CausaLab: A SCALABLE ENVIRONMENT FOR INTERACTIVE CAUSAL DISCOVERY TOWARD AI SCIENTISTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce *CausaLab*, a scalable environment for evaluating interactive causal discovery by LLM agents. Unlike prior evaluations, *CausaLab* evaluates both whether an agent can solve a problem using causal evidence and whether its answer is grounded in a faithful recovered causal mechanism. Each episode hides a randomly sampled structural causal model (SCM) that no prior knowledge supplies: the agent reads prior records, intervenes on a manipulator crystal, and predicts the held-out frequency of a reactor crystal built from the same SCM. Because the reactor exposes every property except `frequency`, the prediction needs only that variable's parents and coefficients, so we score this Target Mechanism Recovery separately from Full System Discovery. Experiments show that correct prediction does not imply Full System Discovery. Across matched controls on functional form, hidden perturbations, and graph structure, the two levels move independently or in opposite directions. From ten observations alone on 6-node graphs, `GPT-5.2-high` recovers the parents of `frequency` exactly and answers 86% of reactors correctly, yet reaches 0.471 all-edge F_1. The same ten rounds spent on interventions lift all-edge F_1 from 0.47 to 0.82 for ten accuracy points. Failures come from premature commitment, not exhausted budget, and a consistency check recovers part of the loss. A shift control on a quarter of the agents' budget recovers every main-suite graph exactly, so the gap measures experiment design. *CausaLab* therefore separates the two levels and exposes current LLM agents' limits as experimental causal reasoners. We release the full SCM dataset and benchmark code to advance future research.

1 INTRODUCTION

Causal reasoning matters because scientific, medical, and policy decisions depend on how systems respond to interventions, not only on observed associations (Pearl, 2009; Pearl & Mackenzie, 2018; Imbens & Rubin, 2015), yet measuring it remains challenging for today's large language models (LLMs). Existing benchmarks generally translate causal graphs, datasets, or narratives into question-answering and classification tasks (Qin et al., 2019; Romanou et al., 2023; Stolfo et al., 2023; Jiang et al., 2024; Vashishtha et al., 2025; Jin et al., 2023a; Wang, 2024; Chen et al., 2024b; Jin et al., 2023b), leaving open the "causal parrot" concern (Zečević et al., 2023): models can succeed with memorized causal facts or linguistic cues rather than the causal reasoning behaviors needed to discover mechanisms (Zheng et al., 2023; Liu et al., 2023).

To illustrate, consider a thought experiment. Suppose we want to study the causal relationship between temperature and the resonance frequency of a crystal. An LLM agent might appear useful in two quite different ways. (1) It may retrieve from existing sources, such as Wikipedia or its training data, that temperature causes resonance frequency. (2) It may observe paired measurements of temperature and frequency, formulate hypotheses, design experiments, perform interventions, observe the resulting changes, and infer causation from evidence (Pearl, 2009; Hauser & Bühlmann, 2012; Lampinen et al., 2023). Both are valuable in practice. But (1) offers little help once the relevant causal knowledge lies beyond the current frontier of human knowledge. We therefore argue that (2) matters more for applications such as scientific discovery, because it is what lets an agent help push that frontier the way a human scientist would (Langley, 2019; Dunbar & Fugelsang, 2005; Jansen et al., 2024).

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

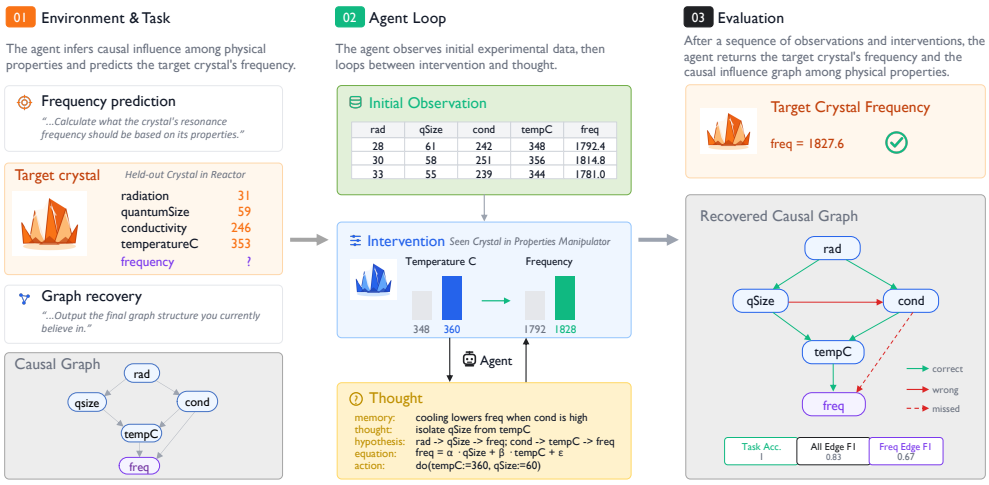


Figure 1: Overview of a *CausaLab* episode: a hidden SCM generates prior records, a manipulator crystal, and a held-out reactor crystal; the agent observes records, performs budgeted interventions, and emits per-step DSL hypotheses parsed against the ground-truth SCM; we score Target Mechanism Recovery and Full System Discovery along the hypothesis trajectory.

We introduce *CausaLab* (Figure 1), an environment for evaluating LLM agents as interactive causal discoverers. Each episode asks the agent to use evidence from prior records and its own interventions on one crystal to predict the held-out frequency of another. The shared data-generating mechanism is a hidden structural causal model (SCM) (Pearl, 2009), a causal graph plus the structural equations that determine the crystal properties and the frequency (§3).

Two design choices distinguish CausaLab from prior causal-reasoning evaluations. First, the hidden SCM is sampled per episode rather than drawn from public causal corpora, which sidesteps the causal parrot concern that scores reflect memorized causal lexicon. Second, a lightweight domain-specific language (DSL; ğ4) records the agents accumulated evidence, current graph and equation hypothesis, planned experiment, and action at each step, so we can score, at every step, both *Target Mechanism Recovery* (the `frequency` prediction and its equation) and *Full System Discovery* (the whole causal graph) against the ground-truth SCM, since it has long been known that a correct prediction need not rest on a correct causal model (Pearl, 2009; Zečević et al., 2023).

We run closed and open-weight models at several sizes and diferent thinking effort. Four findings come out of it that a static benchmark cannot reach. **(1) A persistent gap lies between Target Mechanism Recovery and Full System Discovery.** Across different edge-function forms, different hidden pertubation settings, and different graph structures, we find that the two levels are not strongly positively correlated, and in some cases even trade off against each other. **(2) Intervention drives Full System Discovery better than observation.** With ten rounds of pure observation on 6-node graphs, GPT-5.2-high reaches 86% task accuracy but only 0.471 all-edge F1. The same ten rounds spent on interventions take that setting to 0.82 all-edge F_1 at 76% accuracy, and a mixed regime to 0.80 at 74%. Offline "Golden" intervention chains do the opposite: accuracy up, all-edge F_1 down, indicating that being handed good experiments is not the same as choosing them. **(3) Scale lifts both levels together, but does not close the gap.** Within a model family the larger model wins on both: `GPT-5.2-high` leads `GPT-5-mini` on accuracy and on directed structural Hamming distance (SHD) at every graph size, and still falls to 64% accuracy and SHD 4.640 at 7 nodes. Across families the ranking can disagree: `Qwen3.5-Thinking` predicts better than `GPT-5-mini` at every graph size and recovers structure worse. **(4) Most failures are early commitment, not exhausted budget.** Runs leave roughly half the intervention budget unspent, failed runs end holding a hypothesis their own data contradicts, and a per-step consistency-check prompt lifts 4-node accuracy from 48% to 60%. CausaLab therefore separates the two levels, revealing how current LLM agents still struggle to explore unfamiliar environments interactively, test candidate mechanisms, and revise toward the causal regularities that govern them. We release the full SCM dataset and benchmark code to advance future research.

2 Background and Related Work

Causal reasoning asks how a system would respond to an intervention, or what would have happened under a different one. That is a stronger question than associational prediction, and structural causal models (SCMs) are the standard way to make the assumptions behind it explicit: a directed graph over the variables, plus a structural equation for each (Pearl, 2009; Pearl & Mackenzie, 2018; Imbens & Rubin, 2015). In *CausaLab* each episode's hidden SCM plays both roles at once. It generates the data (§3.1) and it is what we score against (§3.3).

Classical causal discovery and intervention design. A rich literature learns causal structure from data: constraint-based PC (Spirtes et al., 2000), score-based greedy equivalence search (Chickering, 2002), functional-form LiNGAM/DirectLiNGAM (Shimizu et al., 2006; 2011), continuous-optimization NOTEARS (Zheng et al., 2018), and score matching (Rolland et al., 2022). A complementary line studies which experiments to run: intervention bounds (Eberhardt et al., 2005), active learning (Hauser & Bühlmann, 2012; 2014), budgeted design (Ghassami et al., 2018), and adaptivity/off-target complexity (Choo & Shiragur, 2023; Choo et al., 2024). *CausaLab* does not replace either line. It uses their target objects (graphs, equations, informative interventions) as ground truth for evaluating whether an LLM agent can design and interpret experiments through a language-mediated laboratory interface; §5.1 uses several of these algorithms as reference baselines, and §6 argues that the residual agent gap sits in the intervention-design half of this literature.

Static LLM causal evaluation. Most LLM causal evaluations are static. They ask a model to answer a textual causal question, reason over a graph it was given, classify cause–effect direction, or solve a formal causal-inference query (Kıcıman et al., 2023; Jin et al., 2023a;b; Chen et al., 2024b; Wang, 2024; Chen et al., 2024a). A second group uses the LLM as a causal prior rather than the object of study: for scoring edges, fixing an order, or making discovery query-efficient (Long et al., 2023; Darvariu et al., 2024; Vashishtha et al., 2023; Jiralerspong et al., 2024), for reasoning over variable metadata during discovery (Abdulaal et al., 2024), or for eliciting coefficients once the DAG is supplied (Yamaoka et al., 2026). HypoBench adds the reminder that hypothesis-generation benchmarks have to account for what the model already knows (Liu et al., 2025). These settings tell us what causal knowledge an LLM can express, but the model always works from inputs fixed in advance (a passage, a dataset, a supplied graph, or a formal query) and never decides which evidence to collect next.

Interactive LLM causal evaluation. Interactive benchmarks hand that decision to the agent. LLM agents recover a hidden causal graph through budgeted experiments or iterative queries (Havrilla et al., 2025; Chen et al., 2025), run full discovery cycles in simulated scientific worlds (Jansen et al., 2024), play causal games (Chen et al., 2026), or reverse-engineer black-box systems, where actively designing queries is not equivalent to receiving another agent's intervention data (Geng et al., 2025); DODO plans budgeted interventions without an LLM (Gregorini et al., 2025). The agents are typically scaffolded by ReAct-style reasoning and acting (Yao et al., 2023). Graph-discovery benchmarks score the recovered graph but not whether it transfers to a new instance, while discovery worlds and causal games score task outcomes rather than the mechanism itself. *CausaLab* scores both. The agent must predict a held-out reactor crystal generated by the same SCM as the manipulator crystal it experimented on, and its per-step DSL hypotheses expose the graph, the `frequency` structural equation, and the coefficients it commits to, so its SCM revisions can be audited over time. Appendix Table 2 compares these families on five capabilities (agent-chosen experimentation, per-episode hidden mechanisms, held-out transfer, direct mechanism scoring, and stepwise hypothesis scoring); *CausaLab* is the only benchmark combining all five.

3 The Construction of CausaLab

This section defines the episode-level task. It then specifies the SCM (§3.1), the interaction protocol (§3.2), and the evaluation targets (§3.3). Artifact and implementation details are in Appendix B.1. Figure 1 runs through the section as an example: the agent first observes prior crystal records, then intervenes on a controllable property of the manipulator crystal, and finally predicts the reactor crystal's hidden `frequency`.

Design principles. The benchmark is built around three questions. Can a model infer a causal mechanism that transfers to a new instance, rather than fitting an isolated value pattern? Can it choose informative interventions, rather than passively consume a fixed dataset? And how do both abilities scale with graph size, topology, functional form, intervention budget and hidden disturbances? The design choices below follow from those three.

Task formulation. An episode hides an SCM and asks the agent to transfer it. The graph, the structural equations and the coefficients are all hidden. What the agent gets is a batch of prior measurement records, each listing a crystal's properties and the `frequency` they produced, plus a budget for interventions (Figure 1).

Two crystals come out of that SCM. The agent spends its budget on the manipulator crystal: it sets one controllable property, reads the new measurement, and repeats. The reactor crystal is the test. Its `frequency` is withheld until the agent commits to a number.

The records, the manipulator and the reactor share the SCM but not their property values, so no observed frequency can simply be copied across. The agent has to work out which properties feed `frequency` and with what coefficients, then apply that to the reactor's own readings (Target Mechanism Recovery). How the remaining properties relate to each other is scored separately (Full System Discovery, §3.3).

