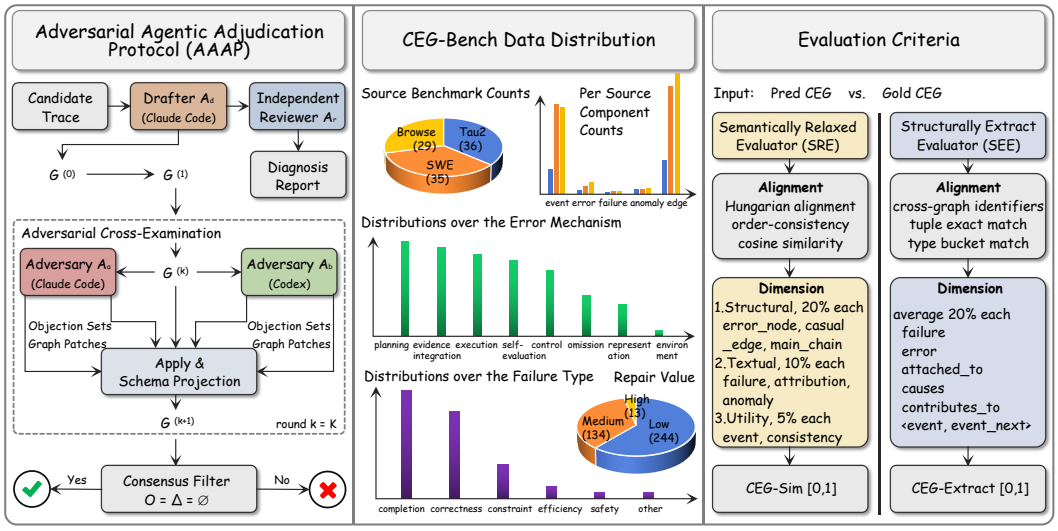


Figure 3: Construction and evaluation overview of CEG-Bench.

$\mathcal{Q}_t$ containing todo list and sub-agent reports. At every step, the agent samples a tool u_t from the registered tool set $\mathcal{U}$ with its execution arguments $\mathbf{a}_t$, and induces a state transition. A distinguished terminal action $u_T = \texttt{finalize}$ halts the trajectory; otherwise the loop runs for at most K iterations.

$$\big(u_t,\, \mathbf{a}_t\big) \sim \pi_\theta(\,\cdot \mid S_t, \mathcal{U}), \quad S_{t+1} = \Phi_{u_t}\big(S_t,\, \mathbf{a}_t\big). \tag{3}$$

Specifically, the tool set $\mathcal{U}$ of CEG-Agent is designed to mirror the diagnostic structure presented in §3.2. It consists of seven *core diagnosis tools* that construct, validate, and value the components of the CEG, and five *auxiliary tools* that support the diagnostic process without directly writing graph components. We outline high-level role of each tool below, deferring implementation details to §A.

3.3.1 Core Diagnosis Tools

Event construction segments the normalized trace into ordered thought–action–observation units (Yang et al., 2026), yielding the event set V_E that provides the temporal scaffold against which every subsequent diagnostic node is grounded.

Failure analysis identifies the outcome-level failures V_F against which the trace is to be explained, typically yielding a single primary failure $f^\star$, or at most three, which gives downstream error generation a concrete attribution target.

Error hypothesis generation proposes a compact set of causally relevant process-level errors V_R, each grounded in a specific event via $\varepsilon(r)$ and enriched with a mechanism $\mu(r)$, structural role $\rho(r)$, and trace-evidenced description $d(r)$.

Causal graph induction adds the typed causal edges (CAUSES, AMPLIFIES, CONTRIBUTES-TO) that explain how errors propagate and accumulate into failures. Temporal (EVENT-NEXT) and grounding (ATTACHED-TO) edges are populated deterministically from event order and from $\varepsilon(r)$.

Anomaly detection surfaces phenomenon-level irregularities V_A that are observed but explicitly excluded from causal attribution.

Graph validation checks G_t against the well-formedness manifold $\mathcal{G}^\star(T)$ and returns an issue set $\mathcal{I}(G_t)$. Structurally repairable schema violations are handled by a deterministic projection operator $\Pi : \mathcal{G} \to \mathcal{G}^\star(T)$, while semantic inconsistencies are surfaced to an LLM critic, which emits a compact graph patch for iterative refinement. As a final safeguard, CEG-Agent projects the terminal graph G_T to $G^\star = \Pi(G_T)$, ensuring that the released graph satisfies $\mathrm{W}_1 \wedge \mathrm{W}_2 \wedge \mathrm{W}_3$ regardless of policy realization. Further details on the checks, the Π, and the critic loop are provided in §A.4 and §A.5.

Repair-value estimation assigns each error $r \in V_R$ a counterfactual repair value $\rho^\sharp(r) \in \{\text{HIGH}, \text{MEDIUM}, \text{LOW}\}$ that scores whether fixing r alone would prevent or substantially mitigate $f^\star$. The full counterfactual functional is given in §A.6.

3.3.2 AUXILIARY TOOLS

Trace and state inspection provides read-only access to the normalized trace and the current graph G_t, allowing the agent to re-examine evidence when an earlier decision needs to be revisited.

Progress tracking maintains a structured todo buffer inside $\mathcal{Q}_t$, giving the agent an explicit working memory for multi-step refinement, analogous to the TodoWrite mechanism in Claude Code.

Sub-agent delegation dispatches a focused read-only sub-agent to investigate a narrow question and returns a structured report that the parent agent may act on, while keeping the sub-agent isolated from the parent state. Further details are given in §A.7.

Graph visualization serializes the CEG into DOT, HTML, or SVG, enabling the agent and users to inspect error-propagation chains visually.

Finalization is the terminal action that exits the inference loop, after which the schema projection Π is applied to guarantee a well-formed output.

4 CEG-BENCH

4.1 TRACE COLLECTION

We construct **CEG-Bench**, a benchmark of 100 agentic execution traces annotated with consensus Causal Error Graphs. Following Yang et al. (2026), we sample 150 failed GLM-4.6 traces from three real-world agentic benchmarks: BrowseComp (Wei et al., 2025) for deep research, Tau2Bench (Barres et al., 2025) for function calling, and SWE-bench (Jimenez et al., 2024) for agentic coding, with 50 candidate traces from each source. All traces are kept in their original OpenAI-style message format, including assistant reasoning, tool calls, and observations. After AAAP, 100 consensus traces are retained for release: 29 from BrowseComp, 36 from Tau2Bench, and 35 from SWE-bench. The construction process is shown in Figure 3, and detailed analyses are provided in Appendix B.

4.2 BENCHMARK CONSTRUCTION

The complexity of long agentic traces and the difficulty of manually identifying causal structure make conventional annotation workflows impractical. We therefore design an **Adversarial Agentic Adjudication Protocol** (AAAP), in which four heterogeneous agentic annotators play asymmetric roles, i.e., one drafter, one independent reviewer, and two adversarial cross-examiners. A trace is retained only if both adversaries certify the resulting CEG. This asymmetry is deliberate: the drafter–reviewer pair supports constructive graph proposal and refinement, while the two-adversary phase iteratively stress-tests and revises the CEG before imposing a strict consensus filter.

Let $\mathcal{A}_d$ denote the drafter, $\mathcal{A}_r$ the independent reviewer, and $\mathcal{A}_a, \mathcal{A}_b$ two cross-examining adversaries. We instantiate $\mathcal{A}_d, \mathcal{A}_a$ as Claude Code powered by Claude-Opus-4.7, $\mathcal{A}_r$ as the open-source TraceSIR analyzer powered by Claude-Opus-4.6 (Yang et al., 2026), and $\mathcal{A}_b$ as Codex powered by GPT-5.5. The core principle of AAAP is that if a CEG is accepted by two adversarial agents built on frontier LLMs and state-of-the-art agentic coding frameworks, then it provides a credible consensus reference for causal trace diagnosis, despite the limitations of fully automated adjudication.

Specifically, AAAP proceeds in four stages: (1) **Draft.** The drafter produces an initial graph $G^{(0)} = \mathcal{A}_d(T)$, conditioned only on T. (2) **Independent review.** The reviewer independently analyzes the trace and emits a diagnostic report $\kappa^{(0)} = \mathcal{A}_r(T)$, identifying candidate errors from an additional perspective. The drafter incorporates this report to obtain $G^{(1)} = \mathcal{A}_d(T, G^{(0)}, \kappa^{(0)})$. (3) **Adversarial cross-examination.** For rounds $k = 1, \ldots, K$, the two adversaries independently inspect and revise $G^{(k)}$, returning objection sets and graph patches $(O_a^{(k)}, \Delta_a^{(k)}) = \mathcal{A}_a(T, G^{(k)})$ and $(O_b^{(k)}, \Delta_b^{(k)}) = \mathcal{A}_b(T, G^{(k)})$. The next graph is obtained by applying the proposed patches and projecting back to the schema, $G^{(k+1)} = \Pi\big(\mathrm{Apply}(G^{(k)}, \Delta_a^{(k)} \cup \Delta_b^{(k)})\big)$, where Π is the schema projection. (4) **Consensus filter.** The trace is retained iff there exists $k^\star \le K$ such that $O_a^{(k^\star)} = O_b^{(k^\star)} = \varnothing$ and $\Delta_a^{(k^\star)} = \Delta_b^{(k^\star)} = \varnothing$. The accepted annotation is $G^\star = \Pi(G^{(k^\star)})$; otherwise, the trace is discarded. The retained traces therefore come with CEG annotations that have passed iterative adversarial revision and dual-adversary consensus, providing high-quality graphs.

Backbone	System	SRE	Structural			Textual			Utility	
			error	edge	chain	fail.	attr.	anom.	event	cons.
Claude-Opus-4.7	Claude Code (raw)	45.8	39.5	37.1	38.8	44.7	**48.7**	44.7	78.1	100.0
Claude-Opus-4.7	Claude Code (+Process)	49.6	**49.2**	41.8	38.2	**47.7**	43.9	46.0	100.0	100.0
Claude-Opus-4.7	**CEG-Claude** (ours)	**51.6**	46.7	**42.4**	**51.0**	38.8	37.1	**60.0**	100.0	99.9
GPT-5.5	Codex (raw)	32.0	22.8	31.8	27.4	29.9	25.5	2.0	98.0	100.0
GPT-5.5	Codex (+Process)	32.9	27.0	34.5	29.5	21.5	23.5	1.7	100.0	100.0
GPT-5.5	**CEG-Codex** (ours)	**46.5**	**45.9**	**38.0**	**41.5**	**34.9**	**27.6**	**51.3**	100.0	100.0

Table 1: Semantically relaxed evaluation (**SRE**) on CEG-Bench.

Backbone	System	SEE	Dimensions				
			failure	error	attach	cause	contrib.
Claude-Opus-4.7	Claude Code (raw)	13.9	**56.1**	5.3	3.9	1.0	3.3
Claude-Opus-4.7	Claude Code (+Process)	19.3	47.3	18.5	16.3	1.1	**13.2**
Claude-Opus-4.7	**CEG-Claude** (ours)	**20.5**	52.1	**20.5**	**17.8**	**2.0**	9.9
GPT-5.5	Codex (raw)	11.7	43.8	5.6	6.0	0.0	3.1
GPT-5.5	Codex (+Process)	10.4	33.3	8.7	6.0	0.0	4.1
GPT-5.5	**CEG-Codex** (ours)	**16.9**	**48.3**	**15.2**	**13.0**	**1.5**	**6.3**

Table 2: Structurally exact evaluation (**SEE**) on CEG-Bench.

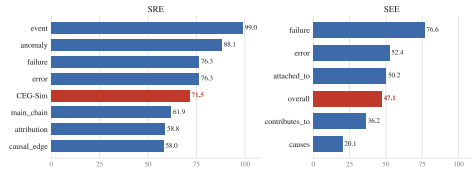


Figure 4: AAAP–human-gold agreement per SRE dimension over all 100 traces.

Human validation. To verify the reliability of AAAP annotations, we additionally invited eight domain experts to independently cross-annotate all 100 traces, yielding a human gold set, **CEG-Bench-Gold**. When evaluated using the metrics in §4.3, AAAP shows strong agreement with this human gold set, as illustrated in Figure 4. Specifically, it achieves an overall CEG-Sim of 71.5, with agreement scores of 76.3, 76.3, and 88.1 on errors, failures, and anomalies, respectively. Achieving this level of agreement on such a challenging causal annotation task is non-trivial. More importantly, high-quality human annotation of long causal traces is prohibitively expensive and unavailable in most real deployments, whereas AAAP is fully automated and now validated against expert judgments. Producing scalable and reliable annotations is therefore itself a core contribution of this work. We adopt the AAAP-constructed **CEG-Bench** as the primary reference throughout the paper, while using **CEG-Bench-Gold** to *audit* its reliability. The construction of CEG-Bench-Gold and per-system scores against the human gold set are provided in Appendix D; all other experiments are reported against the AAAP annotations.

4.3 Evaluation Criteria

A predicted CEG $\hat{G}$ is evaluated against the consensus annotation $G^\star \in \mathcal{G}^\star(T)$ using two complementary criteria. The **semantically relaxed** evaluator (SRE) aligns predicted and gold nodes via Hungarian bipartite matching, using a type-specific similarity that combines event correspondence, categorical agreement on $\mu(r), \rho(r), \theta(f)$, and text similarity (Chen et al., 2024) over $\nu(r), d(r)$. Here, $\mu(r)$, $\rho(r)$, and $\theta(f)$ are controlled categorical fields with 8, 3, and 6 predefined values, respectively, as shown in Figure 3, whereas $\nu(r)$ and $d(r)$ are free-text fields generated by the LLM. After alignment, predicted identifiers are translated into the gold space, and each dimension is scored by precision (P), recall (R), and F1. The resulting $\text{CEG-Sim}(\hat{G}, G^\star) \in [0, 1]$ is a tier-weighted aggregate over three categories: **structural** dimensions, `error_node`, `causal_edge`, and `main_chain`, weighted by 20% each; **textual** dimensions, `failure`, `attribution`, and `anomaly`, weighted by 10% each; and **utility** dimensions, `event` and `consistency`, weighted by 5% each. The **structurally exact** evaluator (SEE) performs no fuzzy matching, resolving errors by the exact key $(\varepsilon(r), \mu(r))$ and failures by the bucket $\theta(f)$. Edges are compared as multisets of typed tuples after translating predicted identifiers into the gold space, with edges incident to unmatched nodes counted as false

positives. The resulting $\mathrm{CEG\text{-}Exact}(\hat{G}, G^\star) \in [0,1]$ is the mean of five discriminative dimensions: `failure`, `error`, `attached_to`, `causes`, and `contributes_to`. Details are provided in Appendix C.