Why a crystal laboratory? The crystal scenario comes from DiscoveryWorld's Reactor Lab (Jansen et al., 2024). It gives us inspectable properties, manipulable variables, prior records, and a held-out target, all over a hidden SCM whose ground truth the evaluator can execute. The laboratory framing is only the interface. All the causal content comes from the randomized SCM underneath, which is why an episode cannot be answered from domain knowledge. On top of that interface *CausaLab* adds the benchmark layer: randomized SCM suites, controlled mechanisms, several evidence regimes, two-level scoring, and trajectory logging.

The agent knows the property names and the functional family. It may intervene only on a configured subset $C \subseteq O$ of the controllable observable non-`frequency` variables. Everything outside C, including Y, can be read but not set, and the reactor crystal shows only its non-`frequency` variables (Appendix Table 3). At every step the agent also emits a DSL hypothesis, which we parse into a directed graph, a `frequency` equation and its coefficients. That is what lets us score how much of the hidden SCM the agent actually recovered, not just whether the reactor prediction was right (§3.3).

3.1 Structural Causal Models

Each episode instantiates an SCM $\mathcal{M} = (\mathbf{U}, \mathbf{V}, F, P(\mathbf{U}))$ (Pearl, 2009), with exogenous source terms $\mathbf{U} \sim P(\mathbf{U})$, endogenous variables $\mathbf{V}$, and structural equations F. In *CausaLab*, $\mathbf{V} = O \cup \{Y\}$: observable properties O plus the target $Y =$ `frequency`. We sample a DAG G over $\mathbf{V}$, assign root nodes from their exogenous sources, then compute non-root variables in topological order. We use exactly two structural-equation families: linear, $X = b + \sum_{p \in \mathrm{pa}(X)} w_p p$, and quadratic, $X = b + \sum_{p \in \mathrm{pa}(X)} (u_p p^2 + w_p p)$. Here $\mathrm{pa}(X)$ are the parents of X in G. For an edge $p \to X$, w_p and u_p are the linear and quadratic coefficients, and b is the intercept, or base value, of X. Root variables have no parents; their base values come from exogenous distributions.

One SCM is drawn per episode and reused: the graph, the equations and the coefficients, including `frequency`'s base value, are the same for the prior records, the manipulator and the reactor. Figure 1 shows that graph. Only the controllable properties' base values differ between the three instances, and that is the asymmetry the task rests on. The agent cannot read a frequency off one instance and reuse it; it has to recover the mechanism and evaluate it on the reactor's own values.

Some graph families add an unobserved exogenous disturbance H. After every intervention, H is resampled and added as a fixed-weight shift to a designated set of observable endogenous variables. Those shifted values then propagate downstream through F. The agent cannot observe or set H; it sees only the downstream effects. This tests whether an agent can tell a stable mechanism apart from post-intervention noise. Coefficient ranges and formal SCM and hidden-disturbance details appear in Appendix B.2.

3.2 Interaction and Outputs

Each episode proceeds through a repeated hypothesis–experiment loop. The agent receives an initial batch of measurement records, may intervene by setting one controllable non-`frequency` property on the manipulator crystal (the environment recomputes and returns the resulting measurement under the hidden SCM), and observes but never intervenes on the reactor crystal, whose `frequency` remains hidden until the agent submits a final value.

Concretely, the loop alternates *intervene on one controllable property* $\rightarrow$ *observe the resulting measurement* $\rightarrow$ *revise the DSL hypothesis and choose the next intervention*: e.g., set `radiation`, see how `temperature`, `conductivity`, and `frequency` change, and decide whether the evidence supports a direct edge into `frequency` or an indirect path (Figure 1; Appendix Figure 6).

The intervention semantics are shift-style rather than hard $\mathrm{do}(X{=}v)$ (Rothenhäusler et al., 2015), modeling a laboratory control that shifts a controllable baseline while preserving upstream dependencies across sequential interventions.

For a controllable variable $X \in C$, an intervention request with value v re-solves that variable's base term so that X reads exactly v on the next environment update. In the linear family this sets $b_X \leftarrow v - \sum_{p \in \mathrm{pa}(X)} w_p p$; the quadratic family works the same way with its extra terms retained. The edges into X survive, so a later change upstream still moves X. Only the intercept was reset. A hard intervention would instead pin $X = v$ and cut the incoming influence. Appendix B.2 states the prediction tolerance and the reactor's frequency ceiling.

The episode ends with two submissions: a number for the reactor `frequency`, and a final DSL hypothesis giving the causal edges, the proposed `frequency` equation and its coefficients. The agent may emit the same DSL at intermediate steps, which is what gives us a trajectory of hypotheses rather than one endpoint.

3.3 Evaluation

We score the agent's prediction and parsed hypothesis against the ground-truth SCM at two levels.

Target Mechanism Recovery. In Figure 1, `frequency` has two direct causes, `temperature` and `conductivity`. To predict the reactor's `frequency`, the agent must name these two parents, estimate how much each one moves `frequency`, and plug in the reactor's own readings. *Task success* checks the reactor number, `frequency`-edge F_1 the parent set, and `frequency`-weight F_1 the coefficients. Since the reactor shows every property except `frequency`, nothing else is needed.

Full System Discovery. The same graph also has edges that never touch `frequency`: `radiation` drives `quantumSize` and `conductivity`, and both of those drive `temperature`. The agent must report these edges too, but the reactor prediction never uses them. All-edge precision, recall and F_1 score every edge of the agent's graph; structural Hamming distance (SHD) counts missing, extra and reversed edges, so lower is better; and root F_1 checks whether the agent finds where the causal chains into `frequency` start, the upstream variables with no parents (here `radiation`).

The two levels diverge both ways: an agent can predict the reactor correctly with most other edges wrong, or recover most of the graph and miss the `frequency` coefficients (§5.2).

4 A DSL for Causal Trajectories

Final-answer accuracy cannot distinguish guessing from transferable mechanism discovery, so we introduce a domain-specific language (DSL) that records per-step causal commitments as scoreable SCM artifacts.

At each interaction step t, the agent emits a compact DSL record with five fields: *Memory* M_t, the persistent episode notes; *Thought* T_t, a short interpretation of the current evidence; *Past data* $\mathcal{D}_{\leq t}$, the accumulated observations and intervention outcomes; *Hypothesis* H_t, the current causal claim; and *Experiment* E_t, the next planned intervention and its rationale. Only H_t is scored: a fixed schema (rather than free-form prose) states directed edges as `(parent, child)` pairs, a `frequency` structural

equation in the declared functional family, and numeric coefficients. A deterministic parser converts each valid hypothesis into a candidate graph G_t and target mechanism $\hat{f}_t$, producing a trajectory $\{(G_t, \hat{f}_t)\}_{t=1}^{T}$ scored with the same graph, root, and coefficient metrics used for final evaluation. Appendix Figure 6 shows parsed hypotheses rendered as candidate graphs and recovery metrics over time; prompting details appear in Appendix C.1.

5 EXPERIMENTS

We use *CausaLab* to ask four questions. (RQ1) Does Target Mechanism Recovery imply Full System Discovery? (RQ2) Which interaction regime best balances the two, and can offline intervention traces replace online experimental choice? (RQ3) How do model family, scale, and thinking traces affect both levels across graph sizes? (RQ4) Why do agents fail, and what simple check reduces these failures?

5.1 EXPERIMENTAL SETUP

Setup. The main suite covers four models on *CausaLab*'s 3–7 node graph families: `GPT-5-mini`, `GPT-5.2-high`, `Qwen3.5-Thinking` and `Qwen3.5-Non-thinking`. Each (graph size, model) cell uses up to 50 topologies, one run per task. RQ3 adds `DeepSeek-V4-Flash/Pro` on the 6-node suite, bringing the comparison to six configurations across three families. Observation–intervention scaling experiments use the two GPT models on the 4-/6-node suites; targeted follow-ups use the 4-/6-node suites, primarily with `GPT-5-mini`. Runs use fixed evidence budgets per graph size and, except DeepSeek, temperature 0.1 (Appendix D.1). The reactor's hidden `frequency` is the target in every episode, so end-task accuracy is just the rate at which that value is predicted correctly. Target Mechanism Recovery and Full System Discovery are scored separately against the episode SCM (§3.3). Except for the scaling and Golden suites, all follow-up analyses use the mixed regime with two initial observations (motivated in RQ2).

Evidence-acquisition regimes. Our analyses compare four regimes: pure observation, pure intervention, observation-conditioned intervention, and the Golden offline trace. They differ in the *initial* evidence (passive records, none, or passive plus a fixed intervention trace) and in whether the agent chooses new interventions *online* (Appendix Table 5). Every intervention returns an observed state, so pure versus observation-conditioned intervention differ only in initial evidence; pure observation and Golden differ in whether adjacent records are linked by a known intervention.

Reference baselines. Before reading anything into the agents' scores, we need to know the environment is solvable. We run four classical causal-discovery baselines on exactly the passive records the observation-only agents saw: PC (Spirtes et al., 2000), a greedy equivalence search variant (Chickering, 2002), DirectLiNGAM (Shimizu et al., 2011) and NOTEARS (Zheng et al., 2018). We add a *paired-shift* control that shifts one controllable variable at a time (Appendix D.4).

Paired shift exploits the suites' linearity and recovers every one of the 250 main-suite graphs exactly, Directed $F_1 = 1.000$, from $k - 1$ transitions. That is a quarter of the agents' $4(k - 1)$ budget. The environment therefore carries enough interventional signal for exact recovery. Whatever the agents are missing (on 6-node graphs `GPT-5.2-high` reaches 0.837 from 20 interventions and 0.799 from 5 observations plus 20) is experiment design and language-mediated inference, not an underdetermined environment (§6).

On passive records alone, `GPT-5.2-high` holds its own against DirectLiNGAM and PC at 4 nodes but not at 6, and trails NOTEARS and the greedy equivalence search variant, DGES (Table 1).

5.2 RQ1: ENDPOINT SUCCESS DOES NOT CERTIFY FULL SYSTEM DISCOVERY

CausaLab pairs each episode with a ground-truth SCM, so we can score Target Mechanism Recovery and Full System Discovery separately (§3.3). Based on this, we designed three controlled experiments to explore the relationship between endpoint success and Full System Discovery under different edge function and different graph structures.

	Recovered from the observation-only agents' passive records					$k-1$ shifts
Suite	`GPT-5.2-high`‡	DirectLiNGAM	NOTEARS	DGES†	PC†*	Paired shift
4-node	0.648	0.628	0.883	0.832	0.638	**1.000**
6-node	0.471	0.523	0.780	0.702	0.518	**1.000**

Table 1: Directed F_1 against the episode SCM. † cannot fully orient edges, so that column reports skeleton F_1. ‡ the agent reverses no edge, so its skeleton and directed scores coincide. * averaged over the graphs where PC succeeds, 35/50 at 4 nodes and 41/50 at 6; every other column averages all 50. Paired shift shifts one controllable variable at a time and recovers every graph exactly from $k-1$ transitions, a quarter of the agent's own budget. The agent gap is therefore not an artifact of an underdetermined environment. Full per-metric version in Appendix Table 8.

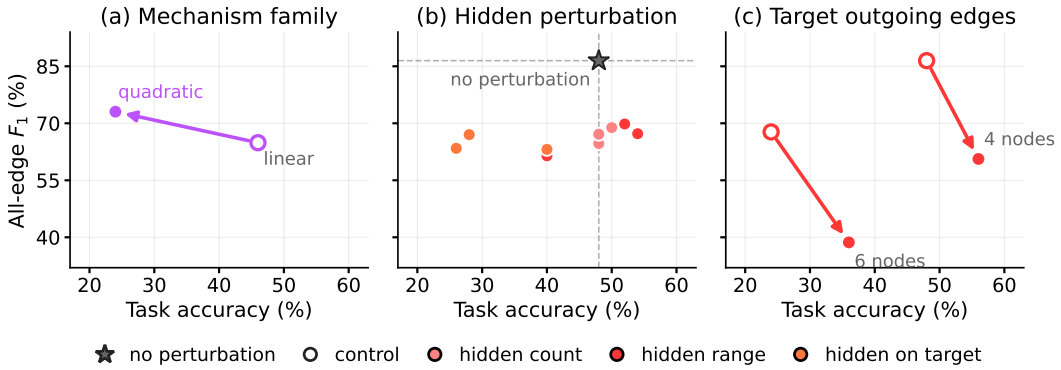


Figure 2: Three matched controls in one (task accuracy, all-edge F_1) frame. Open circles and the star are controls, filled circles the matched perturbation. (a) Replacing the linear mechanism family with a hard-quadratic one on fixed 4-node topologies costs 22 points of accuracy while all-edge F_1 *rises* 8 points. (b) On 4-node graphs, hidden disturbances off the target move accuracy little and drop F_1; disturbances that can perturb `frequency` itself cut accuracy without dropping F_1 further. (c) Letting `frequency` have outgoing edges raises accuracy on 4- and 6-node graphs while all-edge F_1 falls. No panel moves along a single diagonal, which is what it would do if one score predicted the other. `GPT-5-mini` throughout, 2 initial observations and $4(k-1)$ interventions (6-node `FreqParent`: 12); per-setting numbers in Appendix Tables 9, 10, and 11.

Function form. Holding the 50 four-node topologies fixed but replacing the linear mechanism with a hard-quadratic one cuts `GPT-5-mini` accuracy from 46% to 24% (Figure 2a). Full System Discovery survives that drop: root-node F_1 rises from 0.739 to 0.829, all-edge precision holds at 0.887 and 0.892, and all-edge F_1 rises from 0.649 to 0.731 (Appendix Table 9). What the agent loses is Target Mechanism Recovery: the coefficients of the `frequency` equation. It can find plausible parents and still fail.

Hidden perturbations. Off-target hidden noise leaves accuracy near baseline (40–54% versus 48%) but all-edge F_1 falls to 0.61–0.70, partly a batch effect. Once the disturbance can perturb `frequency` itself, accuracy drops to 26–40% (Figure 2b, Appendix Table 10). Noise off the target erodes Full System Discovery; noise on it breaks the endpoint.

Target outgoing edges. `FreqParent` keeps mean edge counts matched but lets `frequency` have outgoing edges: accuracy rises on 4- and 6-node graphs (fewer incoming target edges to fit) while all-edge recovery falls (global directionality is harder; Figure 2c, Appendix Table 11).

Takeaway: Endpoint accuracy is not evidence of Full System Discovery; *CausaLab* therefore scores Target Mechanism Recovery and Full System Discovery separately.

5.3 RQ2: Intervention, Not Observation, Drives Full System Discovery

RQ2 separates two questions: whether agents need observations, interventions, or both; and whether offline intervention data is enough when the agent does not choose the experiments online. Figure 3 compares the three regimes at a matched evidence budget across our four scaling families. Pure observation has the lowest all-edge F_1 everywhere; spending the same budget on interventions gains 6–35 points of F_1. In the `GPT-5.2-high` 6-node setting at budget ten, observation-only has the highest accuracy (86% against 76% and 74%) and the lowest F_1 (0.47 against 0.82 and 0.80). The accuracy cost is not universal: at `GPT-5-mini` 6 nodes, where every regime sits near the floor, the mixed regime leads on both axes. The two interventional regimes stay within 0.05 F_1 of each other, so we anchor the follow-up controls on the mixed regime, the better setting for the weaker model. Budget sweeps, curves and tables are in Appendix Figures 10, 11 and 12 and Appendix F.1.

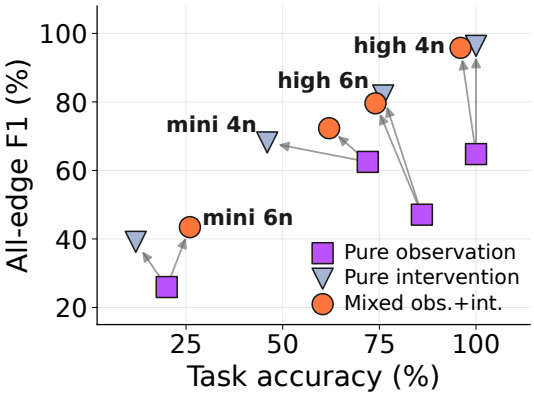


Figure 3: Three regimes at a matched evidence budget: six rounds at 4 nodes, ten at 6, split evenly by the mixed regime. Arrows run from pure observation to each interventional regime.

The *Golden* control asks whether the agent needs to choose its own experiments. It replaces the online loop with a fixed intervention chain, chosen to minimize the size of the interventional Markov equivalence class (MEC) that the evidence leaves open. Golden lifts task accuracy well above the main-suite baselines, 90% against 48% at 4 nodes and 44% against 24% at 6. All-edge F_1 drops at both sizes (Appendix Figure 13, right, and Table 16). A good intervention chain the agent did not pick behaves like stronger observation: it helps fit the target equation but does little for the graph.

Takeaway: Interventions, not observations, drive Full System Discovery, often at some cost in accuracy; a Golden intervention chain the agent did not choose raises accuracy but lowers all-edge F_1.

5.4 RQ3: Scale Lifts Both Axes, but Cross-Family Rankings Disagree

Within a family, the larger model wins on both axes. `GPT-5.2-high` has the best endpoint accuracy and the lowest directed all-edge SHD of the four models at every graph size (Appendix Figure 14), and its gain over `GPT-5-mini` concentrates in accuracy and `frequency-weight` F_1 (Appendix Figure 15). On the 6-node suite, `DeepSeek-V4-Pro` beats `DeepSeek-V4-Flash` on task success by 0.200 (95% CI $[0.060, 0.340]$), on `frequency-weight` F_1 by 0.169 ($[0.064, 0.282]$), and on SHD (1.900 against 3.520).

Across families the ranking disagrees. `Qwen3.5-Thinking` beats `GPT-5-mini` on accuracy at every graph size but trails it on SHD. Thinking lowers Qwen's SHD on 3–6 nodes, but at 7 nodes the non-thinking variant edges ahead on SHD (8.000 against 8.120) and all-edge F_1 (0.495 against 0.454). Scale does not close the gap: `GPT-5.2-high` still falls to 64% accuracy, SHD 4.640, and all-edge F_1 0.745 at 7 nodes (Appendix G). We scope these conclusions to the six configurations tested.

Takeaway: Within a family, scale lifts both levels together, yet `GPT-5.2-high` still falls to 64% accuracy at 7 nodes; across families, different models lead on different levels.

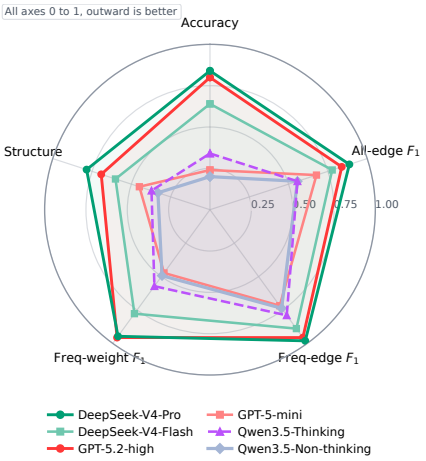
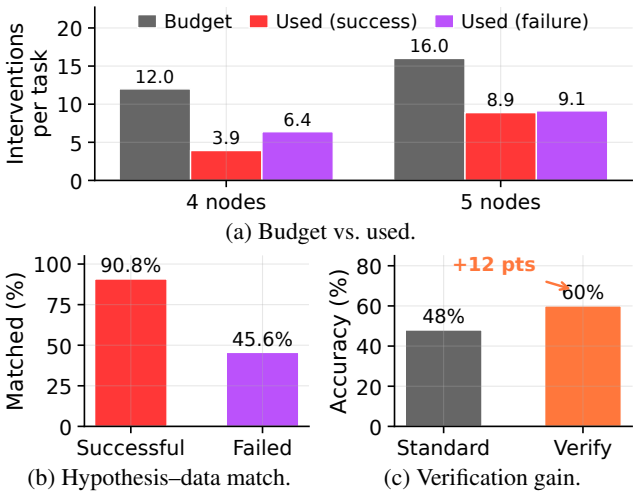


Figure 4: All six configurations on the same 50 6-node graphs (2 initial observations, 20 interventions).

5.5 RQ4: Agents Stop Early, and a Consistency Check Recovers Part of the Loss

Agents fail by committing early, not by running out of data. Across the main 4- and 5-node experiments, successful and failed runs alike leave 43–67% of the intervention budget unused. At least 8 of 50 four-node runs and 14 of 50 five-node runs stop before they have intervened once on every controllable variable. More budget does not rescue them. Quadrupling it moves `GPT-5.2-high`'s 6-node mixed-regime all-edge F_1 by $+0.003$, and it even *lowers* `GPT-5-mini`'s 4-node pure-intervention all-edge F_1 from 0.754 to 0.596.



(a) Budget vs. used.

(b) Hypothesis–data match.

(c) Verification gain.

Figure 5: Early-commitment diagnostics for `GPT-5-mini` on 4- and 5-node graphs: both outcome groups leave 43–67% of the budget unused; failed hypotheses fit their data worse; a consistency check lifts 4-node accuracy.

The DSL lets us inspect *why* agents stop: at the step where each run commits to a final answer, we compare the recorded hypothesis H_t against the data already collected. Successful runs mostly end with hypotheses consistent with their own $\mathcal{D}_{\leq t}$, whereas failed runs end with hypotheses that mispredict the data that produced them (Figure 5b). The failure mode is *overconfidence*: agents treat an unverified hypothesis as a final theory instead of spending budget on disambiguating experiments. Prompting the agent to check each H_t against $\mathcal{D}_{\leq t}$ raises 4-node accuracy from 48% to 60% (Figure 5c).

Takeaway: Agents stop with budget to spare, and failed runs commit to SCMs their own data contradicts; a per-step consistency-check prompt lifts 4-node accuracy from 48% to 60%.

6 Discussion and Conclusion

We introduced *CausaLab*, which hides a freshly sampled SCM in every episode and logs the agent's hypotheses in a DSL, so both levels can be scored at each step. They come apart: agents often predict `frequency` from an incomplete graph, interventions drive structure recovery, scale lifts both levels without closing the gap, and most failures are early commitment. **Endpoint accuracy can mislead.** Observation-only evidence, a Golden chain and outgoing edges from `frequency` each raised accuracy in some setting while lowering all-edge F_1. Grading only the answer would rank these settings backwards, so agent evaluations should also score the model. **The missing skill is experiment design.** A non-adaptive paired-shift control recovers all 250 main-suite graphs exactly on a quarter of the agents' budget. It is handed the suites' linearity, so it only bounds the recoverable signal, yet `GPT-5.2-high` reaches 0.837 all-edge F_1 at 6 nodes with the full intervention budget. The gap lies in choosing experiments, the adaptive-design problem studied for classical discovery (Eberhardt et al., 2005; Hauser & Bühlmann, 2012; Ghassami et al., 2018; Choo & Shiragur, 2023), here posed for language-mediated agents. **Verification is a cheap lever.** Runs stop with half their budget unspent and failed runs commit to hypotheses their own data contradicts, yet a consistency check lifts 4-node accuracy from 48% to 60%, which makes self-verification a concrete target for agent design. *CausaLab* uses 3–7 node SCMs, mostly linear mechanisms and shift rather than hard-*do* interventions. Its sampler makes harder suites (denser graphs, richer functions, latent confounders, noisy measurements) a configuration change, and the 6- and 7-node suites are not yet saturated, so *CausaLab* can keep testing whether agents succeed for the right mechanistic reason.

Ethics Statement

All authors have read and adhere to the ICLR Code of Ethics. *CausaLab* is a fully synthetic benchmark: every episode is generated programmatically from a randomly sampled structural causal

model over fictional laboratory variables. The work involves no human subjects, annotators, or crowdworkers, and the released data contain no personal information, demographic attributes, or naturally occurring web text, so the dataset itself raises no privacy, consent, or fairness concerns. The environment builds on DiscoveryWorld (Jansen et al., 2024), released under Apache-2.0, and our code and SCM dataset are released under the same license (Appendix B.1). The evaluated LLMs are used only as test subjects through inference endpoints; no model is trained or fine-tuned. The main risk is overgeneralization. Results on templated SCM tasks in a simulator should not be read as evidence that an agent is ready to run experiments in real scientific, medical, or policy settings, where interventions carry physical, safety, or ethical costs and domain constraints apply. We see no further dual-use risk beyond that of agent evaluation in general.

Reproducibility statement

Environment mechanics, SCM sampling, and evaluation metrics are specified in §3, §4, and Appendices B.3, B.2, and C.1; the prompt files are reproduced in Appendix C.3, and models, decoding temperature, budgets, and evidence regimes are given in §5.1 and Appendix D. The anonymized supplementary material contains the complete SCM dataset (19 graph suites of 50 graphs each, with Croissant metadata and checksums), the graph-sampling scripts, the environment, agent, and evaluation code with the model configurations and prompt files, the classical and paired-shift baseline implementations, and the DSL parsing and visualization tooling. The paired-shift baseline runs offline, without model access, and reproduces its reported output exactly. Re-running the agents requires access to the evaluated models; because each cell uses one sampled run per task and hosted models can change, a re-run reproduces the protocol rather than identical trajectories. We will release all of these materials publicly, together with the per-episode results and the full episode trajectories.

AI use statement

We used AI tools for three purposes, matching the disclosure on the submission form. For writing, we used them to aid and polish the manuscript text and to format LaTeX and references. For retrieval and discovery, we used them to help find related work. For research execution, we used them to assist in implementing code for the environment, experiments, evaluation, result aggregation, and figures, and in preparing the supplementary material. We did not use AI tools to formulate the research questions, design the benchmark or the experiments, or generate the benchmark data, which are sampled programmatically. Beyond these uses, LLMs appear in this paper only as the systems under evaluation. The authors reviewed all AI-assisted text and code and take full responsibility for the content of this paper.

References

Ahmed Abdulaal, adamos hadjivasiliou, Nina Montana-Brown, Tiantian He, Ayodeji Ijishakin, Ivana Drobnjak, Daniel C. Castro, and Daniel C. Alexander. Causal modelling agents: Causal graph discovery through synergising metadata- and data-driven reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=pAoqRlTBtY.

Steen A. Andersson, David Madigan, and Michael D. Perlman. A characterization of Markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505 – 541, 1997. doi: 10.1214/aos/1031833662. URL https://doi.org/10.1214/aos/1031833662.

Sirui Chen, Bo Peng, Meiqi Chen, Ruiqi Wang, Mengying Xu, Xingyu Zeng, Rui Zhao, Shengjie Zhao, Yu Qiao, and Chaochao Lu. Causal evaluation of language models, 2024a.