5 EXPERIMENTS

We evaluate **CEG-Agent** on **CEG-Bench**. For each backbone family, we compare three systems on the same 100 traces: a raw coding-agent baseline given the original trace and CEG schema (**raw**), a stronger baseline given deterministic event segmentation V_E and EVENT-NEXT edges (**+Process**), and **CEG-Agent** instantiated with the same backbone. We use Claude-Opus-4.7 for the Claude Code / CEG-Claude comparison and GPT-5.5 for the Codex / CEG-Codex comparison. Because no existing system, to our knowledge, emits typed causal error graphs under our schema, a strictly like-for-like external baseline does not yet exist; adapting a flat-classification or step-scoring method would itself require the schema-projection scaffolding we contribute. Our **+Process** baseline instead reproduces the deterministic event-segmentation stage that prior trace-diagnosis methods rely on. We report **SRE** ($\mathrm{CEG\text{-}Sim}$) and **SEE** ($\mathrm{CEG\text{-}Exact}$) from §4.3; higher is better, and all scores are percentages. CEG-Agent uses inference budget $K = 24$, critic budget $C = 2$, and temperature $= 1$, while baselines use their CLI defaults. Detailed experimental analyses are provided in Appendix E.

5.1 OVERALL PERFORMANCE

Tables 1 and 2 report the headline results. **CEG-Agent achieves the best total score under both SRE and SEE for both backbone families.** With Claude-Opus-4.7, CEG-Claude improves over the stronger Claude Code (+Process) baseline by $+2.0$ points in SRE and $+1.2$ points in SEE. With GPT-5.5, the gains are larger, i.e., CEG-Codex improves over Codex (+Process) by $+13.6$ points in SRE and $+6.5$ points in SEE. These gains are not merely due to better event parsing, since the +Process baselines already receive the deterministic event scaffold. The improvements therefore mainly reflect better causal diagnosis, graph construction, and refinement. The benefit is especially pronounced for GPT-5.5, suggesting that structured tool use and validation can compensate for weaker raw diagnostic behavior. Overall, CEG-Agent's anomaly–error–failure taxonomy and typed causal graph construction make long trace diagnosis more actionable and structurally faithful.

5.2 FINE-GRAINED ANALYSIS

The per-dimension results in Tables 1 and 2 localize the main sources of improvement. Both CEG-Agent variants lead their same-backbone baselines on the core structural dimensions, with the largest gains on SRE `main_chain`, `causal_edge`, `anomaly`, and SEE `error`, `attached_to`, and `causes`. The few apparent exceptions under Claude, including SRE `error`/`failure`/`attribution` and SEE `contributes_to`, are mainly due to gold-empty traces, where abstaining systems are mechanically favored, and textual-style mismatch between detailed predictions and compact gold descriptions (§E.5, §E.4). On the non-empty subset, CEG-Agent leads same-backbone baselines by $+5.6/+1.4$ points under Claude and $+13.0/+6.2$ points under GPT-5.5 on SRE/SEE, respectively, suggesting that the headline results understate its advantage on substantive causal diagnosis. Additional analyses in §E explain these gains. First, CEG-Agent better matches the scale of the gold graphs, producing $|\bar{V}_R| \approx 4.6$ errors and $|\bar{E}_C| \approx 6.9$ causal edges, close to the gold averages of 3.9 and 7.5, while baselines under-generate errors, anomalies, and causal edges (§E.6). Second, its largest recall gains occur on difficult mechanisms such as *representation*, *evidence integration*, and *planning*, as well as on PROPAGATED errors that require causal-chain reconstruction (§E.7). Third, the advantage is concentrated on medium-to-large traces that require sustained causal reconstruction, where incremental graph construction is most useful (§E.3). Repair-value calibration, cost and tool use, and visualization case studies are analyzed in §E.9, §E.12, and §F.1, respectively. Moreover, CEG construction remains challenging. Even the best system reaches only 51.6 SRE and 20.5 SEE, indicating strong agents still struggle to identify causally relevant errors and reconstruct their propagation chains.

5.3 CROSS-BACKBONE GENERALIZATION

Table 3 evaluates CEG-Agent with five LLM backbones under the same schema, tools, and budgets. CEG-Agent produces useful diagnostic graphs across all tested models: even the weakest SRE

backbone, Qwen3.7-Max, reaches 43.7, well above the strongest Codex baseline at 32.9. This suggests that the validator-guided construction loop provides a backbone-agnostic structural scaffold.

The SRE and SEE rankings also diverge beyond the top model. In particular, Qwen3.7-Max outperforms GPT-5.5 and Gemini-3.1-Pro on structurally exact reconstruction despite lower semantic similarity, indicating that fluent diagnostic descriptions and exact causal structure are distinct capabilities. The fixed framework further exposes backbone-specific profiles, such as Claude's overall strength, GLM-5.1's strong exact reconstruction, GPT-5.5's better semantic than structural behavior, Gemini's conservative contribution edges, and Qwen's weaker anomaly coverage. Together, these results show that CEG-Agent is not tied to a single LLM family and that CEG evaluation can serve as a discriminative probe of causal-diagnosis capability.

Backbone	**SRE**	**SEE**
Claude-Opus-4.7	**51.6**	**20.5**
GLM-5.1	46.7	19.6
GPT-5.5	46.5	16.9
Gemini-3.1-Pro	44.2	15.0
Qwen3.7-Max	43.7	18.7

Table 3: CEG-Agent composite SRE/SEE under five LLM backbones.

5.4 Closed-Loop Repair with CEG Feedback

Beyond diagnostic quality, we also probe whether CEGs can serve as actionable repair signals. Since all CEG-Bench traces are failed attempts, the original runs have zero success by construction. We re-attempt each task with a feedback hint derived from the AAAP CEG, comparing a content-free generic retry note with three compact CEG-based hints: the outcome-level failure, the causal errors, and priority errors, i.e., errors with high or medium repair value. Details are provided in Appendix F.

Table 4 shows that CEG is most effective when agent can act on it interactively. For example, on Tau2Bench, all CEG-derived hints outperform the generic retry, with the failure-level hint giving the largest gain. This suggests that explicitly naming the outcome-level failure provides a concrete correction target, while causal errors and repair-prioritized errors offer useful process-level guidance. The fact that structured CEG hints consistently win further indicates that the gains directly come from the diagnostic content of CEG.

Feedback	**Tau2**	**Browse**	**SWE**
generic	77.8	6.9	14.3
CEG: failure	**91.7**	10.3	**17.1**
CEG: errors	80.6	**17.2**	**17.1**
CEG: priority errors	83.3	**17.2**	8.6

Table 4: Closed-loop repair pass@1 (%).

The preferred granularity of feedback depends on the task. On BrowseComp, error-based hints are more useful than the failure-only hint, likely because the task requires revisiting evidence and reasoning steps rather than merely knowing that the final answer is wrong. However, the absolute success rate remains low, since many failed traces do not contain enough missing evidence for one-shot re-derivation. On SWE-bench, concise failure or error hints slightly improve over generic feedback, whereas priority-error filtering hurts. This suggests that repair values are not yet sufficiently calibrated for single-shot code patching: filtering to high-value or medium-value errors may omit contextual errors needed for a correct patch, or over-focus the model on an incomplete local fix. Overall, these results should be viewed as directional rather than conclusive, given the small per-source sample sizes. Still, they show that CEGs can provide useful repair signals beyond diagnosis, especially in interactive settings. We leave larger-scale closed-loop optimization to future work.

6 Conclusion

We presented **CEG-Agent**, a tool-augmented agentic framework for causal diagnosis of agentic execution traces. Instead of treating all diagnostic signals as generic errors, CEG-Agent separates anomalies, errors, and failures, and represents their relationships using **Causal Error Graphs** (CEGs), a typed graph structure that connects trace events, process-level causal errors, and outcome-level failures. To support systematic evaluation, we further introduced **CEG-Bench**, a benchmark of agentic traces annotated with consensus CEGs through an **Adversarial Agentic Adjudication Protocol**, and validated its annotations against the expert-curated **CEG-Bench-Gold**. Experiments under both **semantically relaxed** and **structurally exact** evaluation criteria show that CEG-Agent outperforms strong agentic baselines in constructing causal error graphs. By producing visualizable propagation structures and repair-oriented recommendations, CEG-Agent provides actionable diagnostic evidence for debugging agent executions and improving future agent behavior.

AI USE STATEMENT

In this work, we used generative AI tools for generating AI-assisted dataset annotations: the CEG annotations in **CEG-Bench** are constructed by agentic LLM systems through the Adversarial Agentic Adjudication Protocol (AAAP, §4.2), in which frontier LLMs draft, independently review, and adversarially cross-examine each Causal Error Graph annotation until dual-adversary consensus is reached. These AI-generated annotations, together with the deterministic repair values derived from them, are central artifacts of this paper. The CEG-Agent diagnoses and the closed-loop repair study are likewise produced by LLM-driven agents, as described in the corresponding sections. Additionally, we used generative AI tools to aid and polish the writing of the manuscript, improving grammar and clarity of author-drafted text. We did not use generative AI tools to prove mathematical claims or for the core research ideation of this work, and generative-AI-assisted retrieval or discovery of related work is not applicable here. We reviewed all AI-assisted outputs. The AI-generated CEG-Bench annotations were validated against an independent expert-curated human gold set, **CEG-Bench-Gold** (§D), and we report and analyze their close agreement with this human reference. All quantitative results were re-checked by the authors against the released evaluation code, and all AI-polished text was verified for accuracy and faithfulness to the intended meaning. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of AI.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023. URL https://arxiv.org/abs/2306.06070.

Shiuli Subhra Ghosh, Anmol Dwivedi, Ali Tajer, Kyongmin Yeo, and Wesley M. Gifford. Cascading failure prediction via causal inference, 2024. URL https://arxiv.org/abs/2410.19179.

Yina Hou, Shourav B. Rabbani, Liang Hong, Norou Diawara, and Manar D. Samad. Causal explainability of machine learning in heart failure prediction from electronic health records, 2025. URL https://arxiv.org/abs/2506.03068.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Han Li, Yifan Yao, Letian Zhu, Rili Feng, Hongyi Ye, Jiaming Wang, Yancheng He, Pengyu Zou, Lehan Zhang, Xinping Lei, Haoyang Huang, Ken Deng, Ming Sun, Zhaoxiang Zhang, He Ye, and Jiaheng Liu. Codetracer: Towards traceable agent states, 2026. URL https://arxiv.org/abs/2604.11641.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025. URL https://arxiv.org/abs/2308.03688.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Shangshu Qian, Lin Tan, and Yongle Zhang. Csnake: Detecting self-sustaining cascading failure via causal stitching of fault propagations, 2025. URL https://arxiv.org/abs/2509.26529.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Khurram Yamin, Shantanu Gupta, Gaurav R. Ghosal, Zachary C. Lipton, and Bryan Wilder. Failure modes of llms for causal reasoning on narratives, 2025. URL https://arxiv.org/abs/2410.23884.

Shu-Xun Yang, Cunxiang Wang, Yidong Wang, Xiaotao Gu, Minlie Huang, and Jie Tang. Stepmathagent: A step-wise agent for evaluating mathematical processes through tree-of-error, 2025. URL https://arxiv.org/abs/2503.10105.

Shu-Xun Yang, Cunxiang Wang, Haoke Zhang, Wenbo Yu, Lindong Wu, Jiayi Gui, Dayong Yang, Yukuo Cen, Zhuoer Feng, Bosi Wen, Yidong Wang, Lucen Zhong, Jiamin Ren, Linfeng Zhang, and Jie Tang. Tracesir: A multi-agent framework for structured analysis and reporting of agentic execution traces, 2026. URL https://arxiv.org/abs/2603.00623.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

A CEG-Agent Implementation Details

This section instantiates the abstractions in §3 as the operational components of CEG-Agent. We describe trace normalization, the registered tool set and its dispatch table, the agentic runtime, the deterministic validator and auto-repair rules that realize the schema projection Π, the LLM-based critic loop with its compact patch language, the counterfactual repair-value functional, the sub-agent delegation interface, and the per-trace output bundle.

A.1 Trace Normalization

A deterministic preprocessor maps each provider-specific message $m_i \in \mathcal{M}$ to a normalized turn. Assistant messages are decomposed into a *thought* field, extracted from `reasoning_content` when available and otherwise from `content`, and an *action* field, extracted from serialized `tool_calls` when present. Tool messages, as well as user messages that serve as environment or human responses to a preceding assistant action, are attached as the *observation* of the corresponding assistant turn, yielding a thought–action–observation triple for each execution unit. The first user message is treated as the task specification τ, while subsequent user turns are retained when they provide feedback or observations during the interaction. The normalized trace preserves the original execution order and is stored in the output bundle for auditability.

Category	Tool	Method	Mode	Effect on S_t
Inspection	`read_trace`	Trace inspection	read-only	Read the normalized trace T.
Inspection	`inspect_state`	State inspection	read-only	Read the current graph G_t and todo buffer $\mathcal{Q}_t$.
Phase	`build_events`	Event construction	generator	Write event nodes V_E.
Phase	`find_failures`	Failure analysis	generator	Write failure nodes V_F.
Phase	`propose_errors`	Error hypothesis generation	generator	Write causally relevant error nodes V_R.
Phase	`build_causal_graph`	Causal graph induction	generator	Write causal edges E_C.
Phase	`find_anomalies`	Anomaly detection	generator	Write anomaly nodes V_A.
Phase	`estimate_repair_values`	Repair-value estimation	optional	Write repair values $\rho^\sharp$.
Critic	`validate_graph`	Graph validation and schema projection	deterministic	Emit issue set $\mathcal{I}(G_t)$ and apply $G_t \leftarrow \Pi(G_t)$ to structurally repairable violations.
Critic	`apply_critic_patch`	Patch-based critic loop	critic patch	Apply an LLM critic patch p and project back to the schema: $G_t \leftarrow \Pi(\mathrm{Apply}(G_t, p))$.
Meta	`write_todo`	Progress tracking	state update	Update the todo buffer $\mathcal{Q}_t$.
Meta	`spawn_subagent`	Sub-agent delegation	delegation	Dispatch a read-only sub-agent for localized investigation.
Meta	`render_graph`	Graph visualization	serialization	Export the CEG as DOT, HTML, or SVG.
Terminal	`finalize`	Finalization	terminal	Halt the inference loop and trigger final projection.

Table 5: Registered tool set $\mathcal{U}$ of CEG-Agent. Tools are grouped by category, linked to the corresponding method component, and summarized by their execution mode and effect on the diagnosis state S_t.