Sirui Chen, Mengying Xu, Kun Wang, Xingyu Zeng, Rui Zhao, Shengjie Zhao, and Chaochao Lu. Clear: Can language models really understand causal graphs?, 2024b. URL https://arxiv.org/abs/2406.16605.

Tingting Chen, Srinivas Anumasa, Beibei Lin, Vedant Shah, Anirudh Goyal, and Dianbo Liu. Auto-Bench: An automated benchmark for scientific discovery in LLMs, 2025. URL https://arxiv.org/abs/2502.15224.

Zhenhao Chen, Yongqiang Chen, Chenxi Liu, Junchi Yu, Xiangchen Song, Zijian Li, Jialin Li, Philip Torr, Bo Han, and Kun Zhang. Causalgame: Benchmarking causal thinking of llm agents in games. In *ICLR 2026 Workshop on Foundation Models for Science*, 2026. URL https://openreview.net/forum?id=SEFSkn4l6d.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

Davin Choo and Kirankumar Shiragur. Adaptivity complexity for causal graph discovery. In *Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence*, 2023.

Davin Choo, Kirankumar Shiragur, and Caroline Uhler. Causal discovery under off-target interventions. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, 2024.

Victor-Alexandru Darvariu, Stephen Hailes, and Mirco Musolesi. Large language models are effective priors for causal graph discovery, 2024. URL https://arxiv.org/abs/2405.13551.

K. Dunbar and J. Fugelsang. Causal thinking in science: How scientists and students interpret the unexpected. In M. E. Gorman, R. D. Tweney, D. C. Gooding, and A. P. Kincannon (eds.), *Scientific and Technological Thinking*, pp. 57–79. Lawrence Erlbaum Associates, Mahwah, NJ, 2005.

Frederick Eberhardt, Clark Glymour, and Richard Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among N variables. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

Jiayi Geng, Howard Chen, Dilip Arumugam, and Thomas L. Griffiths. Are large language models reliable AI scientists? assessing reverse-engineering of black-box systems, 2025. URL https://arxiv.org/abs/2505.17968.

AmirEmad Ghassami, Saber Salehkaleybar, Negar Kiyavash, and Elias Bareinboim. Budgeted experiment design for causal structure learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Matteo Gregorini, Chiara Boldrini, and Lorenzo Valerio. DODO: Causal structure learning with budgeted interventions, 2025. URL https://arxiv.org/abs/2510.08207.

Alain Hauser and Peter Bühlmann. Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs. *Journal of Machine Learning Research*, 13(79): 2409–2464, 2012. URL http://jmlr.org/papers/v13/hauser12a.html.

Alain Hauser and Peter Bühlmann. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55(4):926–939, 2014.

Alex Havrilla, David Alvarez-Melis, and Nicolo Fusi. Igda: Interactive graph discovery through large language model agents, 2025. URL https://arxiv.org/abs/2502.17189.

Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences*. Cambridge University Press, 2015. ISBN 978-0521885884. doi: 10.1017/CBO9781139025751. URL https://doi.org/10.1017/CBO9781139025751.

Peter A. Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. DISCOVERYWORLD: A virtual environment for developing and evaluating automated scientific discovery agents. *CoRR*, abs/2406.06769, 2024. doi: 10.48550/ARXIV.2406.06769. URL https://doi.org/10.48550/arXiv.2406.06769.

Liwei Jiang, Taylor Sorensen, Sydney Levine, and Yejin Choi. Can language models reason about individualistic human values and preferences? *arXiv:2410.03868*, 2024.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng LYU, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. Cladder: Assessing causal reasoning in language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Proceedings of the Advances in Neural Information Processing Systems*, 2023a. doi: 10.48550/ARXIV.2312.04350. URL https://doi.org/10.48550/arXiv.2312.04350.

Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation?, 2023b. URL https://arxiv.org/abs/2306.05836.

Thomas Jiralerspong, Xiaoyin Chen, Yash More, Vedant Shah, and Yoshua Bengio. Efficient causal graph discovery using large language models, 2024. URL https://arxiv.org/abs/2402.01207.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2023. URL https://arxiv.org/abs/2305.00050.

Andrew K. Lampinen, Stephanie C. Y. Chan, Ishita Dasgupta, Andrew J. Nam, and Jane X. Wang. Passive learning of active causal strategies in agents and language models. In *Proceedings of the Advances in Neural Information Processing Systems*, 2023.

Pat Langley. Scientific discovery, causal explanation, and process model induction. *Mind & Society*, 18(1):43–56, 2019. doi: 10.1007/s11299-019-00216-1. URL https://doi.org/10.1007/s11299-019-00216-1.

Haokun Liu, Sicong Huang, Jingyu Hu, Yangqiaoyu Zhou, and Chenhao Tan. Hypobench: Towards systematic and principled benchmarking for hypothesis generation. *arXiv:2504.11524*, 2025.

Jintao Liu, Zequn Zhang, Zhi Guo, Li Jin, Xiaoyu Li, Kaiwen Wei, and Xian Sun. Kept: Knowledge enhanced prompt tuning for event causality identification. *KnowledgeBased Systems*, 259, 2023.

Stephanie Long, Tibor Schuster, and Alexandre Piché. Can large language models build causal graphs?, 2023. URL https://arxiv.org/abs/2303.05279.

Joris M. Mooij, Sara Magliacane, and Tom Claassen. Joint causal inference from multiple contexts. *Journal of Machine Learning Research*, 21(99):1–108, 2020. URL http://jmlr.org/papers/v21/17-123.html.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, sep 2009. ISBN 9780511803161. doi: 10.1017/cbo9780511803161. URL https://doi.org/10.1017/CBO9780511803161.

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018. ISBN 046509760X.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. Counterfactual story reasoning and generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5043–5053, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1509. URL https://aclanthology.org/D19-1509/.

Paul Rolland, Volkan Cevher, Matthäus Kleindessner, Chris Russell, Dominik Janzing, Bernhard Schölkopf, and Francesco Locatello. Score matching enables causal discovery of nonlinear additive noise models. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Angelika Romanou, Syrielle Montariol, Debjit Paul, Léo Laugier, Karl Aberer, and Antoine Bosselut. Crab: Assessing the strength of causal relationships between realworld events. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, pp. 15198–15216, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.940. URL https://aclanthology.org/2023.emnlp-main.940.pdf.

Dominik Rothenhäusler, Christina Heinze, Jonas Peters, and Nicolai Meinshausen. backshift: Learning causal cyclic graphs from unknown shift interventions, 2015. URL https://arxiv.org/abs/1506.02494.

Shohei Shimizu, Patrik O. Hoyer, Aapo Hyvärinen, and Antti Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O. Hoyer, and Kenneth Bollen. DirectLiNGAM: A direct method for learning a linear non-Gaussian structural equation model. *Journal of Machine Learning Research*, 12:1225–1248, 2011.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. MIT Press, 2nd edition, 2000.

Alessandro Stolfo, Zhijing Jin, Kumar Shridhar, Bernhard Schölkopf, and Mrinmaya Sachan. A causal framework to quantify the robustness of mathematical reasoning with language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2023. URL https://arxiv.org/abs/2210.12023.

Aniket Vashishtha, Abbavaram Gowtham Reddy, Abhinav Kumar, Saketh Bachu, Vineeth N Balasubramanian, and Amit Sharma. Causal order: The key to leveraging imperfect experts in causal inference, 2023. URL https://arxiv.org/abs/2310.15117.

Aniket Vashishtha, Abhinav Kumar, Atharva Pandey, Abbavaram Gowtham Reddy, Kabir Ahuja, Vineeth N Balasubramanian, and Amit Sharma. Teaching transformers causal reasoning through axiomatic training. In *Proceedings of the International Conference on Machine Learning*, 2025.

Zeyu Wang. CausalBench: A comprehensive benchmark for evaluating causal reasoning capabilities of large language models. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pp. 143–151, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.sighan-1.17/.

Kanta Yamaoka, Sumantrak Mukherjee, Thomas Gärtner, David Antony Selby, Stefan Konigorski, Eyke Hüllermeier, Viktor Bengs, and Sebastian Josef Vollmer. Linear-LLM-SCM: Benchmarking LLMs for coefficient elicitation in linear-gaussian causal models, 2026. URL https://arxiv.org/abs/2602.10282.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Matej Zečević, Moritz Willig, Devendra Singh Dhami, and Kristian Kersting. Causal parrots: Large language models may talk causality but are not causal. *Transactions in Machine Learning Research*, 2023. doi: 10.48550/ARXIV.2308.13067. URL https://arxiv.org/abs/2308.13067.

Junhao Zheng, Qianli Ma, Shengjie Qiu, Yue Wu, Peitian Ma, Junlong Liu, Huawen Feng, Xichen Shang, and Haibin Chen. Preserving commonsense knowledge from pre-trained language models via causal inference. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9155–9173, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.509. URL https://aclanthology.org/2023.acl-long.509/.

Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. DAGs with NO TEARS: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems 31*, 2018.

APPENDIX OVERVIEW

The appendix follows the order of the main text. Appendix A expands the related-work comparison; Appendix B gives the environment, SCM, and graph-sampling details behind §3; Appendix C gives the DSL implementation, the trajectory visualization, and the prompt templates behind §4; Appendix D gives the evidence budgets, the evidence-acquisition regimes and their identifiability, and the classical baselines behind §5.1; and Appendices E, F, G, and H give the full results behind RQ1–RQ4 (§5.2–§5.5).

A RELATED-WORK DETAILS

This appendix expands the Background and Related Work section: it places *CausaLab* relative to classical causal discovery and compares its capabilities with the benchmark families discussed there.

A.1 CLASSICAL CAUSAL-DISCOVERY CONTEXT

Classical causal discovery studies how causal structure can be learned from observational, interventional, or shifted data, including constraint- and score-based discovery, interventional Markov equivalence, discovery across multiple contexts, and unknown shift interventions (Pearl, 2009; Spirtes et al., 2000; Chickering, 2002; Andersson et al., 1997; Hauser & Bühlmann, 2012; Mooij et al., 2020; Rothenhäusler et al., 2015). *CausaLab* is closest to the shift-intervention regime of this literature, but it does not propose a new discovery algorithm or assume perfect do-interventions. Instead, it evaluates whether an LLM agent can recover the graph and the `frequency` equation of a hidden SCM from a finite budget of observations and shift-style interventions on controllable properties.

A.2 BENCHMARK CAPABILITY COMPARISON

Table 2 gives the per-capability comparison referred to in the related-work section.

Work type	Agent exp.	Ep. mech.	Transfer	Mech. score	Step score
Static LLM causal evaluation (Kıcıman et al., 2023; Jin et al., 2023a; Chen et al., 2024b; Wang, 2024; Jin et al., 2023b; Chen et al., 2024a)	—	—	—	✓	—
LLM coefficient elicitation (Yamaoka et al., 2026)	—	—	—	✓	—
Interactive graph discovery (Havrilla et al., 2025; Chen et al., 2025)	✓	✓	—	✓	✓
Simulated discovery worlds (Jansen et al., 2024)	✓	✓	✓	○	—
Causal games (Chen et al., 2026)	✓	—	✓	○	—
Black-box reverse engineering (Geng et al., 2025)	✓	—	○	✓	—
CausaLab	✓	✓	✓	✓	✓

Table 2: Capability comparison with related benchmark families: agent-chosen experiments determine the next evidence (*Agent exp.*), each episode has a generated hidden mechanism (*Ep. mech.*), learned mechanisms must transfer to a held-out task (*Transfer*), recovery of the graph/equation/parameters is directly scored (*Mech. score*), and intermediate structured hypotheses are scored along the trajectory (*Step score*). A ✓ means the capability appears in at least one task of at least one benchmark in the family. A ○ marks partial support: the mechanism is graded only as a free-text explanation against rubric questions, or held-out prediction appears only as an auxiliary analysis.

B Environment and SCM Details

This appendix supports §3: the artifact and its implementation, the formal SCM with each variable's access and the endpoint tolerance, and the graph-sampling procedure with the resulting main-suite statistics.

B.1 Artifact and Implementation Details

CausaLab builds its embodied interface on DiscoveryWorld (Jansen et al., 2024), released under the Apache-2.0 license; its SCM generator, intervention/reactor mechanics, DSL traces, and scoring code are new synthetic research and evaluation artifacts. This use is consistent with DiscoveryWorld's role as a virtual environment for evaluating scientific-discovery agents. The benchmark is generated from fixed templates over synthetic laboratory variables, so it contains no demographic attributes, personal identifiers, or naturally occurring offensive web text. Appendices B.2 and B.3 document artifact coverage and variable access, and Appendix D documents the evaluation and experiment settings.

The supplementary material contains the complete SCM dataset (19 graph suites of 50 graphs each, with Croissant metadata and checksums), the graph-sampling scripts, the environment, agent, and evaluation code with the model configurations and prompt files, the classical and paired-shift baseline implementations, and the DSL parsing and visualization tooling. We will release all of it publicly, together with the per-episode results and full episode trajectories.

B.2 SCM Formalism, Variable Access, and Endpoint Tolerance

Formally, each episode instantiates an SCM $\mathcal{M} = (\mathbf{U}, \mathbf{V}, F, P(\mathbf{U}))$ (Pearl, 2009): $\mathbf{V}$ are endogenous variables determined inside the system, $\mathbf{U}$ are exogenous variables drawn from $P(\mathbf{U})$, and F is a collection of structural equations mapping each variable's parents and exogenous term to its value. In *CausaLab*, $\mathbf{V} = O \cup \{Y\}$, where O are observable property variables and $Y =$ `frequency`. Root variables remain endogenous nodes, but their values are generated from exogenous source terms; in hidden-disturbance suites (called hidden-noise settings in Appendix E), the exogenous terms also include an unobserved disturbance H.

In the linear suites every edge coefficient w_p is an integer drawn uniformly from $\{1, 2, 3\}$, and `frequency`'s base value is an integer in $[15, 25]$, fixed per graph. The base value of every controllable variable is drawn separately for each passive prior record, the manipulator and the reactor, as a uniform integer in $[0, 100]$. In the hard-quadratic suite the quadratic coefficients u_p lie in $\{0.008, 0.012, 0.016\}$, the linear coefficients w_p in $[-2.05, -0.56]$ and `frequency`'s base value in $[52.7, 190.7]$; the controllable base values of the records and crystals are preset per graph rather than drawn at run time.

The agent observes O and Y in the prior evidence records and on the manipulator crystal, may intervene on the configured subset $C \subseteq O$ of the manipulator crystal, and observes only O on the reactor crystal. The reactor display can also list default crystal attributes that are not variables of the episode's SCM (e.g. `temperatureC` at 25); they are constant and do not enter `frequency`. Variables in $O \setminus C$ are observable but not controllable; Y is never controllable; and H, when present, is neither observable nor controllable. Hidden-disturbance suites resample H uniformly from $\{-v, 0, v\}$ after each intervention, with $v \in \{0.5, 5, 50\}$ ($v = 5$ in the count and `freqnode` settings), and add it with weight 1 to 1–3 designated observable endogenous variables, which include `frequency` in the `freqnode` settings; these shifted values then propagate downstream through the structural equations. The agent sees only the resulting observed values, not H itself. Table 3 summarizes each variable's SCM role and access.

Three environment settings govern the endpoint task. An episode succeeds when the agent activates the reactor, which requires setting the reactor's dial strictly within 5 Hz of the true frequency. The agent may set the dial up to three times and sees after each setting whether the reactor activated, so end-task accuracy throughout the paper is a ± 5 Hz hit rate over at most three feedback-guided attempts rather than an exact-match or single-shot rate. Separately, the dial is clamped to $[0, 10^4]$ Hz; an episode whose true frequency lies 5 Hz or more above that ceiling therefore cannot be answered correctly regardless of the agent's reasoning. In the GPT runs, true frequencies above the ceiling occur

Variable role	SCM role	Observable	Intervenable
Controllable property ($X \in C$)	Endogenous	✓	✓
Non-controllable property ($X \in O \setminus C$)	Endogenous	✓	—
Target $Y =$ `frequency` (evidence records / manipulator crystal)	Endogenous	✓	—
Target $Y =$ `frequency` (reactor crystal)	Endogenous	—	—
Hidden disturbance H (when present)	Exogenous	—	—

Table 3: Per-variable SCM role and access for the agent in a *CausaLab* episode. The agent observes prior evidence records, intervenes only on the configured controllable subset C of the manipulator crystal, never observes hidden exogenous disturbances, and predicts the held-out `frequency` of the reactor crystal. In every reported suite $C = O$, so the non-controllable-property row is supported by the configuration format but empty in all experiments.

Nodes	Graphs	Edge mean	Edge var.	Fork mean	Collider mean
3	50	2.56	0.25	0.68	0.72
4	50	4.54	1.01	2.18	2.22
5	50	7.26	4.23	5.54	5.26
6	50	8.82	5.15	6.72	6.22
7	50	10.26	4.95	7.84	7.00

Table 4: Main-experiment graph statistics. Forks and colliders are counted as three-node motif instances; edge variance is the population variance over the 50 graphs in each row.

in no 3–4-node episodes, in about 3% of 5-node episodes, and in 7–10% of 6–7-node episodes (2–15% across individual 6-node suites). We report accuracy over all sampled graphs without excluding them.

B.3 Graph Sampling Procedure and Main-Suite Statistics

Each episode's DAG is sampled by the following procedure. The node set is {`resonanceFreq`} plus $k-1$ sampled properties; the edge count is either fixed to m or sampled from $[\,k-1,\ \min(k(k-1)/2,\ 2k)\,]$.

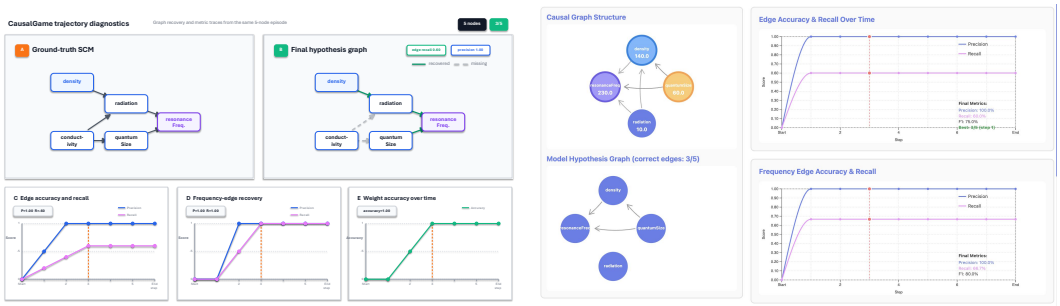
```
nodes = {resonanceFreq} + (k-1) sampled properties
target_edges = m  or  sample from [k-1, min(k(k-1)/2, 2k)]
edges = {sample(property_nodes) -> resonanceFreq}
while |edges| < target_edges:
   sample source != target
   reject frequency-as-source unless frequency_can_be_parent
   reject duplicates/cycles; otherwise add source -> target
mark all nodes observable; all but resonanceFreq controllable
sample edge coefficients and non-root intercepts
(linear suites: controllable base values are drawn per record and crystal
at run time)
```

In the main suites, `frequency_can_be_parent` is false, so `resonanceFreq` is a sink; in the `FreqParent` suites the flag is true, allowing outgoing `frequency` edges while keeping graph sizes and edge counts matched. The supplementary material includes the 50 sampled graphs of every suite together with the sampling scripts, alongside the full SCM dataset and benchmark code, so that our results can be reproduced and future agents can be evaluated on the same fixed graphs.

The main experiments use 50 sampled graphs per size for 3–7 node SCMs; Table 4 summarizes their topology distribution.

C DSL and Prompt Details

This appendix supports §4: how DSL records are presented and parsed, how the hypothesis trajectory is visualized, and the prompt templates that elicit it.



(a) Illustrative schematic (5-node example). (b) Platform screenshot of one 4-node episode.

Figure 6: Trajectory-level causal graph visualizations in *CausaLab*. The schematic and screenshot both expose the ground-truth graph, the agent's hypothesis graph, and recovery metrics over the intervention sequence. Panel (a) is a hand-specified illustration of the layout (its "same 5-node episode" subtitle refers to that illustration), not a logged run, and panel (b) shows a different graph from panel (a); neither panel is drawn from the reported suites, so no evaluation budget applies.

C.1 DSL Implementation Details

The DSL hypothesis schema is presented with field-by-field descriptions and a small worked example so the model knows the expected key names, edge format, coefficient format, and convention for referring to `frequency`. The episode configuration fixes the property names and functional family, so the parser only accepts variables emitted by the environment prompt.

C.2 Trajectory-Level DSL Visualization

The DSL records make the agent's evolving causal hypothesis auditable, rather than leaving Full System Discovery to be inferred from a final reactor prediction. Figure 6 shows this trajectory-level view with an illustrative schematic and a platform screenshot.

C.3 Prompt Templates

The benchmark uses two nested prompt templates. The first prompt governs iterative hypothesis–experiment interaction in the environment, while the second prompt frames the reactor activation objective and embeds the task-specific laboratory rules. The templates below reproduce the prompt files used in the implementation (`causalab/agents/recoma/prompts/` in the supplementary material), with non-ASCII symbols such as arrows and check marks rendered in ASCII; runtime placeholders are filled by the environment during evaluation.

Controller Prompt (rendered every step)

```
You are playing a video game about making scientific discoveries.  The game is in the style of a 2D top-
      down RPG (you are the agent in the center of the image), and as input you get both an image, as
      well as information from the user interface (provided in the JSON below) that describes your
      location, inventory, objects in front of you, the result of your last action, and the task that
      you're assigned to complete.
Because this is a game, the actions that you can complete are limited to a set of actions that are
      defined by the game. Those are also described below.
This game is played step-by-step.  At each step, you get the input that I am providing, and output a
      single action to take as the next step.

Navigation note:
Moving forward moves you in the direction you're facing. You are currently facing "{{ facing_direction
      }}"`. From your current location, the directions that you can move to (i.e. they don't have an
      object blocking them) are: {{ valid_dirs }}.

Interaction note:
You can only interact (i.e. take actions with) objects that are in your inventory, or directly (i.e. one
       square) in front of you, in the direction that you're facing.  E.g. if you want to pick an object
       up, you need to move directly in front of it, and face it, before using the pick-up action on it.
```

```
Teleportation note:
To make moving easier, you can teleport to a list of specific locations in the environment, using the
      teleport action.  In this case, 'arg1' is the name of a location. An example teleport action would
       be: `{"action": "TELEPORT_TO_LOCATION", "arg1": "school"}.


Action Format note:
{{ additional_instructions }}

**Required Action Structure** (no memory or thought fields):

Your response MUST be a single JSON object with keys in this exact order: memory -> thought -> past_data
       -> hypothesis -> experiment -> (action fields)

** CRITICAL: INCREMENTAL UPDATE RULE **
You will see a "Previous State" section showing your last `past_data` and `hypothesis` from the previous
       step.
- **ALWAYS start from the previous state and make INCREMENTAL updates**
- **DO NOT discard previous data or reset known coefficients to null**
- For `past_data`: COPY all previous entries and APPEND new observations (do not recreate from scratch)
- For `hypothesis`: COPY all previous edges/coefficients and UPDATE only what changed based on new
      evidence
- Only update when there is NEW evidence; otherwise keep the previous values unchanged

1. **memory** :a concise but comprehensive running summary that you update EVERY step using (a) your
      last memory and (b) the most recent few action-observation pairs.
   - Keep memory as complete as possible without being verbose or repeating low-level noise.
   - Include: current subgoal and plan, useful facts discovered, items/locations that matter, pending
      checks/unknowns, recent failures and why...
   - Do not list raw logs; summarize to help immediately guide the next step.
   - When you observe measurement data, property values, or frequency readings, you should remember them
       in your memory section"
   - This helps you track patterns, relationships, and make informed decisions"
   - Record key observations such as: crystal properties, frequency measurements, causal relationships
      you discover, and experimental results"
2. **thought** (string): Natural language explanation of why the next action is chosen given current
      data and hypothesis.
   - Should reference specific gaps in data or hypothesis that motivate the action
   - MUST include accessibility checks: explicitly verify that any UUIDs you reference appear in the
      interactable objects list or your inventory, and state which list (inventory or interactable)
   - If previous actions failed, explain why they failed and justify why the new action will succeed
   - If transitioning between phases (e.g., property manipulator to reactor), explain the transition
      logic

3. **past_data** (JSON array): Records all available evidence, including baseline, passive observations,
       and intervention results.
   - Structure: `[{"id":"0/N","props":{"pH":95,"Pressure":103,...},"freq":610}, {"id":"1/N","props":{"pH
     ":95,"Pressure":50,...},"freq":451}, ...]`
   - Each entry MUST include: `id` (experiment counter), `props` (all measured properties as key-value
      pairs), `freq` (resonance frequency)
   - If available, the first entry (id="0/N") is the baseline before any interventions
   - **INCREMENTAL UPDATE**: Look at "Last past_data" in Previous State -> COPY ALL existing entries ->
      APPEND new observations
   - Update rule: APPEND whenever you obtain a new informative observation, whether from passive
      evidence or from a property-manipulator intervention; keep unchanged during reactor operations
      unless genuinely new evidence appears there
   - **DO NOT recreate the array from scratch; always preserve all previous entries**

4. **hypothesis** (JSON object): Current understanding of causal structure and frequency relationships.
   - Structure: `{"edges":[{"from":"PropA","to":"PropB"},...], "freq_equation":"resonanceFreq = base +
      c_PropA*PropA + c_PropB*PropB", "coefficients":{"base":value,"c_PropA":value,"c_PropB":value}}`
   - `edges`: Directed causal relationships between properties(including frequency)
   - `freq_equation`: String representation of hypothesized frequency formula
   - `coefficients`: Numerical values for the equation (use null for unknown coefficients, use real
      value for known coefficients)
   - **CRITICAL NAMING CONVENTIONS**:
     * Coefficient names MUST follow the pattern `c_{property_name}` (e.g., `c_temperatureC`, `
      c_conductivity`)
     * ONLY include properties in freq_equation and coefficients that have edges pointing TO the
      frequency property
     * DO NOT include coefficients with value 0 (except "base") - if a property doesn't affect frequency,
       exclude it entirely
     * Example: If edges show `temperatureC->resonanceFreq` and `conductivity->resonanceFreq`, and
      coefficients calculated as c_temperatureC=2 and c_conductivity=1, then:
       - freq_equation: `"resonanceFreq = base + c_temperatureC*temperatureC + c_conductivity*
      conductivity"`
       - coefficients: `{"base":20, "c_temperatureC":2, "c_conductivity":1}` (no other properties)
```

```
   - **INCREMENTAL UPDATE**: Look at "Last hypothesis" in Previous State -> COPY ALL edges/coefficients
     -> UPDATE only parts with new evidence
   - **PRESERVE known coefficients**: If a coefficient was determined in previous steps, DO NOT reset it
      to null unless contradicted
   - Update rule: MAY update when new evidence gathered; keep unchanged during reactor operations or
     when no new discovery
   - **CRITICAL**: Update your frequency equation by REASONING about causal relationships, NOT by curve-
     fitting past_data. Do NOT directly fit observation data. Before operating the reactor, your
     hypothesis MUST contain a definite frequency equation with all coefficients determined through
     causal reasoning.

5. **experiment** (JSON object): Specification of the planned intervention when using the Property
     Manipulator, example:
   - `{"target_prop":"PropName","target_value":number}`
   - If no intervention is available or you are not planning one on this step, use `{}` instead of
     inventing a fake intervention

6. **Execution action fields** (one of the following):
   - Non-dialog actions: `"action":"ACTION_NAME", "arg1":value, "arg2":value` (arg1/arg2 optional
     depending on action)
   - Dialog option selection: `"chosen_dialog_option_int":integer`
   - Dialog Value input: `"value":number`

**Output Format** (keys in exact order):
```json
{
  "thought": "string explanation",
  "past_data": [array of observation objects],
  "hypothesis": {hypothesis object},
  "experiment": {experiment specification},
  ... action execution fields ...
}
```

Important Guidelines:

Causal Discovery Methodology (CRITICAL):
- **Derive frequency equations through CAUSAL REASONING, not data fitting**: Your hypothesis should be
     built by understanding which properties causally influence frequency, NOT by curve-fitting
     past_data or observation data.
- **Use intervention experiments when available**: When you intervene on property A and observe changes
     in property B, this reveals causal relationships. If no interventions are available, extract as
     much structure as possible from the passive observations and be explicit about what remains
     ambiguous.
- **Before operating the reactor**: Your hypothesis MUST contain a complete frequency equation with all
     coefficients determined. Do not leave coefficients as null when transitioning to reactor
     operations.
- **For reactor frequency calculation**: Read crystal properties directly from the reactor dialog text
     (`dialog_box`).
- **One-way phase transition**: Once you open/talk to the Crystal Reactor, the Property Manipulator
     becomes locked and cannot be used again.
- **If a deterministic candidate-graph list is provided in Previous State**: treat it as a hard
     consistency filter derived from the configured graph family and the collected data. Use it to
     narrow your reasoning, but still explain why the remaining candidates differ.

Data Management:
- `past_data`: Always maintain complete history. When adding passive observations or intervention
     results, append to the array (don't replace). During reactor operations, keep past_data unchanged
     unless genuinely new evidence appears.
- `hypothesis`: Update edges when causal relationships are discovered or refuted. Update coefficients
     through causal reasoning from the full evidence set. Keep hypothesis frozen during reactor
     operations.
- `thought`: Must explain the logic connecting your current hypothesis/data to the next experiment. If
     previous actions failed, explain why and justify why the new action will succeed.
- `experiment`: Must precisely specify what you plan to do next and align with the execution action
     fields.

Example of Incremental Update:
```
Previous State shows:
  Last past_data: [{"id":"0/3","props":{"pH":50},"freq":100}]
  Last hypothesis: {"edges":[{"from":"pH","to":"resonanceFreq"}],"freq_equation":"resonanceFreq = base +
      c_pH*pH","coefficients":{"base":50,"c_pH":1}}