A.2 Tool Set

The registered tool set $\mathcal{U}$ contains fourteen tools, organized by their role in the diagnosis state S_t and by the method component they implement, as shown in Table 5. Each tool is exposed through an OpenAI-style function-calling JSON schema and dispatched by name. Core diagnosis tools update specific components of the CEG, while inspection and meta tools provide read-only access, working-memory management, sub-agent delegation, and visualization. The repair-value tool is enabled by default and can be disabled through `CE_ENABLE_REPAIR`; when disabled, `estimate_repair_values` is not registered and therefore cannot be invoked by the agent. The number of critic-based refinement rounds is controlled by `CE_CRITIC_ROUNDS`. At runtime, CEG-Agent automatically selects and invokes these tools based on the evolving diagnosis state and refinement needs.

A.3 Agentic Runtime

The runtime implements the inference loop in Algorithm 1. At each iteration, the top-level LLM receives the conversation history H_t, a JSON serialization of the current diagnosis state S_t, and the tool specifications in $\mathcal{U}$. It then emits zero or more tool calls, which are dispatched sequentially and applied to the state through the corresponding transition functions Φ_u. The loop terminates when the model invokes `finalize`, emits no tool call, or reaches the iteration budget, which is set to $K = 24$ in our experiments.

Per-tool exception isolation. If a tool handler raises an exception or returns malformed output, the runtime intercepts the failure and returns a structured error object $\{\texttt{error} : \cdot\}$ as the tool result, while

Algorithm 1 CEG-Agent Inference

Require: trace T, tool set $\mathcal{U}$, budget K
$S_0 \leftarrow \textsc{InitState}(T)$; $G_0 \leftarrow \varnothing$
for $t = 0, \ldots, K-1$ **do**
  $(u_t, \mathbf{a}_t) \sim \pi_\theta(\cdot \mid S_t, \mathcal{U})$
  $S_{t+1} \leftarrow \Phi_{u_t}(S_t, \mathbf{a}_t)$
  if $u_t =$ `finalize` **then break**
end for
return $G^\star \leftarrow \Pi(G_{t+1})$ ▷ schema projection

leaving the graph state G_t unchanged. This allows the policy to recover from local tool failures, such as malformed LLM-generated JSON inside a phase tool, without aborting the entire trajectory.

Termination safety net. After the loop exits, the schema projection Π is invoked once more regardless of whether the model explicitly called `finalize`. This final projection ensures that the released graph $G^\star = \Pi(G_t)$ always conforms to the CEG schema and satisfies the required well-formedness constraints.

A.4 Schema Projection

CEG inference can produce two qualitatively different types of issues. *Structural* violations, such as illegal edge types, dangling endpoints, duplicate edges, missing schema edges, and role–topology mismatches, can be repaired mechanically. In contrast, *semantic* violations, such as duplicated failures, anomaly–error confusion, or evidence-dependent role errors, require trace-level judgment. CEG-Agent separates these cases using a deterministic validator together with an LLM-driven patch critic (§A.5).

Validator and issue partition. Let $\mathcal{V} : \mathcal{G} \to 2^{\mathrm{IssueCode}}$ be a deterministic validator that maps a graph G to an issue set $\mathcal{I}(G) = \mathcal{I}_{\mathrm{det}}(G) \dot{\cup} \mathcal{I}_{\mathrm{sem}}(G)$, where the partition is determined by whether an issue admits a mechanical repair. The deterministic partition $\mathcal{I}_{\mathrm{det}}$ includes unknown edge types, illegal or dangling endpoints, duplicate edges, missing EVENT-NEXT chains, missing ATTACHED-TO edges, and role–topology mismatches. The semantic partition $\mathcal{I}_{\mathrm{sem}}$ includes anomaly–error overlap, missing or excessive failures, errors unreachable from any failure, and role assignments that require trace evidence to resolve.

Deterministic auto-repair $\mathcal{R}_{\mathrm{det}}$. The auto-repair routine realizes the schema projection Π through five deterministic rules: (i) remove edges that violate Eq. (2) or reference absent endpoints; (ii) deduplicate edges by (*source*, *type*, *target*); (iii) regenerate the EVENT-NEXT chain from event order; (iv) regenerate ATTACHED-TO edges from the event grounding $\varepsilon(r)$ of each error; and (v) recompute the structural role $\rho(r)$ from graph topology, demoting a ROOT with incoming CAUSES edges to PROPAGATED and assigning AMPLIFICATION when r participates in any AMPLIFIES edge. These rules remove all mechanically repairable violations without relying on trace semantics, yielding:

$$\Pi(G) \;=\; \lim_{k \to \infty} \mathcal{R}_{\mathrm{det}}^{k}(G), \quad \mathcal{I}_{\mathrm{det}}\big(\Pi(G)\big) = \varnothing. \tag{4}$$

By construction, Π is idempotent, i.e., $\Pi \circ \Pi = \Pi$. This allows the runtime to invoke schema projection both inside critic-based repair tools and once again at termination, ensuring that the released graph remains structurally well formed.

A.5 LLM Critic Loop

Semantic issues in $\mathcal{I}_{\mathrm{sem}}(G)$ require evidence-based judgment and cannot be resolved by deterministic projection alone. CEG-Agent therefore uses an LLM critic policy $\pi_\theta^{\mathrm{crit}}$ to propose a compact *graph patch* $p \in \mathcal{P}$, rather than regenerating the entire graph from scratch.

Patch language. A patch is represented as:

$$p = \big(\Delta_E^+,\, \Delta_E^-,\, \Delta_R^-,\, \Delta_F^-,\, \Delta_A^-,\, \Delta_{\mathrm{attr}},\, \Delta_{\mathrm{merge}}\big), \tag{5}$$

where Δ_E^+ and Δ_E^- denote edge insertions and deletions, $\Delta_R^-, \Delta_F^-, \Delta_A^-$ denote deletions of error, failure, and anomaly nodes, Δ_{attr} denotes in-place attribute updates such as role, mechanism, or description edits, and the merge set $\Delta_{\text{merge}} \subset V_R \times 2^{V_R}$ specifies error-node merges. The interpreter $\text{Apply}(\cdot, p)$ executes these patch operations, including merges, deletions, updates, and insertions, and discards any operation that references an absent node or violates the typed-edge constraints in Eq. (2).

Iterative refinement. Each critic round applies one patch and then projects the result back to the CEG schema:

$$\begin{aligned} p^{(k)} &\sim \pi_\theta^{\text{crit}}\big(\cdot \mid G^{(k)}, \mathcal{I}(G^{(k)})\big), \\ G^{(k+1)} &= \Pi\Big(\text{Apply}\big(G^{(k)}, p^{(k)}\big)\Big). \end{aligned} \tag{6}$$

The number of critic rounds is bounded by a critic budget C, with default $C = 2$. Since Π is applied after every patch and once again at termination, the final graph $G^\star = \Pi(G^{(C)})$ satisfies $\text{W}_1 \wedge \text{W}_2 \wedge \text{W}_3$ even when the critic proposes malformed edits. Restricting the critic to local patches makes the repair process auditable and limits the potential damage of any single revision.

A.6 Counterfactual Repair Valuation

For each error $r \in V_R$ in the CEG, the tool `estimate_repair_values` assigns a counterfactual repair value $\rho^\sharp(r) \in \{\text{HIGH}, \text{MEDIUM}, \text{LOW}\}$, estimating whether correcting only r would avoid or substantially mitigate the primary failure $f^\star$. The valuation combines four graph-structural signals with a bounded lexical signal in a clipped linear functional.

Structural primitives. Let G_{-r} denote the induced subgraph obtained by deleting r and all incident edges from G. We define the set of errors contributing to $f^\star$ as:

$$\mathcal{C}_G(f^\star) = \big\{q \in V_R \mid f^\star \in \text{Reach}_G(q)\big\}. \tag{7}$$

The *softened counterfactual severance* of r is the fraction of contributing errors whose reachability to $f^\star$ is cut after removing r:

$$\sigma(r, f^\star) = \frac{\big|\{q \in \mathcal{C}_G(f^\star) \setminus \{r\} : f^\star \notin \text{Reach}_{G_{-r}}(q)\}\big|}{|\mathcal{C}_G(f^\star)|}. \tag{8}$$

We set $\sigma(r, f^\star) = 0$ when $r \notin \mathcal{C}_G(f^\star)$ or $\mathcal{C}_G(f^\star) = \varnothing$. A value $\sigma = 1$ indicates that r is a unique bottleneck, whereas smaller values indicate that r is one of several parallel contributors. It is worth noting that this graded computation is used for the gold annotations in CEG-Bench. In CEG-Agent inference, for simplicity, we ask the LLM to provide a binary estimate of this signal, i.e., whether r acts as a counterfactual bottleneck or not.

Amplification credit. Let $\text{Succ}_{\text{AMPL}}(r) = \{q : (r, \text{AMPLIFIES}, q) \in E\}$. The amplification credit is:

$$\alpha(r) = 1 - \exp\big(-|\text{Succ}_{\text{AMPL}}(r)|\big), \tag{9}$$

a saturating function of the AMPLIFIES out-degree. This prevents an error from dominating the score solely by amplifying many downstream errors.

Softened upstream hazard. For propagated errors, we penalize cases where the upstream cause remains unresolved. Let $\text{Pred}_{\text{CAUSE}}(r) = \{u : (u, \text{CAUSES}, r) \in E\}$. The softened upstream hazard is:

$$\begin{aligned} \eta(r) = \mathbb{1}\big[\rho(r) = \text{PROPAGATED}\big] \cdot \frac{1}{|\text{Pred}_{\text{CAUSE}}(r)|} \\ \cdot \sum_{u \in \text{Pred}_{\text{CAUSE}}(r)} \big(1 - \sigma(u, f^\star)\big), \end{aligned} \tag{10}$$

with the convention $\eta(r) = 0$ when $\text{Pred}_{\text{CAUSE}}(r) = \varnothing$. Intuitively, $\eta(r)$ is small when upstream causes are themselves strong bottlenecks, and large when they are weak parallel contributors that would remain unresolved by fixing r alone. As with σ, this graded form is used for gold annotations in CEG-Bench, while CEG-Agent inference uses a binary surrogate $\eta(r) \in \{0, 1\}$ that triggers when r is propagated and has at least one upstream CAUSES edge.

Composite score and bucketing. Let $\pi(\rho)$ be a role prior with $\pi(\text{ROOT}) = 1.0$, $\pi(\text{AMPLIFICATION}) = 0.5$, and $\pi(\text{PROPAGATED}) = 0.3$. Let $h(r) \in [0.3, 0.7]$ be a bounded lexical hedging signal extracted from $d(r)$. The composite repair score is:

$$s(r) = \text{clip}_{[0,1]}\Big(w_\sigma \sigma(r, f^\star) + w_\pi \pi(\rho(r)) + w_\alpha \alpha(r) + w_h h(r) - w_\eta \eta(r)\Big), \tag{11}$$

with weights:

$$(w_\sigma, w_\pi, w_\alpha, w_h, w_\eta) = (0.45, 0.25, 0.15, 0.10, 0.30). \tag{12}$$

The positive terms reward counterfactual severance, root-like structural role, amplification effects, and confident descriptions, while the hazard term discourages assigning high repair value to propagated symptoms whose upstream causes remain unresolved. The final bucket is:

$$\rho^\sharp(r) = \begin{cases} \text{HIGH}, & s(r) \geq 0.66, \\ \text{MEDIUM}, & 0.33 \leq s(r) < 0.66, \\ \text{LOW}, & s(r) < 0.33. \end{cases} \tag{13}$$

Two hard overrides are applied: if $r \notin \mathcal{C}_G(f^\star)$, then $\rho^\sharp(r)$ is forced to LOW; and if $\eta(r) > 0$, then $\rho^\sharp(r)$ is capped at MEDIUM. All terms in Eq. (11) except $h(r)$ are mechanically computable from G, making discrepancies between the LLM-assigned bucket and the structural bucket localizable.

A.7 Sub-Agent Delegation

The top-level agent may invoke `spawn_subagent` to delegate a narrow diagnostic question without mutating the parent state. Each sub-agent is launched with a restricted tool set $\mathcal{U}_{\text{sub}} = \{$`read_trace`, `inspect_state`, `validate_graph`$\}$, where `validate_graph` is wrapped with `run_auto_repair=false` to prevent implicit state changes. The sub-agent operates on a snapshot of the parent state, with write operations intercepted so that its analysis remains effectively read-only. It returns a final JSON report containing *finding*, *evidence*, and *suggested_patch*. The parent agent then decides whether to apply the suggested patch through `apply_critic_patch`, separating diagnostic consultation from state mutation.

A.8 Output Bundle

Each trace produces a self-contained output bundle containing the final graph $G^\star$ as JSON (`diagnosis.json`), the full agentic step sequence (`agentic_steps.jsonl`), per-call token and latency logs (`tool_trace.jsonl`), aggregate metadata (`summary.json`), the console log (`run.log`), and graph visualizations in DOT, HTML, and SVG when Graphviz is available. Token accounting is handled by a `TokenAccountant`, which labels each LLM call by phase and tool invocation, enabling fine-grained cost analysis over the trajectory. These bundles serve as the unit of downstream evaluation: benchmark-level statistics and metrics can be computed by traversing the `diagnosis.json` files across a full run.

B CEG-Bench Dataset Analysis

This appendix provides a detailed analysis of **CEG-Bench**, the 100-trace benchmark introduced in §4. CEG-Bench is designed to evaluate causal trace diagnosis beyond final-task success, guided by the principle that a diagnostic system should identify outcome-level failures V_F, recover causally relevant process-level errors V_R with event grounding $\varepsilon(r)$, reconstruct the typed causal propagation structure linking errors to failures, and distinguish surface anomalies V_A from genuine causal contributors. The benchmark is constructed by sampling failed agentic traces, annotating them through the **Adversarial Agentic Adjudication Protocol** (AAAP), and retaining only cases with dual-adversary consensus.

During the AAAP construction process, the asymmetric four-role design is intended to reduce systematic annotation bias. A single drafter has no external mechanism for correcting its own misattributions, while a drafter paired with only one reviewer may still inherit shared model priors. AAAP therefore separates constructive graph proposal from adversarial certification: the drafter

Component	Tau2 $n = 36$	SWE $n = 35$	Browse $n = 29$	All $n = 100$
Events	13.3	47.8	46.1	34.9
Errors	2.0	3.9	6.2	3.9
Failures	0.9	1.4	1.5	1.2
Anomalies	2.3	2.5	2.8	2.5
Edges	18.0	57.7	64.6	45.4

Table 6: Mean per-trace component counts in CEG-Bench by source benchmark. Tau2, SWE, and Browse denote Tau2Bench, SWE-bench, and BrowseComp.

and reviewer propose and refine the CEG, while two adversaries from distinct model families independently cross-examine the result. A trace is retained only when both adversaries certify the graph, making the final annotation an intersection of two independent diagnostic perspectives. This conservative protocol rejects ambiguous cases rather than resolving them by tie-breaking, yielding the final consensus set of 100 traces.

As noted above, AAAP is still a fully LLM-based annotation protocol and therefore has inherent limitations. Although the initial pool contains 150 failed agentic traces following TraceSIR, some candidate traces are judged by AAAP to contain no clear causal error. This mainly arises from a difference in annotation setting: TraceSIR can leverage explicit failure signals such as wrong final answers or evaluator feedback when identifying errors, whereas CEG-Bench does not expose such external outcome feedback to the annotators. We make this choice to preserve the generality and fairness of CEG construction, so that the annotation process relies only on the trace itself and can be directly reproduced or extended by future users. Under this setting, even human annotators may judge some traces as acceptable if the failure is not evident from the trace alone. We therefore defer to the final AAAP consensus for inclusion decisions. Despite these limitations, we believe CEG-Bench provides a valuable reference benchmark, since its retained annotations are produced and certified through agreement between two strong agentic frameworks powered by independent frontier LLM families.

Under this construction protocol, the resulting benchmark exhibits diverse causal structures across events, errors, failures, anomalies, and typed relations. The 100 retained traces collectively contain 3,488 events, 391 errors, 125 failures, 253 anomalies, and 4,540 typed edges, including 1,152 non-temporal edges. The annotations cover all eight mechanisms in Mech; the four most frequent are *planning* (19.7%), *evidence integration* (18.4%), *execution* (16.9%), and *self-evaluation* (15.6%), together accounting for 70.6% of all errors. The role distribution is skewed toward PROPAGATED errors (39.6%), followed by ROOT errors (33.2%) and AMPLIFICATION errors (27.1%), suggesting that failures in long agentic traces often arise through cascading downstream consequences of a smaller set of root causes. Repair-value buckets are also highly imbalanced: only 3.3% of errors receive a HIGH value under $\rho^\sharp$, reflecting the strict bottleneck requirement in the counterfactual repair criterion. Detailed benchmark distributions are reported below.

B.1 Per-Source Size Distributions

Table 6 reports the mean per-trace counts of events, errors, failures, anomalies, and edges for each source benchmark. The three sources exhibit substantially different trace profiles. Tau2Bench consists of relatively short customer-service dialogues, with a median of 14 events, where failures often trace to a small number of dominant errors. SWE-bench contains longer edit–test–debug trajectories, with a median of 35 events and a maximum of 250 events, where errors may accumulate across many tool calls. BrowseComp is the densest source in terms of errors per trace, with a mean of 6.2 errors compared with 2.0 for Tau2Bench, reflecting the multi-step evidence gathering and reasoning demands of deep-research tasks. Overall, the 50$\times$ length ratio between the longest SWE-bench trace and the shortest Tau2Bench trace stress-tests both trace normalization and the agentic inference budget K from §A.3.

Label	Count	%
Mechanism μ (*errors*, $n = 391$)		
planning	77	19.7
evidence integration	72	18.4
execution	66	16.9
self-evaluation	61	15.6
control	53	13.6
omission	33	8.4
representation	25	6.4
environment	4	1.0
Structural role ρ (*errors*, $n = 391$)		
propagated	155	39.6
root	130	33.2
amplification	106	27.1
Repair value $\rho^\sharp$ (*errors*, $n = 391$)		
low	244	62.4
medium	134	34.3
high	13	3.3
Failure type θ (*failures*, $n = 125$)		
completion	53	42.4
correctness	43	34.4
constraint	17	13.6
efficiency	6	4.8
safety	3	2.4
other	3	2.4

Table 7: Marginal distributions over the controlled vocabularies in CEG-Bench. Percentages for μ, ρ, and $\rho^\sharp$ are computed over the 391 annotated errors; percentages for θ are computed over the 125 annotated failures.

B.2 Vocabulary Distributions

Table 7 summarizes the marginal distributions over the controlled vocabularies defined in §3.2: error mechanism μ, structural role ρ, failure type θ, and bucketed counterfactual repair value $\rho^\sharp$. These distributions characterize the diagnostic diversity of CEG-Bench across process-level error mechanisms, causal roles, outcome-level failures, and repair priorities. Specifically, all eight mechanisms are represented in the benchmark. The most frequent mechanisms are *planning* (19.7%), *evidence integration* (18.4%), *execution* (16.9%), and *self-evaluation* (15.6%), which together cover 70.6% of all annotated errors. Failure types are dominated by *completion* (42.4%) and *correctness* (34.4%), with *constraint* failures contributing another 13.6%; these three categories account for 90.4% of all failures. The role distribution is skewed toward PROPAGATED errors, reflecting the cascading nature of long agentic traces: many observed errors are downstream consequences of a smaller set of root causes, which the CEG structural-role typology is designed to expose.

B.3 Edge Structure of the Causal Subgraph

The full edge set E contains 3,388 temporal EVENT-NEXT edges, forming one event chain per trace, and 1,152 non-temporal edges. The non-temporal edges consist of 403 ATTACHED-TO grounding edges, 422 CONTRIBUTES-TO error-to-failure edges, 266 CAUSES edges, and 61 AMPLIFIES edges. Excluding temporal and grounding edges, the causal subgraph G_C contains 749 causal edges, averaging 7.5 edges per trace. The relative sparsity of AMPLIFIES edges compared with CAUSES edges, at roughly $1 : 4.4$, indicates that amplification is a specialized relation rather than a generic propagation link. Most causal structure in CEG-Bench therefore flows through error propagation chains that terminate in CONTRIBUTES-TO edges to failures, while AMPLIFIES is reserved for cases where one error meaningfully magnifies the impact of another.

Mechanism	HIGH	MED.	LOW	Total
planning	0	40	37	77
evidence integration	3	19	50	72
execution	3	18	45	66
self-evaluation	0	14	47	61
control	1	15	37	53
omission	2	11	20	33
representation	0	17	8	25
environment	4	0	0	4
Total	**13**	**134**	**244**	**391**

Table 8: Repair-value buckets cross-tabulated by error mechanism. Counts are computed over the 391 annotated errors in CEG-Bench.

B.4 Repair-Value Distribution

The bucketed counterfactual repair values $\rho^\sharp$ from §A.6 are highly long-tailed: 244 errors (62.4%) fall in LOW, 134 errors (34.3%) in MEDIUM, and only 13 errors (3.3%) in HIGH. This skew follows from the repair-value rubric. To receive a HIGH value, an error must act as a strong counterfactual bottleneck for the primary failure $f^\star$, reflected by a large severance score $\sigma(r, f^\star)$ in Eq. (8), and must also avoid the hard overrides in the bucketing rule of Eq. (13). Table 8 cross-tabulates repair-value buckets against error mechanisms. The 13 HIGH-valued errors are concentrated in *environment*, *evidence integration*, *execution*, and *omission*. In particular, all four *environment* errors receive HIGH values, while the remaining high-value cases typically correspond to situations where a single missing, incorrect, or unavailable action blocks task completion under the annotated causal graph. Thus, CEG-Bench provides not only diagnostic structure but also a built-in repair-prioritization signal for downstream debugging methods.

C CEG Evaluation Protocols

This appendix specifies the two evaluation protocols introduced in §4.3. We release both as reference implementations.

C.1 Semantically Relaxed CEG Evaluation

The semantically relaxed evaluator first aligns predicted and gold nodes across graphs using bipartite matching. It then translates predicted identifiers into the gold identifier space, computes precision, recall, and F1 for each scoring dimension, and aggregates the resulting dimension scores into a tier-weighted composite.

Node alignment. For each node type $c \in \{\text{event}, \text{error}, \text{failure}, \text{anomaly}\}$, let $V_c^\star$ denote the gold node set and $\hat{V}_c$ the predicted node set. We construct a similarity matrix $\mathbf{M}_c \in [0,1]^{|V_c^\star| \times |\hat{V}_c|}$ and solve a maximum-weight bipartite matching problem:

$$\Lambda_c^\star = \arg \max_{\Lambda \in \text{Match}(V_c^\star, \hat{V}_c)} \sum_{(g,p)\in\Lambda} \mathbf{M}_c[g,p]. \tag{14}$$

Only matched pairs satisfying $\mathbf{M}_c[g,p] \geq \tau_c$ are retained. The default type-specific thresholds are $\tau_{\text{event}} = \tau_{\text{error}} = \tau_{\text{failure}} = \tau_{\text{anomaly}} = 0.60$.

Per-node similarity. Node similarity is type-specific. For error nodes, we use the following empirical similarity function:

$$\begin{aligned} \text{sim}(r, \hat{r}) &= w_e \, \text{sim}_E\big(\varepsilon(r), \varepsilon(\hat{r})\big) \\ &+ w_\mu \mathbb{1}[\mu(r) = \mu(\hat{r})] + w_\rho \mathbb{1}[\rho(r) = \rho(\hat{r})] \\ &+ w_\nu \, \text{sim}_T\big(\nu(r), \nu(\hat{r})\big) + w_d \, \text{sim}_T\big(d(r), d(\hat{r})\big), \end{aligned} \tag{15}$$

with weights $(w_e, w_\mu, w_\rho, w_\nu, w_d) = (0.25, 0.20, 0.15, 0.15, 0.25)$. We set these weights empirically; in practice, the final scores are not sensitive to small perturbations of them. Here sim_E denotes cross-graph event-alignment similarity, computed using an inner Hungarian alignment over events together with an order-consistency factor based on the longest non-decreasing subsequence of matched event indices. For free-text fields, sim_T is computed as cosine similarity between BGE-M3 embeddings (Chen et al., 2024). Other node types use the corresponding subset of categorical, event, and textual fields.

Dimensional scoring. After node alignment, predicted identifiers are translated into gold identifiers through the matched pairs. Each scoring dimension is then evaluated using precision, recall, and F1 over either typed node multisets or translated edge tuples. The relaxed evaluator reports eight dimensions grouped into three tiers:

- **Tier 1: structural dimensions** with total weight 0.60: `error_node`, `causal_edge`, and `main_chain`, each weighted by 0.20.
- **Tier 2: textual and sparse dimensions** with total weight 0.30: `failure`, `attribution`, and `anomaly`, each weighted by 0.10.
- **Tier 3: utility dimensions** with total weight 0.10: `event` and `consistency`, each weighted by 0.05.

The `main_chain` dimension is the only non-elementary dimension. For each graph, we extract a canonical root-to-failure path in the causal subgraph G_C, defined as the longest weighted path from any ROOT error to the primary failure $f^\star$ along causal labels in $\mathcal{L}_C$. Chain similarity is then computed by a weighted combination of node overlap, edge overlap, and length-normalized longest common subsequence (LCS) over the matched error sequence. The `consistency` dimension checks intra-graph structural invariants, such as the reachability constraint W_3, and penalizes violations in $\hat{G}$.

Composite score. The final relaxed score is the tier-weighted sum of dimension-level F1 scores:

$$\text{CEG-Sim}(\hat{G}, G^\star) = \sum_c w_c \, \mathrm{F1}_c \in [0, 1], \tag{16}$$

where $\sum_c w_c = 1$. We use CEG-Sim as the headline semantically relaxed metric.

C.2 Structurally Exact CEG Evaluation

The structurally exact evaluator is rule-based, LLM-free, and performs no fuzzy matching. It asks whether a prediction $\hat{G}$ reconstructs the consensus graph $G^\star$ as a structured object, rather than merely matching it semantically.

Cross-graph identifier resolution. Gold and predicted CEGs use independent local identifier spaces for errors and failures, but both refer to the same shared event identifiers V_E produced by the deterministic trace parser in §A.1. We therefore resolve cross-graph identifiers without text comparison: errors are matched by the exact tuple key $(\varepsilon(r), \mu(r))$, combining event grounding and controlled mechanism, while failures are matched by the type bucket $\theta(f)$. The resulting maps $\phi_R : \hat{V}_R \to V_R^\star$ and $\phi_F : \hat{V}_F \to V_F^\star$ translate predicted nodes and edges into the gold identifier space.

Dimensional scoring. The evaluator scores seven dimensions, each using precision, recall, and F1 over finite sets or multisets of typed tuples:

- **event**: identifier-set match over V_E.
- **event_next**: edge-set match over $\{(u, v) : (u, \text{EVENT-NEXT}, v) \in E\}$.
- **failure**: multiset match over failure buckets $\theta(f)$, including count agreement.
- **error**: match over $(\varepsilon(r), \mu(r))$ tuples, with an additional $\rho(r)$ agreement check on matched pairs.
- **attached_to**: translated ATTACHED-TO edge-set match using ϕ_R.

- **causes**: translated match over the union of CAUSES and AMPLIFIES edges using ϕ_R.
- **contributes_to**: translated error-to-failure edge-set match using ϕ_R and ϕ_F.

Predicted edges incident to unmatched errors or failures are counted as false positives. This propagates node-resolution errors into downstream edge dimensions without introducing additional cascading penalties.

Composite score. The composite score is the unweighted mean of five model-discriminative dimensions. Let $\mathcal{D}^\star = \{\texttt{failure}, \texttt{error}, \texttt{attached_to}, \texttt{causes}, \texttt{contributes_to}\}$. Then,

$$\text{CEG-Exact}(\hat{G}, G^\star) = \frac{1}{|\mathcal{D}^\star|} \sum_{c \in \mathcal{D}^\star} \text{F1}_c. \tag{17}$$

The two parser-driven dimensions, `event` and `event_next`, are reported but excluded from the composite because they are determined by the shared trace parser rather than by model capability.

D HUMAN GOLD VALIDATION OF CEG-BENCH

The annotations in **CEG-Bench** are produced by agentic systems through AAAP rather than by exhaustive human annotation. A natural concern is therefore whether these automatically constructed CEGs are reliable. To provide an independent audit, we construct **CEG-Bench-Gold**, an expert-curated human reference set over the same 100 traces. This appendix describes the construction of CEG-Bench-Gold, measures the agreement between AAAP and human annotations, and re-evaluates all systems against the human reference under the same SRE and SEE criteria used in the main paper.