After new observation (pH=80, freq=130):
[CORRECT] CORRECT: {"past_data":[{"id":"0/3","props":{"pH":50},"freq":100},{"id":"1/3","props":{"pH
     ":80},"freq":130}],"hypothesis":{"edges":[{"from":"pH","to":"resonanceFreq"}],"freq_equation":"
     resonanceFreq = base + c_pH*pH","coefficients":{"base":50,"c_pH":1}}}
```
```

```
[WRONG] WRONG: {"past_data":[{"id":"1/3","props":{"pH":80},"freq":130}],...}  // Missing previous entry!
[WRONG] WRONG: {"hypothesis":{"edges":[],"coefficients":{"base":null,...}}}  // Reset known coefficients!
[WRONG] WRONG: {"freq_equation":"resonanceFreq = base + c1*pH",...}  // Wrong coefficient naming! Should be c_pH
[WRONG] WRONG: {"coefficients":{"base":50,"c_pH":1,"c_pressure":0}}  // Don't include zero coefficients!
```

Hints:
- **CAUSAL REASONING FIRST**: Build your frequency equation by reasoning about causal relationships discovered through interventions, NOT by fitting curves to past_data. Your equation should reflect the causal structure, not just correlations.
- **REACTOR DIALOG FIRST**: When calculating target frequency, use the crystal properties shown directly in the reactor dialog.
- **DO NOT GO BACK**: After entering the reactor phase, do not return to the Property Manipulator. Continue with reactor frequency setting and activation.
- If your last action failed, or other last recent actions failed, please consider thinking why they failed, and trying different actions unless you believe things have changed to make failed actions work this time.
- If you don't see what you're looking for, and anticipate it might be in another location, consider teleporting to that location.
- Use verbose "thought" statements to maintain your progress, running hypotheses, your knowledge about the world, etc. Always include an ACCESSIBILITY CHECK: confirm any UUIDs you reference are from the interactable list/inventory now; otherwise first act to make them accessible.
- REMEMBER, IF YOU'RE GOING TO AN OBJECT, INSTEAD OF MOVING NORTH/EAST/SOUTH/WEST, or ROTATING, YOU SHOULD TRY TELEPORTING DIRECTLY TO OBJECTS. IT'S MUCH FASTER AND LESS ERROR-PRONE. YOU CAN TELEPORT TO AN OBJECT NO MATTER WHERE IT IS, EVEN IF ITS NOT LISTED IN THE ACCESSIBLE OBJECTS LIST. JUST REMEBER ITS UUID.
- WHEN TELEPORTING, REMEMBER TO LOOK AT THE ENTIRE OBJECT LIST IN THE OBSERVATION, NOT JUST THE ACCESSIBLE OBJECTS LIST.
- FOR ACTIONS INVOLVING TWO ARGS (E.G. USE AND PUT), ONE OF THE OBJECTS MUST BE IN YOUR INVENTORY AND YOU MUST BE NEXT TO THE OTHER OBJECT. E.G. TO USE SHOVEL ON THE SOIL, YOU MUST HAVE THE SHOVEL IN YOUR INVENTORY AND BE NEXT TO THE SOIL.
- IF YOU FIND YOURSELF WAITING FOR SOMETHING TO HAPPEN, CHECK TO SEE IF IT'S ALREADY HAPPENED. YOU MIGHT BE WAITING FOR SOMETHING THAT'S ALREADY HAPPENED.
- IF YOU FIND YOURSELF WAITING FOR A LONG TIME, YOU MIGHT BE STUCK. TRY TO REEXAMINE YOUR TASK, REASSESS WHERE YOU ARE, AND MAKE A NEW PLAN.
- IF YOU THINK YOU ARE DONE, SUBMIT WITH THE ACTION: {"action": "SUBMIT", "arg1": "Task completed!"}.
- IF YOU ARE NOT MAKING ANY PROGRESS AND CAN NOT FIGURE OUT HOW TO PROCEED, USE THE SUBMIT ACTION WITH ARG1 EXPLAINING YOUR REASONING, e.g., {"action": "SUBMIT", "arg1": "Task failed! Unable to explore the building!"}.
- If the target item is nearby but inaccessible, consider ROTATE_DIRECTION to help yourself face the target item before attempting to move or interact.

Task: {{ input_str }}

Current Environment Observation:
```json
{{ observation }}
```

{% if in_dialog %}
NOTE: You are currently in a dialog.
For reference, here is the dialog that you are currently in:
```json
{{ dialog_box }}
```

{% else %}
Valid Actions:
```json
{{ known_actions}}
```

Valid Teleport Locations:
```json
{{ teleport_destinations }}
```

CRITICAL accessibility rules (strict):
- For any `arg1`/`arg2`, choose strictly from the List of objects that are interactable (from your inventory, directly in front of you, or on adjacent tiles north/east/south/west): {{ interactable_objects }}. If an object is not in those lists, first make it accessible (MOVE/ROTATE/TELEPORT/PICK_UP) before acting.
- In your "thought", explicitly state which list you took each UUID from (inventory vs interactable list) and why it is accessible now. If not accessible, state the plan to make it accessible first.

List of objects that are interactable (from your inventory, and directly in front of you):

```
{{ interactable_objects }}

{% endif %}

Return ONLY a single JSON object. Keys MUST appear in this exact order: memory -> thought -> past_data
      -> hypothesis -> experiment -> (action+args OR chosen_dialog_option_int OR value). Do not include
      any extra text outside the JSON.
```

Task Prompt (inserted as `input_str`)

```
You are at the Causal Discovery Lab on Planet X. Quantum Crystals have interesting properties that may
      be causally related.

=== YOUR MISSION ===

THEORETICAL BACKGROUND:

1. CRYSTAL RESONANCE FREQUENCY:
   - Frequency is determined by certain properties (range: 0-100) through LINEAR relationships
   - Frequency CANNOT be directly modified by human intervention
   - Frequency can only change indirectly as a causal consequence of other property changes

2. CAUSAL STRUCTURE:
   - Properties have LINEAR causal relationships with each other
   - These relationships form a DAG (Directed Acyclic Graph):
     * No bidirectional influences (if A affects B, B cannot affect A)
     * No cycles (no circular chains like A -> B -> C -> A)
     * Acyclic structures are allowed (e.g., A -> B, A -> C, B -> D, C -> D)

3. PROPERTY INTERVENTION:
   - You can intervene on ONE property at a time using the Property Manipulator
   - When you modify a property (e.g., prop_A):
     * Other properties CANNOT be simultaneously modified by direct intervention
     * However, other properties MAY change automatically as a causal consequence
     * Frequency MAY also change if it depends on the modified property or its effects

4. MATHEMATICAL MODEL:
   - Each property has a BASE VALUE that can be adjusted (except frequency)
   - Each property's value = base_value + sum of causal influences from other properties
   - Example: If frequency depends on properties A and B:
     frequency = freq_base_value + k_A * A + k_B * B
     (Note: freq_base_value is fixed and cannot be adjusted)
   - Example: If temperature depends on properties C, D, and E:
     temperature = temp_base_value + k_C * C + k_D * D + k_E * E
     (Note: When you adjust temperature, you are actually adjusting temp_base_value)
   - IMPORTANT: When you modify a property(frequency cannot be directly modified), you are changing its
      base_value,
     which then propagates through the causal graph according to the linear relationships

CAUSAL DISCOVERY STAGE (If the Property Manipulator is available)
- There is an EXPERIMENT CRYSTAL (quantum crystal 3) currently in the Property Manipulator
- Depending on the experiment setting, you may start with zero or more existing observations and zero or
      more remaining property adjustments
- If `{{ max_uses }}` is greater than 0, you may use the Property Manipulator to discover causal
     relationships
- If `{{ max_uses }}` is 0, rely on the available observations only and proceed without Property
     Manipulator interventions
- Goal: Figure out the relationship between controllable and derived properties as far as the available
     evidence allows

REACTOR ACTIVATION STAGE
- Once you have extracted all useful information from the available observations/interventions, activate
      the Crystal Reactor
- Calculate what the crystal's resonance frequency should be based on its properties
- Place the target crystal into the Crystal Reactor
- Set the reactor's frequency to match the crystal's resonance frequency
- Tolerance: +/-{{ frequency_tolerance }} Hz

IMPORTANT NOTICE:
- The EXPERIMENT CRYSTAL (used in Phase 1) and the TARGET CRYSTAL (used in Phase 2) are DIFFERENT
     crystals
- They are not movable between locations
- You CANNOT directly measure the target crystal's frequency - you must calculate it from its properties
- The Property Manipulator shows current crystal properties
- The Property Manipulator has a usage counter showing remaining adjustments
```

```
EXISTING OBSERVATIONS (possibly zero, from previous experiments):
{{ existing_observations }}

IMPORTANT USAGE INSTRUCTIONS:
- Property Manipulator: Use TALK action, e.g. { "action": "TALK", "arg1": <property_manipulator_uuid> }
  - After dialog opens, the main menu shows available properties to adjust (e.g., Temperature, Moisture,
      etc.)
  - Step 1: Select the property you want to adjust, e.g., {"chosen_dialog_option_int": 1} for the first
      property
  - Step 2: You will enter the value input mode DIRECTLY (no intermediate menu)
  - Step 3: In value input mode, provide ONLY the "value" field in your JSON - do NOT select dialog
      options
  - Example workflow:
    * First TALK: {"reasoning": "Selecting property to adjust", "past_data": [...], "hypothesis": {...},
       "experiment": {...}, "action": "TALK", "arg1": <uuid>, "chosen_dialog_option_int": 1}
    * Next action: {"reasoning": "Setting property to test hypothesis", "past_data": [...], "hypothesis":
       {...}, "experiment": {"target_prop":"PropName","target_value":50,"purpose":"..."}, "value": 50}
      # Set value (NOT chosen_dialog_option_int!)
  - CRITICAL: When in value input mode, you will see "[Do not select this option]" in dialog options
    * This is NOT a real option - ignore it!
    * Simply provide the "value" field with your desired number
    * After setting the value, you automatically return to the main menu
  - IMPORTANT: Each property adjustment counts toward your usage limit when interventions are available.
  - Watch the usage counter if it is positive; in some experiment settings it may already be zero.
  - CRITICAL: Once you use the Crystal Reactor (set its frequency), you CANNOT use the Property
      Manipulator anymore!
  - Plan your experiments carefully: use Property Manipulator FIRST, then activate the reactor.

- Crystal Reactor: Use TALK action, e.g. { "action": "TALK", "arg1": <crystal_reactor_uuid> }
  - This opens a dialog menu showing the current resonance frequency (range: 0-10,000 Hz)
  - Step 1: Select option 1 ("Set frequency to a specific value") to enter frequency input mode
  - Step 2: After "is_in_dialog" is True and you are in frequency input mode, include the numeric "value
      " field in your JSON with the desired value (0-10000 Hz)
  - Example: {..., "chosen_dialog_option_int": 1} to select input mode, then {..., "value": <
      frequency_value>} to set frequency
  - You can select "Back to main menu" to return to the root menu
  - The reactor activates when: (1) target crystal is inside, AND (2) frequency matches crystal's
      frequency
```

D Experimental Setup Details

This appendix supports §5.1: the evidence budgets, the four evidence-acquisition regimes and what each of them can identify, and the classical baselines.