Construction of CEG-Bench-Gold. We invited eight domain experts, each with graduate-level training in computer science, to annotate and verify the 100 traces in CEG-Bench. Given the raw trace and the CEG schema in §3.2, experts marked events, failures, anomalies, causally relevant errors with their mechanisms and roles, and the causal edges among errors and failures. Each trace was reviewed by multiple experts, and disagreements were resolved through discussion into a consensus human graph. Constructing CEGs is difficult even for experts because agentic traces are long, heterogeneous, and often causally ambiguous. We therefore use CEG-Bench-Gold as a carefully curated audit set for validating AAAP, rather than as a scalable replacement for automated annotation. We release CEG-Bench-Gold together with the AAAP-annotated CEG-Bench.

AAAP agreement with the human gold. We quantify the reliability of AAAP by scoring the AAAP annotations against CEG-Bench-Gold, treating the human graph as the reference and the AAAP graph as the prediction. The same SRE and SEE metrics from §4.3 are used. As shown in Figure 4 and Table 9, AAAP shows strong agreement with expert judgments. It achieves an overall CEG-Sim of 71.5, with high SRE agreement on anomalies (88.1), errors (76.3), and failures (76.3). Agreement is lower on the strictest causal-wiring dimensions, such as SEE `causes` (20.1) and `contributes_to` (36.2), which require exact reconstruction of fine-grained graph edges and are difficult even under careful human adjudication. Notably, SEE agreement increases with trace complexity, from 43.0 on small traces to 45.8 on medium traces and 52.4 on large traces, suggesting that AAAP remains reliable on the long, causally rich traces for which structured diagnosis is most needed.

System scores against the human gold. To check whether our main conclusions depend on the AAAP reference, we additionally evaluate every system against CEG-Bench-Gold using the same SRE and SEE criteria. Tables 10 and 11 mirror the two main result tables, but replace the AAAP annotations with the human reference. The within-family ranking is preserved: **CEG-Agent obtains the best composite SRE and SEE within both backbone families**. This confirms that the observed advantage of CEG-Agent is not an artifact of evaluating against agent-generated annotations.

As in the main results, the only dimension where a baseline remains competitive is SEE `failure`, where systems that emit a small number of failure nodes can be mechanically favored on some traces (§E.5). Absolute scores are generally lower than those obtained against the AAAP reference, which is expected because CEG-Bench-Gold is an independent and stricter reference. Importantly, this shift

Criterion	Dimension	Agreement
SRE	**CEG-Sim (overall)**	**71.5**
	anomaly	88.1
	error	76.3
	failure	76.3
	main_chain	61.9
	attribution	58.8
	causal_edge	58.0
	event	99.0
SEE	**overall**	**47.1**
	failure	76.6
	error	52.4
	attached_to	50.2
	contributes_to	36.2
	causes	20.1

Table 9: Agreement between the AAAP annotations in CEG-Bench and the expert-curated CEG-Bench-Gold, measured with SRE and SEE over all 100 traces (%). SEE agreement by complexity tercile is $43.0/45.8/52.4$ for small/medium/large traces.

Backbone	System	SRE	Structural			Textual			Utility	
			error	edge	chain	fail.	attr.	anom.	event	cons.
Claude-Opus-4.7	Claude Code (raw)	38.1	36.3	26.5	24.8	**49.1**	24.4	43.4	61.4	**100.0**
Claude-Opus-4.7	Claude Code (+Process)	38.1	35.6	26.0	23.6	40.5	23.8	46.5	**99.0**	**100.0**
Claude-Opus-4.7	**CEG-Claude** (ours)	**45.4**	**43.2**	**30.2**	**37.1**	45.0	**30.6**	**57.6**	**99.0**	99.9
GPT-5.5	Codex (raw)	27.7	21.4	22.2	15.1	38.3	16.5	6.9	99.0	100.0
GPT-5.5	Codex (+Process)	26.6	21.7	23.5	15.1	26.6	12.4	6.9	99.0	100.0
GPT-5.5	**CEG-Codex** (ours)	**40.8**	**38.1**	**28.0**	**29.3**	**44.6**	**24.9**	**48.8**	99.0	100.0

Table 10: Semantically relaxed evaluation (**SRE**) against the expert-curated **CEG-Bench-Gold**, using the same layout as Table 1. Best score within each backbone block is in **bold**. Scores are lower than those against the AAAP reference, but the within-family ranking is preserved.

lowers all systems together without changing the main ordering, supporting the robustness of our conclusions.

Why the main paper uses AAAP as the primary reference. The human-gold evaluations above confirm that CEG-Agent's advantage holds under an independent expert reference. We nevertheless use the AAAP-constructed CEG-Bench as the primary benchmark in the main paper for two reasons. First, scalable automated annotation is a central contribution of this work: high-quality human annotation of long causal traces is expensive, slow, and rarely available in real deployments, whereas AAAP is fully automated and, as shown above, closely agrees with expert judgments. Second, using a single AAAP reference throughout the paper keeps the benchmark analyses, ablations, cross-backbone comparisons, and closed-loop repair studies directly comparable. We therefore treat CEG-Bench-Gold as an independent audit of AAAP reliability, while reporting the main experiments against the AAAP annotations.

E Experimental Analyses

This appendix section expands §5 with the full set of experimental analyses that are summarized in the main paper. It contains a detailed experimental setup (§E.1), per-source results (§E.2), robustness to trace complexity (§E.3), full dimension-level diagnosis (§E.4), abstention behaviour on gold-empty traces (§E.5), graph-size calibration (§E.6), per-mechanism / per-role recall (§E.7), failure-type

Backbone	System	SEE	Dimensions				
			failure	error	attach	cause	contrib.
Claude-Opus-4.7	Claude Code (raw)	15.5	**60.1**	8.5	6.1	0.5	2.2
Claude-Opus-4.7	Claude Code (+Process)	19.5	49.0	18.4	17.4	1.3	11.3
Claude-Opus-4.7	**CEG-Claude** (ours)	**22.6**	57.9	**22.4**	**18.3**	**2.0**	**12.4**
GPT-5.5	Codex (raw)	16.3	55.3	8.9	8.7	0.0	8.6
GPT-5.5	Codex (+Process)	13.3	39.5	9.2	8.4	1.7	7.5
GPT-5.5	**CEG-Codex** (ours)	**22.4**	**59.5**	**18.5**	**17.6**	**2.3**	**14.1**

Table 11: Structurally exact evaluation (**SEE**) against the expert-curated **CEG-Bench-Gold**, using the same layout as Table 2. Best score within each backbone block is in **bold**.

coverage (§E.8), repair-value calibration (§E.9), inter-system agreement on irreducibly hard traces (§E.10), a per-backbone diagnostic capability profile (§E.11), the cost and tool-use profile (§E.12), and a qualitative case study (§F.1).

E.1 Detailed Experimental Setup

Systems. We compare CEG-Agent against strong agentic baselines on identical input traces. For each backbone family, we evaluate three systems:

- **Agentic baseline (raw).** An off-the-shelf coding agent, either Claude Code or Codex, receives the raw trace JSON together with the CEG schema and is asked to produce a single `diagnosis.json`. It must perform event segmentation and causal diagnosis end-to-end.
- **Agentic baseline (+Process).** The same agent receives traces that have already been segmented into events V_E and connected by EVENT-NEXT edges using our deterministic parser (§A.1). This setting isolates causal diagnosis from trace parsing and provides a stronger baseline for comparison.
- **CEG-Agent (ours).** The full tool-augmented framework described in §3, instantiated with the same backbone LLM as the corresponding baseline.

We use Claude-Opus-4.7 for the Claude Code / CEG-Claude pair and GPT-5.5 for the Codex / CEG-Codex pair. All systems are evaluated on the same 100 traces and against the same consensus CEG annotations.

Evaluation. We report the two complementary criteria defined in §4.3: the semantically relaxed evaluator $\mathrm{CEG\text{-}Sim}$ (**SRE**) and the structurally exact evaluator $\mathrm{CEG\text{-}Exact}$ (**SEE**), together with per-dimension breakdowns. Higher scores are better, and all values are reported as percentages.

Hyperparameters and scoring. CEG-Agent uses inference budget $K = 24$, critic budget $C = 2$, temperature $= 1$, and repair-value estimation enabled. Baselines are run with their respective coding agents' default CLI configurations.

E.2 Per-Source Performance

Table 12 shows that the benefit of CEG-Agent depends on source complexity. Under Claude-Opus-4.7, Claude Code (+Process) is strongest on the short dialogue-style Tau2Bench traces, where the task often reduces to attaching one or two errors to a small event scaffold. On the longer SWE-bench traces and the denser BrowseComp traces, the pattern reverses: CEG-Claude outperforms the +Process baseline by $+9.4$ and $+4.4$ points in SRE, and by $+6.2$ and $+1.7$ points in SEE, respectively. The Tau2Bench regression is largely explained by the gold-empty abstention pattern analyzed in §E.5.

Under GPT-5.5, the improvement is more uniform. CEG-Codex achieves the best SRE score on all three sources and the best SEE score on two of three sources, including a $+21.8$-point SRE gain over Codex (+Process) on Tau2Bench. The weak Tau2Bench SEE score of Codex (+Process) reflects a

Backbone	Evaluation	System	Tau2Bench $n = 36$	SWE-bench $n = 35$	BrowseComp $n = 29$
Claude-Opus-4.7	SRE	Claude Code (raw)	59.8	37.3	38.7
Claude-Opus-4.7	SRE	Claude Code (+Process)	**67.2**	38.3	41.5
Claude-Opus-4.7	SRE	**CEG-Claude** (ours)	60.0	**47.7**	**45.9**
Claude-Opus-4.7	SEE	Claude Code (raw)	16.8	11.1	13.8
Claude-Opus-4.7	SEE	Claude Code (+Process)	**24.9**	13.5	19.3
Claude-Opus-4.7	SEE	**CEG-Claude** (ours)	20.7	**19.7**	**21.0**
GPT-5.5	SRE	Codex (raw)	41.5	24.6	29.3
GPT-5.5	SRE	Codex (+Process)	34.4	33.1	30.7
GPT-5.5	SRE	**CEG-Codex** (ours)	**56.2**	**41.2**	**40.7**
GPT-5.5	SEE	Codex (raw)	12.8	7.8	**15.1**
GPT-5.5	SEE	Codex (+Process)	2.1	15.1	**15.1**
GPT-5.5	SEE	**CEG-Codex** (ours)	**17.3**	**17.1**	16.1

Table 12: Per-source composite scores on CEG-Bench. Best score within each backbone–evaluator block is in **bold**.

Backbone	System	Small $\lvert\bar{V}_E\rvert \approx 9$	Medium $\lvert\bar{V}_E\rvert \approx 23$	Large $\lvert\bar{V}_E\rvert \approx 72$
Claude-Opus-4.7	Claude Code (raw)	16.6	10.9	14.2
Claude-Opus-4.7	Claude Code (+Process)	**23.0**	14.9	20.0
Claude-Opus-4.7	**CEG-Claude** (ours)	22.9	**17.0**	**21.4**
GPT-5.5	Codex (raw)	10.7	9.3	15.0
GPT-5.5	Codex (+Process)	9.1	3.9	**18.0**
GPT-5.5	**CEG-Codex** (ours)	**17.7**	**15.0**	17.8

Table 13: SEE composite by trace-complexity tercile. Best score within each backbone block is in **bold**.

tendency to hallucinate errors when given an event scaffold without sufficient diagnostic grounding; CEG-Agent mitigates this failure mode through structured graph construction and validation (§E.6).

E.3 Robustness to Trace Complexity

Table 13 groups CEG-Bench traces into terciles by gold event count and reports SEE composite scores. Under Claude-Opus-4.7, the +Process baseline is strongest on small traces, but degrades on medium traces and remains below CEG-Claude on medium and large traces. CEG-Claude shows a flatter curve, suggesting that incremental graph construction and schema-guided repair help stabilize diagnosis as traces grow longer. The effect is more pronounced under GPT-5.5. Codex (+Process) collapses on medium-length traces, dropping to 3.9 SEE, while CEG-Codex maintains 15.0, an $+11.1$ point gain in the regime where multiple errors must be assembled into a coherent causal structure. On large traces, CEG-Codex remains competitive with the best baseline. Overall, these results indicate that CEG-Agent is most beneficial when trace diagnosis requires sustained causal graph construction rather than isolated error spotting.

E.4 Dimension-Level Diagnosis

Tables 1 and 2 provide the dimension-level results; here we extend the analysis to understand where the framework gains come from.

Causal-structure recovery is the main source of gain. Across both backbone families, CEG-Agent improves the dimensions most directly tied to causal error reconstruction. The largest gains appear on SRE `main_chain`, where CEG-Claude improves over the Claude baselines by $+12.2$/$+12.8$

Backbone	System	SRE		SEE		Gold-empty	
		empty	non-empt	empty	non-empt	abstain	error
Claude-Opus-4.7	Claude Code (raw)	64.8	42.5	0.0	16.4	1/15	14/15
Claude-Opus-4.7	Claude Code (+Process)	**87.1**	43.0	0.0	22.7	11/15	4/15
Claude-Opus-4.7	**CEG-Claude** (ours)	68.6	**48.6**	0.0	**24.1**	0/15	14/15
GPT-5.5	Codex (raw)	59.9	27.1	0.0	13.7	0/15	15/15
GPT-5.5	Codex (+Process)	60.0	28.1	0.0	12.3	0/15	15/15
GPT-5.5	**CEG-Codex** (ours)	**77.1**	**41.1**	0.0	**19.9**	0/15	7/15

Table 14: Performance split by gold-empty and non-empty CEGs. `abstain` counts empty predictions; `error` counts predictions with at least one error on gold-empty traces.

points and CEG-Codex improves over the Codex baselines by $+14.1/+12.0$ points. CEG-Agent also improves the structurally exact dimensions that require precise error grounding and causal wiring, especially SEE `error`, `attached_to`, and `causes`. These dimensions directly measure whether a system can recover the process-level errors, ground them to trace events, and reconstruct the causal links through which they contribute to failures, which is precisely the role of CEG-Agent's validator-guided graph construction and schema projection.

Anomaly–error separation is a robust win. SRE `anomaly` improves from $44.7/46.0$ to 60.0 under Claude and from near-zero $2.0/1.7$ to 51.3 under GPT-5.5. This reflects the benefit of CEG-Agent's explicit anomaly–error–failure taxonomy. Baselines frequently collapse surface irregularities into process-level errors or omit anomalies altogether; the Codex baselines almost never emit anomaly nodes. In contrast, CEG-Agent uses a dedicated anomaly-detection step, allowing it to recover surface irregularities without conflating them with causally relevant errors.