D.1 Evidence Budgets

The benchmark uses a fixed budget policy except in the explicit observation–intervention scaling suites: for a k-node graph, the observation budget is 2 and the intervention budget is $4(k-1)$, i.e. 8, 12, 16, 20 and 24 interventions for 3–7 nodes. We call this the *default budget*; every result outside the scaling suites uses it, with two exceptions: the 6-node `FreqParent` follow-up (Figure 9) ran with 12 interventions rather than 20, and the Golden runs (Table 16) replace online interventions with an injected trace. Scaling-suite settings are written nomi, meaning n initial observations followed by up to m interventions: no0i is pure observation, 0omi pure intervention, and the scaling suites' mixed regime fixes $n = 3$ on 4-node graphs and $n = 5$ on 6-node graphs.

D.2 Evidence-Acquisition Regimes

Table 5 defines the four evidence-acquisition regimes compared throughout the experiments. The tables and figures call observation-conditioned intervention the *mixed* regime.

D.3 Identifiability Across Evidence Regimes

Identifiability in *CausaLab* depends on the evidence regime and on how the records are generated. In every passive record of the main suites, each controllable variable draws its base value independently as an integer in $[0, 100]$, so every variable except `frequency` carries its own non-Gaussian exogenous term. `frequency` is the only non-controllable variable and its base value is fixed, so it is an exact linear

Regime	Initial evidence	Online?	Purpose
Pure observation	Passive records	No	Static evidence only
Pure intervention	None	Yes	Evidence from agent actions only
Obs.-conditioned (mixed)	Passive records	Yes	Do prior records improve interventions?
Golden offline trace	Passive records + injected trace	No	Fixed pre-generated intervention evidence

Table 5: The four evidence-acquisition regimes. "Online?" indicates whether the agent chooses new interventions during the episode. Each injected Golden trace holds 0–6 interventions (Appendix F.2).

Evidence regime	Exact DAG identifiable?	Skeleton identifiable?
Pure observation	In the population, yes. The non-`frequency` variables form a linear SCM with independent non-Gaussian exogenous terms, whose DAG is identifiable without faithfulness (Shimizu et al., 2006); the ICA-based argument needs only independent non-Gaussian terms, which the discrete-uniform draws satisfy. `frequency`'s parents and coefficients are its unique exact linear fit once at least k records are available. The agent sees only 3–24 records (as few as $k-1$ in the smallest settings), so both the `frequency` fit and the orientations must be estimated from very few samples	Yes, by the same argument. Faithfulness does not hold, because `frequency` is a deterministic function of its parents; that breaks conditional-independence tests (with the 12 four-node and 20 six-node passive records of Table 8, PC's Fisher-Z test hits a singular correlation matrix on 15/50 four-node and 9/50 six-node graphs) but not this argument
Pure intervention	In principle, if known-target interventions sufficiently cover the controllable variables; not guaranteed by every finite agent-chosen trace	Under the same coverage condition
Obs.-conditioned (mixed)	In principle under sufficient known-target interventions; not guaranteed by every finite agent-chosen trace	Under the same coverage condition
Golden offline trace	In principle, if the injected interventions cover the controllable variables; 10/50 released four-node traces reach a singleton interventional MEC, and 14/50 six-node traces hold no intervention	Under the same coverage condition
Hidden perturbation (mixed regime + disturbance)	Not guaranteed: an unobserved variable perturbs one to three observed nodes after each intervention	Not guaranteed: when it perturbs two or more nodes, direct adjacency and shared latent influence can remain ambiguous

Table 6: Per-regime identifiability of the hidden SCM.

function of its parents. Table 6 gives the resulting analysis; its last row is the hidden-perturbation control, which runs the mixed regime with an unobserved disturbance. With finite records, directed edge F_1 in the observation-only and hidden-perturbation settings measures recovery from evidence that need not pin down the DAG; we read these scores as regime comparisons at a matched budget, not as regime-independent measures of model ability.

To make the practical impact explicit, we rescored the four-node `GPT-5-mini` outputs of four settings (50 runs each) with Skeleton F_1 (Table 7). The small differences between directed and skeleton scores show that the reported gaps are mostly missing or spurious adjacencies rather than reversed-edge penalties.

Setting	Directed edge F_1	Skeleton F_1
Pure observation (12o0i)	0.643	0.643
Pure intervention (0o12i)	0.596	0.596
Obs.-conditioned intervention (3o12i)	0.726	0.731
Hidden perturbation (hidden count = 3)	0.671	0.694

Table 7: Skeleton-level rescoring of four-node `GPT-5-mini` runs (50 per setting).

D.4 Classical Baseline Details

The observational baselines of §5.1 receive exactly the passive records of the `GPT-5.2-high` observation-only runs they are compared with, 12 records at 4 nodes and 20 at 6 nodes (the 12o0i and 20o0i rows of Tables 14 and 15), and the paired-shift control is capped by the same per-episode intervention budget as the agents (Appendix D.1), though it needs only a quarter of it. PC (Fisher-Z test, $\alpha = 0.05$) and DGES, causal-learn's variant of GES (Chickering, 2002) for deterministic relations (BIC score, exact-search phase skipped), recover skeletons (with orientations only up to the Markov equivalence class from purely observational data), so they are scored with Skeleton F_1; DirectLiNGAM, NOTEARS, and the paired-shift control emit directed graphs and are scored with Directed F_1. PC fails on some episodes because deterministic structural relations make its Fisher-Z correlation matrix singular; it succeeds on 35/50 four-node and 41/50 six-node graphs, and Table 8 averages PC over the graphs where it succeeds; the `GPT-5.2-high` rows average over all 50 graphs. The paired-shift control receives the variable list and which variables are controllable, and nothing else. It spends one transition per controllable variable: for each controllable c it records the pre-intervention state, applies the environment's set-to-value intervention $c \leftarrow x_c + \delta$, and records the post-state, so the paired difference $T[c][j] = (x'_j - x_j)/\delta$ is c's *total* effect on every other variable j. Because the main-suite equations are linear with state-independent coefficients, these differences do not depend on where in state space the shift is taken, so the shifts can be chained with each post-state serving as the next pre-state and the control needs a single observation rather than one per shift. All main-suite coefficients are strictly positive, so no two directed paths can cancel and $T[c][j] \neq 0$ exactly when c is an ancestor of j: the nonzero pattern of T is the transitive closure of the true graph, which differs from the graph itself in 164 of the 250 main-suite graphs. Direct effects are isolated by splitting each path sum on its first edge, $T[c][j] = W[c][j] + \sum_k W[c][k]\,T[k][j]$. Every contributing k lies strictly between c and j, so sorting variables by ancestor-set size gives a valid topological order in which back-substitution recovers W, and edge $c \to j$ is emitted when $|W[c][j]| > 0.5$. Across all 250 graphs every true edge recovers its integer coefficient (≥ 1) and every non-edge is zero up to floating-point error ($< 10^{-12}$), so any threshold between these values gives the same graph. Orientation is free, because shifting c moves descendants and never ancestors; the control is therefore never subject to the Markov-equivalence ambiguity that limits PC and DGES. It needs no conditional-independence test or score search, and in place of faithfulness it uses only the no-cancellation property that the strictly positive coefficients guarantee.

This recovers Directed $F_1 = 1.000$, with exact structural coefficients, on all 250 main-suite graphs from $k - 1$ transitions. Spending the agents' full budget changes nothing, because repeated shifts reproduce the same noiseless difference matrix. The $k - 1$ transitions are also necessary: dropping the last shift leaves that variable's outgoing edges unidentified and turns paths through it into spurious direct edges, and Directed F_1 falls to 0.64–0.87 across graph sizes (0.60–0.87 over every choice of dropped shift). Two caveats bound how this control should be read. It relies on the suites' linear mechanisms and on `resonanceFreq` being the only non-controllable variable, both of which the agent's prompt also states, and on two properties the agent is not told: strictly positive coefficients, and `resonanceFreq` being a sink, so the design emits no outgoing edges for it. On nonlinear suites the path-splitting identity does not hold, and on hidden-disturbance suites the paired differences no longer equal total effects. It is therefore an upper bound on what the environment's interventional signal permits, not a claim that the task is easy. Separately, the shift must be representable at the environment's two-decimal rounding: recovery (graph and coefficients) is exact whenever δ is a nonzero multiple of 0.01 (the reported runs use $\delta = 10$); small shifts that are not, such as $\delta = 0.015$, lose exactness (Directed F_1 0.93/0.85 on 4/6 nodes), and $|\delta| = 0.001$ collapses recovery to Directed $F_1 = 0$. The supplementary material includes the control as `code/baselines/paired_shift_baseline.py`,

which imports the environment's own intervention oracle rather than reimplementing it, together with its output `data/classical_baselines.json`.

Method	Metric	4-node	6-node
`GPT-5.2-high` (12o0i/20o0i)	Directed F_1	0.648	0.471
`GPT-5.2-high` (12o0i/20o0i)	Skeleton F_1	0.648	0.471
DirectLiNGAM	Directed F_1	0.628	0.523
NOTEARS	Directed F_1	0.883	0.780
DGES	Skeleton F_1	0.832	0.702
PC (successful graphs)	Skeleton F_1	0.638	0.518
Paired shift ($k-1$ shifts)	Directed F_1	1.000	1.000

Table 8: Classical causal-discovery baselines on the agents' passive records, and a paired-shift intervention control that uses $k-1$ transitions, a quarter of the agents' budget, and attains the same 1.000 at every graph size (3–7 nodes). PC is averaged over the graphs where it succeeds (35/50 at 4 nodes, 41/50 at 6 nodes), while the `GPT-5.2-high` rows average over all 50 graphs; none of the agent's predicted edges is reversed (its directed precision is 1.000 at both sizes), so its Skeleton and Directed scores coincide.

E RQ1 Details: Target Mechanism Recovery versus Full System Discovery

The main text uses these results to separate Target Mechanism Recovery from Full System Discovery. Table 9 and Figure 7 support the hard-quadratic comparison by showing that the main degradation is in Target Mechanism Recovery (the `frequency` coefficients), not in Full System Discovery. Figure 8 and Table 10 expand the hidden-noise analysis: ordinary hidden count/range perturbations leave task accuracy near baseline, while perturbing the `frequency` node itself lowers it to 26–40%; the latter comparison is confounded by a prompt variant and a later run batch. The `FreqParent` follow-up in Figure 9 and Table 11 tests a related modelling choice: allowing `frequency` outgoing edges improves prediction but lowers all-edge recovery.

All follow-up arms here are `GPT-5-mini` runs from April 2026 (linear, hard-quadratic and hidden count/range on April 5; `hidden-freqnode` and `FreqParent` on April 29–30). The baseline rows of Tables 10 and 11 are the main-suite runs from February (4 nodes) and March 2026 (6 nodes). Agents in the April runs intervened far less (roughly 1–3 interventions per episode, against about 5 and 12 in the 4- and 6-node baselines). The linear arm of Table 9 measures this batch effect: it repeats the 4-node baseline setting (same 50 graphs, prompts, 2 initial observations and 12-intervention cap; environment-call cap 62 rather than 53). Its accuracy matches the baseline (46 vs. 48%), but it submits fewer edges (all-edge recall 0.560 vs. 0.846) and its all-edge F_1 is 0.649 against 0.865, inside the 0.614–0.698 range of the hidden count/range settings. The all-edge gap between the baseline and those settings is therefore partly a between-batch difference. Root F_1 is not comparable across batches either: the February–March runs usually list every non-target variable as a root, and the April runs mostly do not. Graph metrics should therefore be compared within a batch, as in the linear versus hard-quadratic comparison.