Textual failure dimensions trade off only under Claude. CEG-Claude underperforms Claude Code (+Process) on the SRE textual dimensions `failure` and `attribution` (38.8 vs. 47.7 and 37.1 vs. 43.9). These dimensions rely on embedding-based textual similarity, where the gold annotations and +Process outputs are often concise, while CEG-Claude produces more detailed failure localization and explanatory descriptions. Although such detail can be useful for debugging, it may reduce similarity to compact gold references. The trade-off does not persist under structurally exact evaluation: CEG-Claude outperforms +Process on SEE `failure` by $+4.8$ points, and it disappears under GPT-5.5, where CEG-Codex leads both Codex baselines on SRE `failure` and `attribution`. This suggests a textual-style mismatch rather than a limitation of the CEG-Agent framework.

Where structural gaps remain. The main remaining gap is SEE `contributes_to`. CEG-Claude trails Claude Code (+Process) on this dimension (9.9 vs. 13.2), and although CEG-Codex leads its baselines, the margin is smaller than on other structural dimensions. This occurs because CEG-Agent often emits a richer set of error nodes, but does not always attach every newly introduced error to the failure node with an explicit CONTRIBUTES-TO edge. The schema projection restores reachability when needed, but exact tuple matching still rewards compact graphs with fewer missing or extra failure-link edges. Improving CONTRIBUTES-TO coverage is therefore a natural target for future refinement.

E.5 Gold-Empty Traces and Abstention Behavior

A subset of $15/100$ CEG-Bench traces, disproportionately from Tau2Bench ($10/36$), are retained with *empty* CEGs because the dual-adversary AAAP could not identify causal evidence from the trace alone (§B). These cases primarily test abstention rather than diagnosis: systems that emit empty graphs are mechanically rewarded, while systems that propose causal errors are penalized. Table 14 separates results on gold-empty and non-empty traces.

Abstention is backbone-dependent. Claude Code (+Process) is the only system that frequently abstains, returning an empty graph on $11/15$ gold-empty traces and reaching SRE 87.1 on this subset. This accounts for much of its Tau2Bench advantage over CEG-Claude in Table 12. By contrast,

Slice	System	SRE	error	edge	chain	fail.	attr.	anom.	event	cons.
Gold-empty ($n = 15$)										
empty	Claude Code (raw)	64.8	6.7	100.0	100.0	6.7	100.0	37.2	80.7	100.0
empty	Claude Code (+Process)	**87.1**	**73.3**	100.0	100.0	**73.3**	100.0	51.3	100.0	100.0
empty	**CEG-Claude**	68.6	6.7	100.0	100.0	0.0	100.0	**72.8**	100.0	99.9
empty	Codex (raw)	59.9	0.0	100.0	100.0	0.0	100.0	0.0	98.8	100.0
empty	Codex (+Process)	60.0	0.0	100.0	100.0	0.0	100.0	0.0	100.0	100.0
empty	**CEG-Codex**	**77.1**	**53.3**	100.0	100.0	0.0	100.0	**64.1**	100.0	100.0
Non-empty ($n = 85$)										
non-empty	Claude Code (raw)	42.5	45.3	26.0	28.0	**51.4**	**39.6**	46.0	77.7	100.0
non-empty	Claude Code (+Process)	43.0	45.0	31.5	27.2	43.2	34.0	45.1	100.0	100.0
non-empty	**CEG-Claude**	**48.6**	**53.8**	**32.2**	**42.4**	45.7	26.0	**57.7**	100.0	99.9
non-empty	Codex (raw)	27.1	26.9	19.8	14.6	35.2	12.4	2.4	97.8	100.0
non-empty	Codex (+Process)	28.1	31.8	22.9	17.0	25.3	10.1	2.0	100.0	100.0
non-empty	**CEG-Codex**	**41.1**	**44.6**	**27.1**	**31.2**	**41.1**	**14.8**	**49.0**	100.0	100.0

Table 15: **SRE** per-dimension scores split by gold-empty ($n = 15$) and non-empty ($n = 85$) subsets, with columns aligned to main-paper Table 1. On the empty subset the structural dimensions `edge`, `chain`, and `attr.` are mechanically 100 because gold and prediction are both empty, and the utility dimensions `event`/`cons.` are at or near ceiling for all systems; only `error`, `fail.`, and `anom.` are informative.

CEG-Claude almost always emits a diagnosis, and both Codex baselines emit errors on all gold-empty traces.

CEG-Agent improves on substantive traces. On the $n = 85$ non-empty traces, where causal diagnosis is required, both CEG-Agent variants achieve higher composite scores than their same-backbone baselines. CEG-Claude improves by $+5.6$ SRE and $+1.4$ SEE, while CEG-Codex improves by $+13.0$ SRE and $+6.2$ SEE. This indicates that the framework's advantage is concentrated on traces with substantive causal structure, and that the Claude-family Tau2Bench regression is mainly an abstention artifact rather than a diagnostic failure. A natural improvement is to add a calibrated empty-graph option to the failure-analysis stage, allowing CEG-Agent to abstain when no causal error is supported by the trace. We leave this out deliberately: for generality, we want CEG-Agent to focus on diagnosing and constructing CEGs for substantive traces, rather than tuning its behavior around a separate empty-case classification task.

Per-dimension breakdown on empty and non-empty subsets. Tables 15 and 16 provide the per-dimension breakdown. On the gold-empty subset, several SRE dimensions are mechanically saturated because the gold graph contains no causal structure; the most informative columns are therefore `error`, `failure`, and `anomaly`. Claude Code (+Process) scores highly on this subset largely because it abstains on most gold-empty traces. SEE is zero for all systems on the empty subset because there are no gold errors, failures, or causal edges to match.

On the non-empty subset, CEG-Agent shows stronger composite performance and improves most structure-oriented dimensions. Under Claude, CEG-Claude improves over Claude Code (+Process) on SRE `error`, `causal_edge`, `main_chain`, and `anomaly`, as well as SEE `error`, `attached_to`, and `causes`. Some exceptions remain, especially textual SRE dimensions such as `attribution` and the SEE `contributes_to` dimension, consistent with the style and edge-coverage effects discussed in §E.4. Under GPT-5.5, the pattern is stronger: CEG-Codex improves the composite scores and leads most structural and anomaly-related dimensions over Codex (+Process). Overall, removing gold-empty traces clarifies that CEG-Agent's main advantage lies in substantive causal graph construction rather than abstention behavior.

Slice	System	SEE	failure	error	attach	cause	contrib.
Gold-empty ($n = 15$)							
empty	Claude Code (raw)	0.0	0.0	0.0	0.0	0.0	0.0
empty	Claude Code (+Process)	0.0	0.0	0.0	0.0	0.0	0.0
empty	**CEG-Claude**	0.0	0.0	0.0	0.0	0.0	0.0
empty	Codex (raw)	0.0	0.0	0.0	0.0	0.0	0.0
empty	Codex (+Process)	0.0	0.0	0.0	0.0	0.0	0.0
empty	**CEG-Codex**	0.0	0.0	0.0	0.0	0.0	0.0
Non-empty ($n = 85$)							
non-empty	Claude Code (raw)	16.4	**66.0**	6.2	4.6	1.2	3.8
non-empty	Claude Code (+Process)	22.7	55.7	21.8	19.1	1.3	**15.5**
non-empty	**CEG-Claude**	**24.1**	61.3	**24.1**	**20.9**	**2.4**	11.7
non-empty	Codex (raw)	13.7	51.6	6.6	7.0	0.0	3.6
non-empty	Codex (+Process)	12.3	39.2	10.2	7.0	0.0	4.8
non-empty	**CEG-Codex**	**19.9**	**56.9**	**17.9**	**15.3**	**1.8**	**7.5**

Table 16: **SEE** per-dimension scores split by gold-empty ($n = 15$) and non-empty ($n = 85$) subsets. On the empty subset SEE is 0 for every system because the gold graph contains no errors, failures, or edges to match.

Backbone	System	$\lvert\bar{V}_E\rvert$	$\lvert\bar{V}_R\rvert$	$\lvert\bar{V}_F\rvert$	$\lvert\bar{V}_A\rvert$	$\lvert\bar{E}\rvert$	$\lvert\bar{E}_C\rvert$
Reference	Gold	34.9	3.91	1.25	2.53	45.4	7.49
Claude-Opus-4.7	Claude Code (raw)	10.7	3.14	1.02	1.20	18.2	5.30
Claude-Opus-4.7	Claude Code (+Process)	34.9	2.09	0.74	1.24	39.5	3.56
Claude-Opus-4.7	**CEG-Claude** (ours)	34.9	**4.69**	**1.06**	**2.96**	**45.4**	**6.87**
GPT-5.5	Codex (raw)	34.9	1.64	1.00	0.07	37.2	1.64
GPT-5.5	Codex (+Process)	34.9	1.82	1.00	0.10	37.5	1.82
GPT-5.5	**CEG-Codex** (ours)	34.9	**4.49**	1.00	**2.16**	**43.0**	**4.65**

Table 17: Mean per-trace graph sizes. Bold marks the closest within-backbone system to the gold reference.

E.6 Graph-Size and Structural Calibration

Table 17 compares each system's output graph size with the gold reference before any matching. The baselines exhibit three calibration failures. First, Claude Code (raw) parses far fewer events than the gold trace representation, which limits downstream graph construction. Second, all baselines under-generate causal structure: Codex variants produce only 1.64–1.82 causal-subgraph edges per trace compared with the gold average of 7.49, and almost never emit anomalies. Third, even Claude Code (+Process), despite receiving the full event scaffold, under-generates errors, failures, anomalies, and causal edges.

In contrast, both CEG-Agent variants produce graphs much closer to gold scale. CEG-Claude nearly matches the gold total edge count and approaches the gold causal-edge count, while CEG-Codex substantially increases errors, anomalies, and causal edges over the Codex baselines. This calibration explains many of the structural-dimension gains in §E.4: baselines often fail before matching because they do not propose enough causally relevant nodes and edges, whereas CEG-Agent's incremental construction and validation produce graphs at a scale suitable for causal error reconstruction.

E.7 Per-Mechanism and Per-Role Recall

Table 18 shows where CEG-Agent improves error recovery within the controlled vocabulary. Both CEG-Agent variants achieve the best within-backbone recall on most mechanisms and on all three structural roles. The largest gains occur on cumulative failure modes such as *representation*, *evidence*

Slice	n_{gold}	Claude-Opus-4.7			GPT-5.5		
		CC-raw	CC-proc	CEG-Cl.	Cx-raw	Cx-proc	CEG-Cx
Mechanism recall on gold (ε, μ) keys							
planning	77	11.7	14.3	**23.4**	0.0	1.3	**7.8**
evidence integration	72	1.4	11.1	**20.8**	0.0	0.0	**13.9**
execution	66	3.0	18.2	**22.7**	3.0	3.0	**12.1**
self-evaluation	61	1.6	**23.0**	**23.0**	14.8	18.0	**19.7**
control	53	1.9	34.0	**35.8**	24.5	20.8	**39.6**
omission	33	9.1	**18.2**	15.2	0.0	0.0	**6.1**
representation	25	0.0	12.0	**20.0**	0.0	0.0	0.0
environment	4	0.0	0.0	0.0	0.0	0.0	0.0
Structural-role recall on matched errors							
ROOT	130	10.0	21.5	**23.1**	2.3	2.3	**13.1**
PROPAGATED	155	1.9	18.7	**27.1**	6.5	9.0	**16.1**
AMPLIFICATION	106	0.9	14.2	**17.9**	10.4	7.5	**16.0**

Table 18: Recall (%) by error mechanism and structural role. Best score within each backbone family is in **bold**.

System	Completio	Correct.	Constraint	Efficiency	Safety	Other	Coverage
Gold	53	43	17	6	3	3	6/6
Claude-Opus-4.7							
Claude Code (raw)	62	21	19	0	0	0	3/6
Claude Code (+Process)	49	14	10	1	0	0	4/6
CEG-Claude (ours)	42	33	21	5	0	5	**5/6**
GPT-5.5							
Codex (raw)	40	33	25	0	0	2	4/6
Codex (+Process)	35	59	0	6	0	0	3/6
CEG-Codex (ours)	28	50	6	0	0	16	4/6

Table 19: Failure-type emission counts across 100 traces. Coverage is the number of non-empty failure types emitted.

integration, and *planning*, which require aggregating evidence across multiple trace steps rather than detecting a single local mistake.

The Claude-family results show that CEG-Claude improves recall on nearly every mechanism, with the main exception of *omission*, where Claude Code (+Process) is slightly stronger, and *environment*, which has only four examples and is missed by all systems. Under GPT-5.5, the contrast is sharper: Codex baselines are near zero on several mechanisms, while CEG-Codex recovers non-trivial recall on *planning*, *evidence integration*, *execution*, *control*, and *omission*. The role-level results are particularly revealing: CEG-Agent's largest gains are on PROPAGATED errors, which require identifying downstream errors together with their upstream causal predecessors. This supports the central role of the typed causal subgraph G_C in recovering error propagation rather than only isolated root causes.

E.8 Failure-Type Coverage

Table 19 summarizes failure-type coverage. CEG-Claude covers five of the six failure categories, including the rare *efficiency* and *other* buckets that most baselines ignore. CEG-Codex also broadens the GPT-5.5 emission distribution, partially correcting the degenerate behavior of Codex (+Process), which emits no *constraint* failures across the benchmark. Three patterns stand out. First, most systems overuse *completion* as a default failure type when uncertain, while CEG-Agent is less concentrated in this bucket. Second, *safety* failures are missed by all systems, despite three gold instances, suggesting

Gold	CEG-Claude prediction			CEG-Codex prediction		
	LOW	MEDIUM	HIGH	LOW	MEDIUM	HIGH
LOW	**34**	13	5	**39**	0	0
MEDIUM	14	**7**	9	17	**0**	0
HIGH	1	1	**0**	0	0	**1**
Column total	49	21	14	56	0	1

Table 20: Absolute repair-value confusion on (ε, μ)-matched errors. Diagonal cells indicate exact bucket agreement.

System	Matched errors	Pairwise agree.	Kendall's τ	Bucket acc.
Claude Code (+Process)	65	0.528	$+0.056$	0.062
Codex (+Process)	23	0.500	0.000	0.000
CEG-Claude	84	**0.700**	$\mathbf{+0.400}$	0.488
CEG-Codex	57	0.500	0.000	**0.702**

Table 21: Relative repair-value ranking on (ε, μ)-matched errors. Pairwise agreement measures whether predicted priorities preserve the gold ordering within each trace.

that safety-specific diagnosis may require an explicit policy or oracle beyond trace-only evidence. Third, CEG-Agent generally produces a more diverse failure-type distribution, which is consistent with its explicit separation of outcome-level failures from process-level errors.

E.9 Repair-Value Calibration

Repair-value prediction is one of the most difficult components of the pipeline. Directly comparing predicted buckets with gold buckets is affected by severe class imbalance: the gold distribution is long-tailed, with only 3.3% of errors labeled HIGH, so predicting LOW for most errors can obtain deceptively high bucket accuracy. For downstream debugging, a more useful question is whether a system can *rank* errors by repair priority within a trace, since repair values are primarily intended to help decide which error to fix first. We therefore report both absolute bucket confusion and relative ranking quality in Tables 20 and 21.

Relative ranking is more informative than absolute buckets. Table 21 evaluates whether predicted repair values preserve the relative priority ordering of gold errors within the same trace. This view is more informative than absolute bucket accuracy. Codex (+Process) and CEG-Codex both have Kendall's $\tau = 0.0$, indicating that their predictions carry little priority-ordering information, even though CEG-Codex achieves the highest bucket accuracy by predicting almost all errors as LOW. In contrast, CEG-Claude achieves substantially better pairwise agreement and a positive Kendall's $\tau = +0.400$, suggesting that it captures some useful within-trace priority structure even when its absolute bucket calibration is imperfect.

Backbone-specific calibration patterns. The two CEG-Agent variants fail in different ways. CEG-Claude over-predicts HIGH: 20.0% of its errors are assigned HIGH, compared with a gold prevalence of 3.3%. However, its positive ranking score suggests that its underlying severity ordering is partially useful, and that the main problem is bucket-threshold calibration. CEG-Codex shows the opposite failure mode: it collapses almost entirely to LOW, never emits MEDIUM, and therefore obtains high absolute accuracy without providing meaningful repair prioritization. Thus, bucket accuracy alone is misleading for repair-value evaluation.

Source of calibration error. The calibration error mainly comes from two signals that are sensitive to LLM writing style: the counterfactual severance signal $\sigma(r, f^\star)$, which depends on how the model describes causal bottlenecks, and the lexical hedging signal $h(r)$, which varies with confident or cautious wording. Future work may compute $\sigma(r, f^\star)$ deterministically from the graph, constrain the

Cohort	n	Tau2Bench	SWE-bench	BrowseComp
All systems have SEE $= 0$	19	10	3	6
Gold-empty subset	15	10	3	2
Non-empty all-fail subset	4	0	0	4

Table 22: Traces on which all six systems obtain SEE $= 0$, decomposed by source.

CEG-Agent backbone	SRE	Structural			Textual		
		error	edge	chain	fail.	attr.	anom.
Claude-Opus-4.7	**51.6**	**46.7**	**42.4**	**51.0**	38.8	**37.1**	**60.0**
GLM-5.1	46.7	45.1	38.0	39.7	**41.9**	37.0	42.8
GPT-5.5	46.5	45.9	38.0	41.5	34.9	27.6	51.3
Gemini-3.1-Pro	44.2	40.8	41.0	37.3	35.7	26.5	43.1
Qwen3.7-Max	43.7	41.7	37.8	37.8	34.2	32.8	35.2

Table 23: Per-dimension **SRE** scores of CEG-Agent under five LLM backbones. Best score per column is in **bold**. Utility dimensions (`event`, `cons.`) are at ceiling for all backbones and are omitted; the composite SRE in the second column already incorporates them.

influence of $h(r)$, or tune bucket thresholds for specific downstream applications. We do not further study fine-grained calibration here, since CEG-Claude already provides useful relative ordering in the current evaluation.

E.10 Inter-System Agreement and Irreducibly Hard Traces

We further examine inter-system agreement by pooling per-trace SEE scores from all six evaluated systems. As shown in Table 22, there are 19 traces on which every system obtains SEE $= 0$. Most of these cases are the gold-empty traces analyzed in §E.5, where SEE becomes degenerate because the gold graph contains no causal structure to recover. The remaining four cases are non-empty BrowseComp traces, forming a small irreducibly hard cohort for the current systems. Manual inspection of these four cases suggests a common pattern: the failure often arises from global evidence integration across a long research trajectory, rather than from a single locally identifiable error event. Such cases are especially difficult for structurally exact evaluation because the relevant causal evidence is distributed across multiple evidence-gathering and reasoning steps. This explains why even strong systems fail to recover the exact annotated structure on these traces. At the other end of the spectrum, the easiest traces are mostly from Tau2Bench: among the 10 traces with maximum SEE at least 0.58 under some system, 8 come from Tau2Bench. This source-level contrast reinforces the broader pattern in §E.2: Tau2Bench is structurally easier and more affected by abstention behavior, whereas SWE-bench and BrowseComp require richer causal reconstruction and better expose the benefits of CEG-Agent.

E.11 Per-Backbone Diagnostic Capability Profile

This subsection expands §5.3 with a per-dimension breakdown of CEG-Agent under five LLM backbones: Claude-Opus-4.7, GPT-5.5, Gemini-3.1-Pro, Qwen3.7-Max, and GLM-5.1. The schema, tool set, and inference budget are fixed across backbones, so the differences below reflect backbone-specific diagnostic behavior under the same CEG-Agent framework. Tables 23 and 24 report the SRE and SEE breakdowns.

Claude-Opus-4.7 is the most balanced diagnostician. Claude achieves the highest composite SRE and SEE scores, leads most SRE structural dimensions, and has no clear weak dimension relative to the other backbones. Its largest advantage is on SRE `anomaly` (60.0), indicating reliable use of the anomaly-detection component. It also achieves the strongest SRE `main_chain` score (51.0), suggesting that it is particularly effective at organizing local errors into a coherent causal propagation chain.

CEG-Agent backbone	SEE	Dimensions				
		failure	error	attach	cause	contrib.
Claude-Opus-4.7	**20.5**	52.1	**20.5**	**17.8**	2.0	9.9
GLM-5.1	19.6	**54.9**	18.0	15.0	1.3	8.7
Qwen3.7-Max	18.7	46.1	18.9	16.1	**2.3**	**10.2**
GPT-5.5	16.9	48.3	15.2	13.0	1.5	6.3
Gemini-3.1-Pro	15.0	42.0	15.5	11.4	1.3	4.8

Table 24: Per-dimension **SEE** scores of CEG-Agent under five LLM backbones. Best score per column is in **bold**.

Metric	CEG-Claude	CEG-Codex	CEG-Gemini	CEG-Qwen	CEG-GLM
Mean per trace					
LLM calls	18.7	19.0	23.5	23.2	24.1
Agent iterations	13.5	14.0	18.2	18.0	18.8
Input tokens	556,039	350,076	467,519	622,982	436,678
Output tokens	3,832	4,832	12,567	19,063	20,332
Wall-clock time (s)	95.0	160.0	372.6	214.4	517.5

Table 25: Cost profile of CEG-Agent under five LLM backbones on CEG-Bench.

GLM-5.1 is strong on exact reconstruction and failure typing. GLM-5.1 is the strongest non-Claude backbone on SEE and is competitive with GPT-5.5 on SRE. It leads all backbones on SRE `failure` (41.9) and SEE `failure` (54.9), and nearly matches Claude on SRE `attribution`. Its main weaknesses are SRE `anomaly` and `main_chain`, both of which remain below Claude. The cost and tool-use profile in §E.12 suggests that GLM relies more heavily on validator and critic feedback, consistent with a model that can recover strong exact structure after schema violations are surfaced. Overall, GLM appears strong at identifying failure categories and exact structure, but less effective at recovering the broader anomaly and propagation-chain context.

GPT-5.5 is competitive on error identification but weaker on structure. GPT-5.5 closely tracks Claude on SRE `error` (45.9 vs. 46.7), showing that it can identify individual process-level errors reasonably well. However, it loses ground on SRE `main_chain` and `attribution`, and also trails Claude on SEE `error`, `attached_to`, and `contributes_to`. This indicates that GPT-5.5 is better at describing local errors than at wiring them into an exact causal structure.

Gemini-3.1-Pro is balanced on causal edges but conservative on contribution. Gemini achieves a competitive SRE `edge` score (41.0), close to Claude, but obtains the lowest SEE `contributes_to` score. This suggests that Gemini often captures local causal links between errors, but is more conservative or incomplete when linking errors to outcome-level failures. Its lower SEE `failure` score also indicates weaker alignment with the controlled failure-type vocabulary.

Qwen3.7-Max suppresses anomalies but performs well on exact causal links. Qwen has the lowest SRE `anomaly` score (35.2), suggesting a tendency to collapse surface irregularities into process-level errors rather than emitting separate anomaly nodes. However, it performs strongly on exact structural dimensions: it achieves the best SEE `cause` and `contributes_to` scores, and ranks third overall on SEE. Thus, although Qwen has lower semantic similarity, its predicted causal links are often well formed under exact tuple matching.

SRE and SEE capture different capabilities. The SRE and SEE rankings diverge beyond the top model. In particular, Qwen3.7-Max outperforms both GPT-5.5 and Gemini-3.1-Pro on structurally exact reconstruction despite lower semantic similarity. This shows that fluent diagnostic descriptions and exact causal graph reconstruction are not the same capability. More broadly, composite scores can hide important differences across dimensions: backbones with similar overall scores may fail in

Tool	CEG-Claude	CEG-Codex	CEG-Gemini	CEG-Qwen	CEG-GLM
find_failures	108	101	99	106	101
propose_errors	102	100	96	124	110
find_anomalies	101	100	94	103	100
build_causal_graph	102	100	87	106	100
estimate_repair_values	100	101	88	96	90
validate_graph	112	136	98	111	121
apply_critic_patch	41	71	103	74	94
spawn_subagent	5	13	32	6	9

Table 26: Total agent-issued tool invocations across the evaluated traces for each backbone. Infrastructure tools (read_trace, build_events, render_graph, finalize) and scaffolding tools (write_todo, inspect_state) are omitted from the table, since they are not central to the tool-use analysis.

different ways. This supports reporting both per-dimension and per-backbone results when evaluating systems on CEG-Bench.

Implications for backbone choice. These results suggest that backbone choice should depend on the intended use of the CEG. Claude-Opus-4.7 is the strongest overall option, especially when both semantic quality and structural fidelity matter. GLM-5.1 is a strong non-Claude alternative for exact reconstruction and failure typing. GPT-5.5 remains competitive on local error identification but is weaker on propagation structure. Gemini is better suited when local causal-edge recovery is more important than failure-link completeness, while Qwen is useful when exact causal links matter more than anomaly coverage. Overall, CEG-Agent is not tied to a single LLM family, and CEG-Bench exposes meaningful backbone-specific diagnostic profiles that are obscured by aggregate scores alone.

E.12 Cost and Tool-Use Profile

Tables 25 and 26 summarize the cost and tool-use profile of CEG-Agent across five backbones. Cost varies substantially by backbone. Claude-Opus-4.7 and GPT-5.5 converge in about 13–14 iterations with roughly 19 LLM calls per trace, while Gemini, Qwen, and GLM require more iterations and approach the budget more often on long traces. Across all backbones, input tokens dominate output tokens because the normalized trace is repeatedly attached during iterative diagnosis. Wall-clock time is governed not only by token count but also by provider latency: GLM is the slowest despite moderate input-token usage, while Qwen uses the most input tokens but has lower latency than GLM.

Tool-use patterns are stable for the core diagnosis tools: all backbones call find_failures, propose_errors, find_anomalies, build_causal_graph, and estimate_repair_values at roughly one call per trace. The main differences appear in optional refinement behavior. apply_critic_patch is used much more often by Gemini and GLM than by Claude, indicating that these backbones rely more heavily on critic-guided repair. spawn_subagent is most frequent under Gemini, suggesting greater delegation on difficult traces. Overall, stronger backbones reach high-quality CEGs with fewer repair and delegation steps, while less stable backbones make greater use of CEG-Agent's validator and critic loop to recover from imperfect first drafts.

End-to-end monetary cost. CEG-Agent performs diagnosis as a one-time offline procedure for each failed trace, so its cost should be compared with running the underlying coding agent on the same trace. Under the Claude-Opus-4.7 backbone, the total billed cost over the full benchmark is $300 for Claude Code (raw) and $244 for Claude Code (+Process), whereas **CEG-Claude costs $225**. Thus, the iterative diagnostic procedure does not increase end-to-end cost relative to the coding-agent baselines. The main reason is that the normalized trace is reused across diagnostic steps and is therefore cache-friendly. In addition, CEG-Agent emits compact structured graph updates, averaging about 3.8k output tokens per trace, while free-form baselines spend more tokens on unstructured exploration and long diagnostic reports.

Feedback	Tau2	Browse	SWE
generic	77.8	6.9	14.3
CEG: failure	**91.7**	10.3	**17.1**
CEG: errors	80.6	**17.2**	**17.1**
CEG: priority errors	83.3	**17.2**	8.6

Table 27: Closed-loop repair pass@1 (%) under the feedback conditions used in the main paper.

F Closed-Loop Repair Study

This appendix provides additional details for the closed-loop repair probe in §5.4. The goal of this study is not to claim a fully optimized repair system, but to test whether CEGs contain actionable information that can help a model recover from a previously failed execution.

Protocol. All 100 traces in CEG-Bench are failed attempts, so the original runs have zero success by construction and are omitted from the tables. For each trace, we re-attempt the original task once with a feedback hint derived from the AAAP annotation. We compare a content-free **generic** retry note with three compact CEG-based hints: **CEG: failure**, which exposes only the outcome-level failure; **CEG: errors**, which exposes the causal error nodes with their mechanisms, roles, and descriptions; and **CEG: priority errors**, which exposes only errors with high or medium repair value. The repair-value filter is intended to test whether the counterfactual repair signal can identify a smaller and more actionable subset of errors.

The backbone is fixed to GLM-4.6 with temperature 0. On **Tau2Bench** ($n = 36$), the model is re-run as the agent in the real Tau2 simulator with the feedback injected into the domain policy, and success is measured by the task reward. On **BrowseComp** ($n = 29$), the model re-derives an answer from the failed trace and the feedback without new search, and the answer is judged by the BrowseComp/HLE grader. On **SWE-bench** ($n = 35$), the model generates an oracle-file patch from the issue and gold-patch-touched files at `base_commit`, and the patch is graded by the official test-based harness.

Overall results. Table 27 shows that CEG feedback is most useful when the agent can act on it interactively. On Tau2Bench, all CEG-derived hints outperform the generic retry, and the failure-level hint gives the largest gain (91.7 vs. 77.8). This indicates that naming the outcome-level failure provides a concrete correction target for re-execution. The two error-based hints also improve over generic feedback, suggesting that process-level causal errors can guide the agent toward more appropriate next actions.

The pattern differs in the two single-shot settings. On BrowseComp, error-based hints achieve the best score (17.2), while the failure-only hint is weaker (10.3). This is consistent with the nature of deep-research tasks: merely knowing that the final answer is wrong is often insufficient, whereas causal errors can point the model toward reasoning or evidence-integration mistakes. However, the absolute success rate remains low because the setting does not allow new retrieval; if the failed trace never contains the missing evidence, the answer often cannot be recovered by re-reading the trace alone. On SWE-bench, concise failure and error hints slightly improve over generic feedback, but priority-error filtering hurts. This suggests that the current repair-value signal is not yet robust enough for single-shot code patching, where filtering out medium-context errors may remove information needed to produce an applicable and complete patch.

SWE-bench patch behavior. Table 28 explains why CEG feedback is less stable for single-shot code repair. The concise failure and error hints each solve one additional instance over the generic retry, but they do so through different behaviors. The error hint keeps patch complexity moderate and has a lower apply-failure count than the failure-only hint. The failure-only hint induces larger patches and more test-file edits, which increases apply failures but also reduces the number of apply-but-fail-test cases among the patches that do apply. In contrast, priority-error filtering performs worse than generic feedback. Although it is intended to focus the model on the most repair-relevant errors, it can remove contextual errors that are necessary for constructing a correct patch, leading the model to over-focus on an incomplete local fix.

Feedback	pass@1	apply-fail	test-fail	churn	test%
generic	14.3	13	17	5.6	9
CEG: failure	**17.1**	19	10	16.5	17
CEG: errors	**17.1**	14	15	9.7	14
CEG: priority errors	8.6	19	13	10.0	9

Table 28: SWE-bench closed-loop breakdown ($n = 35$). `apply-fail` counts patches that fail to apply, `test-fail` counts patches that apply but fail tests, `churn` is the mean number of changed lines, and `test%` is the percentage of patches that edit test files.

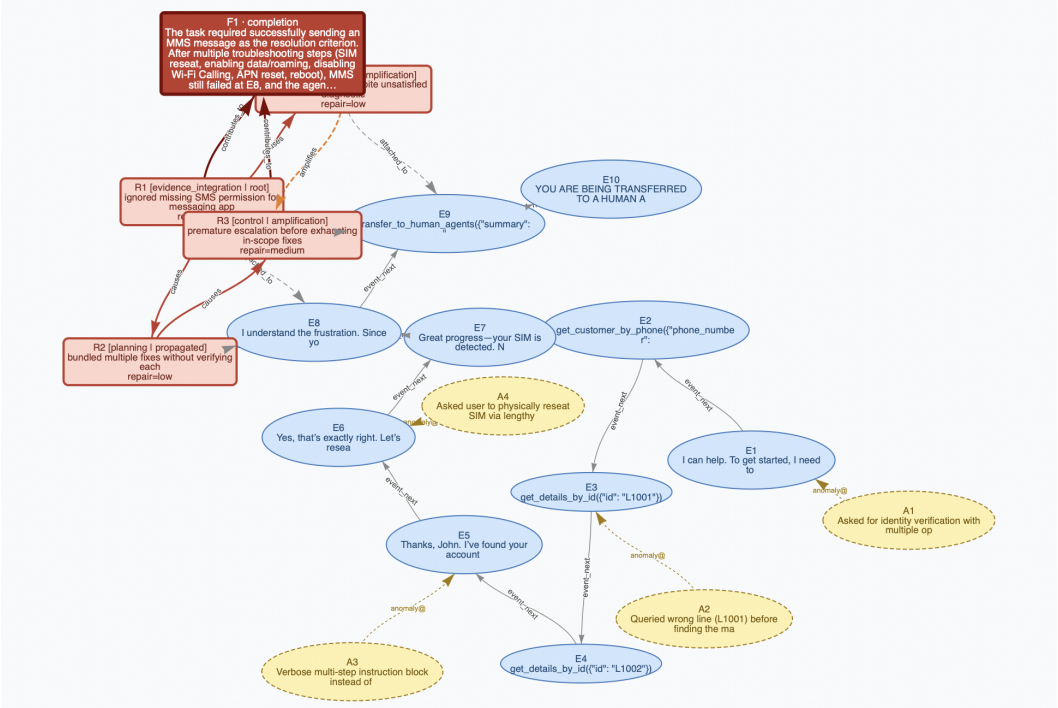


Figure 5: CEG-Claude visualization for a Tau2Bench trace.

These results highlight an important distinction between diagnosis and repair. A CEG can identify causally relevant errors and rank their likely repair value, but the best way to expose this information to a downstream agent depends on the execution setting. Interactive agents can use a compact failure or error signal to adjust future actions, while single-shot patch generators may require broader context to avoid brittle or incomplete edits. Thus, CEGs provide useful repair information, but feedback formatting and repair-value calibration remain open design problems.

Scope. The closed-loop study is intentionally small, with 29–36 traces per source benchmark, so the results should be interpreted as directional rather than statistically decisive. Its main value is to show that CEGs are not only evaluative artifacts: they can also function as structured feedback for downstream recovery. The strongest and most consistent gains appear in the interactive Tau2Bench setting, where the agent can directly act on the diagnostic signal. We leave larger-scale closed-loop optimization, better feedback formatting, and more robust repair-value calibration to future work.

F.1 Case Studies

Figures 5–7 show representative HTML visualizations produced by CEG-Claude on traces from Tau2Bench, SWE-bench, and BrowseComp. Each visualization aligns the chronological event se-

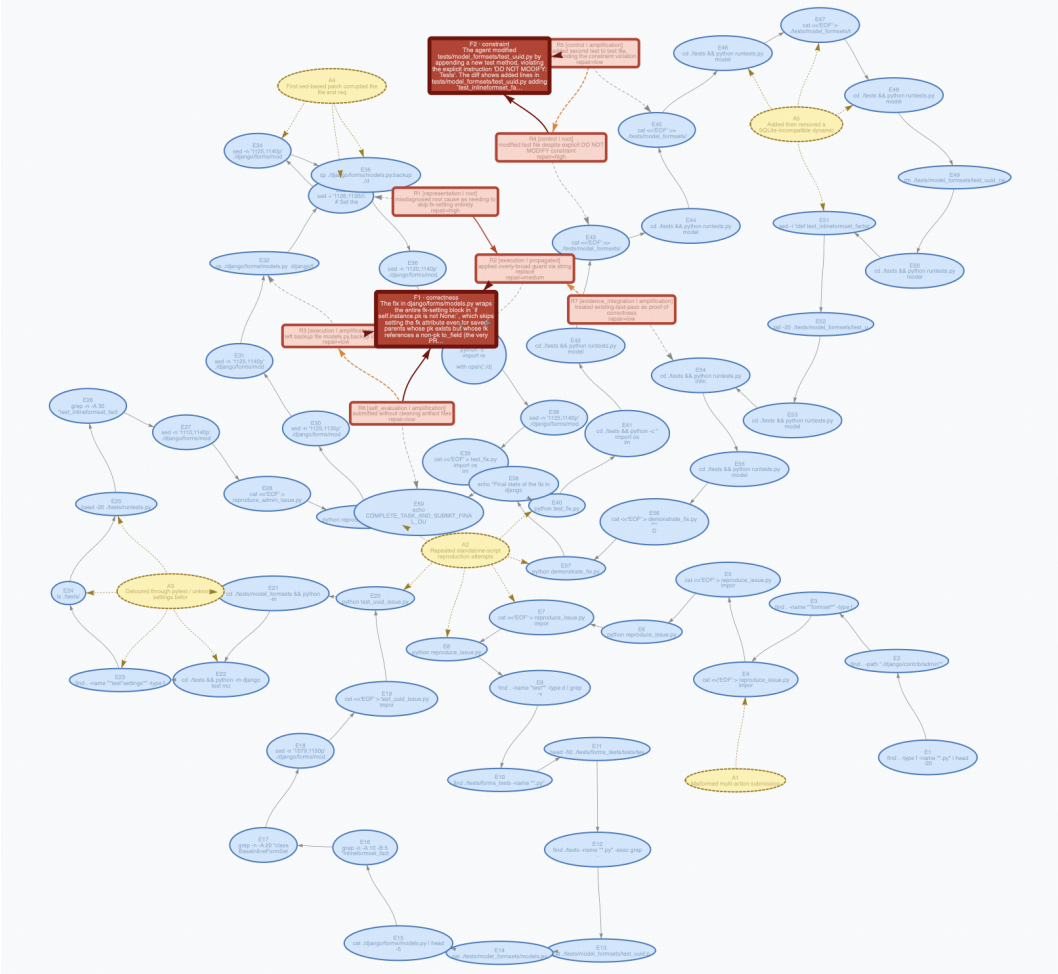


Figure 6: CEG-Claude visualization for a SWE-bench trace.

quence with the induced causal error graph, making errors, anomalies, propagation chains, and outcome-level failures inspectable in a single view. These examples illustrate that CEG-Agent produces not only machine-evaluable CEGs, but also human-readable diagnostic artifacts for debugging long agentic traces.

G LIMITATIONS

This work has several limitations. First, the released CEG-Bench annotations are produced automatically by AAAP rather than by exhaustive human annotation. We validate them against an 8-expert human gold set (CEG-Bench-Gold, §D), which agrees closely with the automatic annotations (CEG-Sim $= 71.5$; error/failure/anomaly $76.3/76.3/88.1$), but the AAAP annotations may still contain systematic biases shared by the agentic annotators. In particular, some failed traces are retained with empty CEGs because no clear causal error can be inferred from the trace alone without external outcome feedback. We make this choice to preserve the generality and reproducibility of CEG construction, but acknowledge that human experts might adjudicate some cases differently. We therefore view AAAP as a scalable, now human-validated route toward automated construction of complex causal-diagnosis benchmarks. Second, the benchmark scope is limited. CEG-Bench contains 100 traces from three representative agentic benchmarks, covering deep research, function calling, and agentic coding, and all candidate traces are drawn from a single source model (GLM-4.6). We note that the source model only determines which raw traces enter the candidate pool: the difficulty of

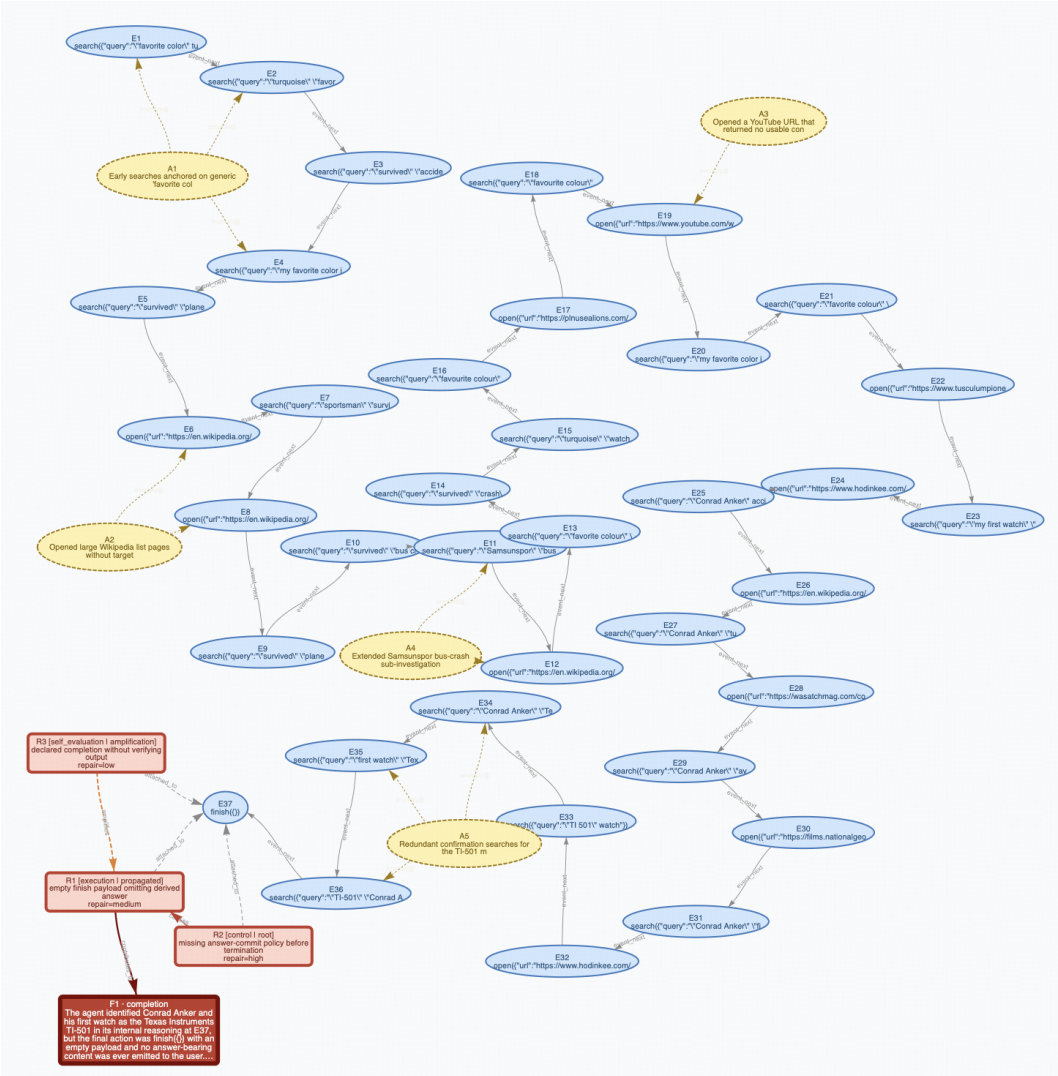


Figure 7: CEG-Claude visualization for a BrowseComp trace.

recovering the gold graph is a property of the trace content, and any error or failure a trace exhibits is expressible within our taxonomy, so changing the generator enlarges the object set rather than altering our conclusions. It does not cover all deployment settings, safety-critical domains, or multi-agent environments. Extending CEG-style diagnosis to more frontier source models and broader tasks remains future work. Third, repair-value estimation remains preliminary. We emphasize that the *gold* repair values in CEG-Bench are computed by a deterministic graph-structural functional (§A.6), *not* by an LLM; the calibration difficulty is specific to the inference-time LLM estimate, whose buckets can be affected by model-specific wording and confidence. We therefore recommend the deterministic graph score as the primary prioritization signal and treat the LLM bucket as a fallback; even when bucket thresholds are miscalibrated the underlying ordering can remain useful (CEG-Claude attains Kendall's $\tau = +0.40$, §E.9). We leave robust cross-backbone repair-value calibration to future work. Finally, due to computational and API cost constraints, our baseline comparison focuses on Claude Code and Codex, and our cross-backbone study uses five representative LLMs. Broader comparisons across more agent frameworks, LLM families, and deployment environments are left for future work.