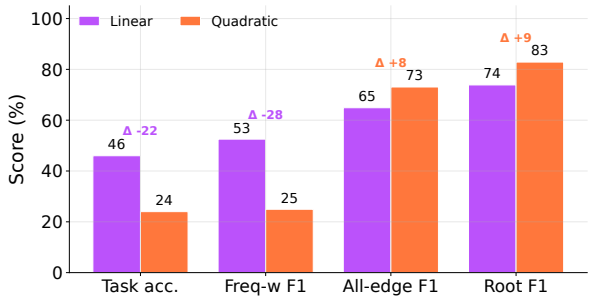


Figure 7: Matched 4-node comparison between linear and hard-quadratic mechanisms for `GPT-5-mini`, both at the default budget (2 initial observations, 12 interventions). Topology is fixed; the functional form changes, and with it the coefficient scale, the hard-quadratic generator's fixed observation presets, and the equation template in the prompt. Task accuracy and `frequency`-weight F_1 collapse while all-edge and root-node F_1 are preserved or even rise — agents lose the mechanism, not the graph.

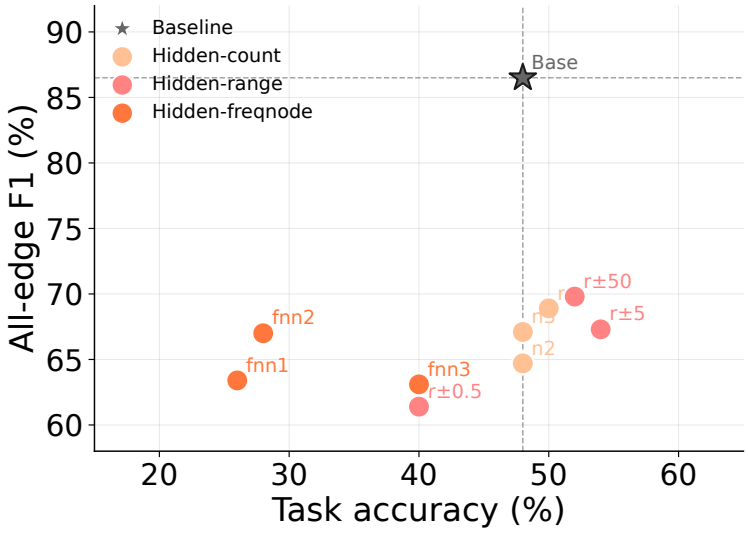


Figure 8: Hidden-noise diagnostics for `GPT-5-mini` across all ten 4-node settings, each at the default budget (2 initial observations, 12 interventions), plotted as (task accuracy, all-edge F_1) and colored by noise category. The dashed cross marks the unperturbed baseline. Hidden-count and Hidden-range settings drop F_1 but keep accuracy near baseline; the `freqnode` family collapses accuracy without further F_1 loss, exposing a fragile parent–target shortcut.

Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
Linear	46	0.739	0.887/0.560/0.649	0.745	0.525
Hard-quadratic	24	0.829	0.892/0.663/0.731	0.741	0.249

Table 9: Linear versus hard-quadratic mechanisms for `GPT-5-mini` on matched 4-node graphs, with identical node and edge sets and the default budget (2 initial observations, 12 interventions) in both arms. Full System Discovery (root and all-edge F_1) survives the switch to quadratic dynamics; the loss is concentrated in the coefficients of the `frequency` equation, i.e. in Target Mechanism Recovery. Quadratic-family coefficients are scored on a separate path: each parent contributes both a linear and a squared coefficient, agents report them as two keys whose order is not consistent across episodes, and the true values are fitted floats spanning two orders of magnitude, so both halves are scored, either key order is accepted, and the comparison is relative at 20% rather than the linear family's 10^{-6} absolute. Linear-family coefficients keep the 10^{-6} absolute rule. Both arms are April 2026 runs; the linear arm's graph-metric gap to the 4-node baseline of Table 10 is a batch effect (Appendix E).

Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
4-node main baseline	48	0.559	0.909/0.846/0.865	0.883	0.622
hidden count $= 1$	50	0.708	0.854/0.610/0.689	0.781	0.477
hidden count $= 2$	48	0.747	0.838/0.576/0.647	0.781	0.507
hidden count $= 3$	48	0.672	0.883/0.581/0.671	0.799	0.493
hidden range $= \pm 0.5$	40	0.666	0.818/0.534/0.614	0.729	0.389
hidden range $= \pm 5$	54	0.755	0.876/0.610/0.673	0.766	0.546
hidden range $= \pm 50$	52	0.675	0.920/0.605/0.698	0.818	0.563
hidden `freqnode` $= 1$	26	0.805	0.800/0.573/0.634	0.678	0.245
hidden `freqnode` $= 2$	28	0.699	0.851/0.603/0.670	0.735	0.368
hidden `freqnode` $= 3$	40	0.623	0.850/0.537/0.631	0.789	0.408

Table 10: Hidden-noise robustness for `GPT-5-mini` on 4-node graphs under exact-context re-evaluation, every row at the default budget (2 initial observations, 12 interventions). The table includes the 4-node main baseline, standard hidden settings, and `hidden-freqnode` settings, all on the same 50 graphs. Each setting has one hidden variable, and every hidden setting adds a notice about it to the task prompt. Hidden count is the number of variables it shifts (range ± 5); hidden range is the shift size, with one shifted variable. Standard settings shift one to three non-target variables, always including a direct parent of `resonanceFreq`; `freqnode` settings shift `resonanceFreq` alone ($= 1$) or together with one or two other variables. Hidden count $= 1$ and hidden range $= \pm 5$ are the same configuration run twice, with the crystal property values redrawn in each run; their differences (4 points of accuracy, 0.07 frequency-weight F_1) indicate run-to-run variability. The `hidden-freqnode` arms also use a hidden-variable variant of the agent prompt, which adds guidance on post-intervention values and omits the default prompt's hint block, and their task prompt states that the hidden variable may perturb `resonanceFreq` or its direct parents. Because of these prompt differences and their later run date, their accuracy is not a controlled comparison with the standard hidden rows. The baseline row comes from an earlier run batch than all hidden rows (Appendix E).

Suite	Setting	n	Acc.	All-edge F_1	Freq-edge F_1	Freq-weight F_1
4-node main	Baseline	50	48.0	0.865	0.883	0.622
4-node FreqParent	FreqParent	50	56.0	0.606	0.733	0.591
6-node main	Baseline	50	24.0	0.677	0.718	0.473
6-node FreqParent	FreqParent	50	36.0	0.387	0.582	0.427

Table 11: FreqParent follow-up under exact-context re-evaluation (GPT-5-mini). Allowing `resonanceFreq` as a parent improves prediction but degrades Full System Discovery. Every row has 2 initial observations and 12 interventions, except the 6-node baseline, which has the default 20. The baseline rows come from an earlier run batch (Appendix E), so part of each all-edge gap may be a batch effect.

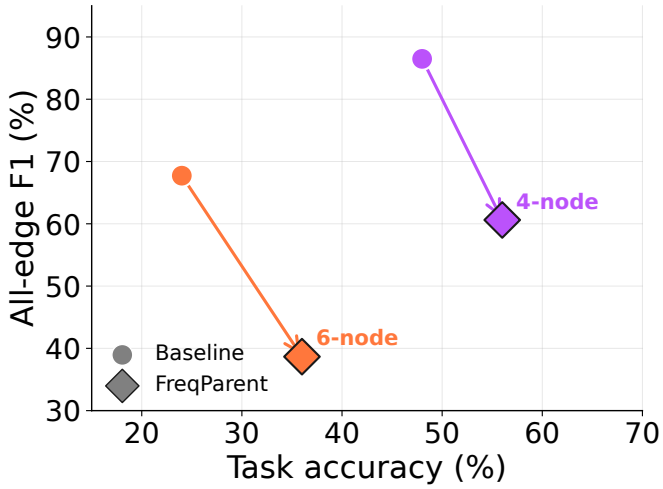


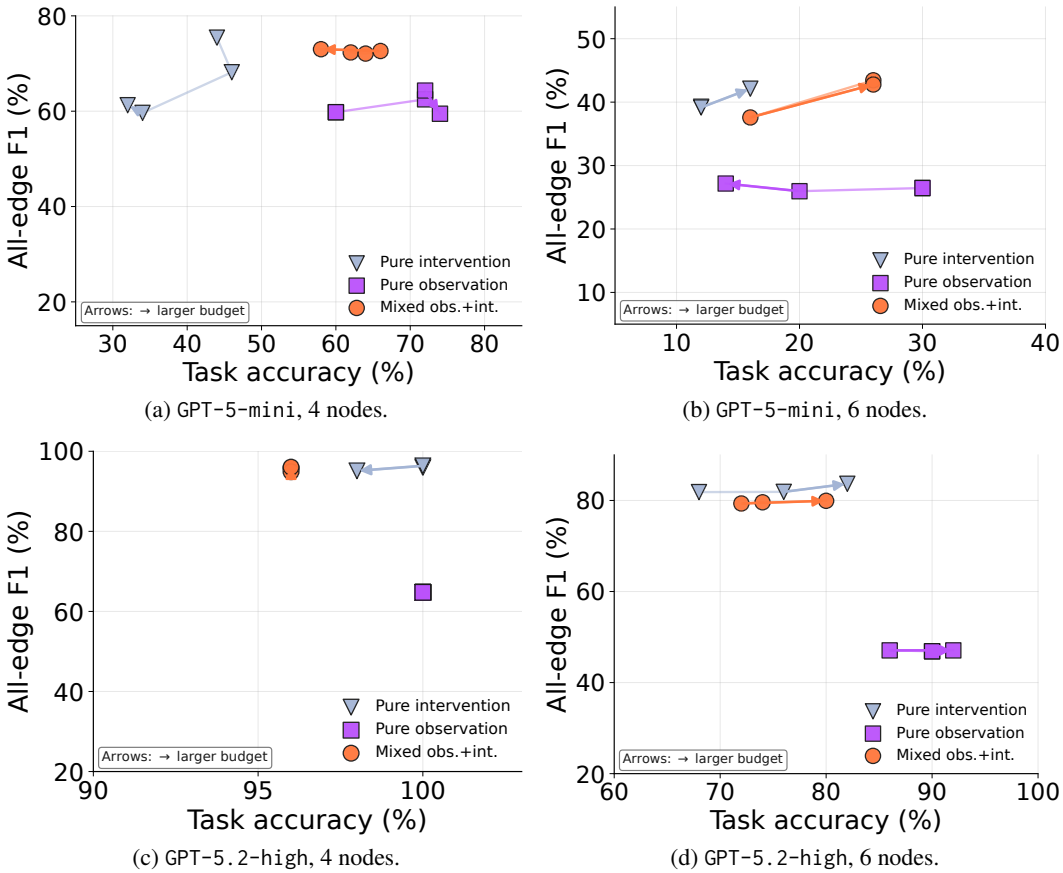
Figure 9: `FreqParent` experiments on `GPT-5-mini` in (task accuracy, all-edge F_1) space on 4- and 6-node graphs, all with 2 initial observations. Baseline $\rightarrow$ `FreqParent` arrows move down-right on both graph sizes: prediction accuracy rises while graph fidelity falls. Both 4-node arms use 12 interventions; at 6 nodes the baseline uses 20 and `FreqParent` 12, so the 6-node arrow is not budget-matched.

F RQ2 Details: Interaction Regimes and Offline Intervention Traces

This appendix supports §5.3. Appendix F.1 gives the full observation–intervention budget sweeps behind the matched-budget comparison of Figure 3, and Appendix F.2 gives the Golden offline-trace control.

F.1 Observation–Intervention Scaling

The scaling appendix supports the RQ2 claim that observations and interventions help different axes of performance. Figure 10 unpacks the matched-budget comparison of Figure 3 into the full budget sweep behind each point, Figure 13 summarizes the observation-only-to-mixed shift at the same matched budget across all four scaling families and the Golden control, Figures 11 and 12 show the task and all-edge recovery curves, and Tables 12–15 provide the numeric metric breakdowns for each model/size suite. In these tables the *Mixed intervention* block is the observation-conditioned intervention regime of Table 5; its first row repeats the zero-intervention setting from the pure-observation block as the start of the mixed curve.



(a) `GPT-5-mini`, 4 nodes. (b) `GPT-5-mini`, 6 nodes.

(c) `GPT-5.2-high`, 4 nodes. (d) `GPT-5.2-high`, 6 nodes.

Figure 10: Every scaling-suite setting in (task accuracy, all-edge F_1) space, one polyline per regime ordered by budget, with the arrow pointing toward the largest budget; 50 graphs per point. Pure observation (no0i) and pure intervention (0omi) use $n, m \in \{3, 6, 12, 24\}$ for `GPT-5-mini` at 4 nodes, $\{3, 6, 12\}$ for `GPT-5.2-high` at 4 nodes, and $\{5, 10, 20\}$ at 6 nodes; the mixed regime adds 3 (4 nodes) or 5 (6 nodes) initial observations to the same intervention grid. Figure 3 reads one matched-budget slice of these panels. Pure observation has the lowest all-edge F_1 at every matched budget except one (`GPT-5-mini`, 4 nodes, budget 12: 0.643 for 12o0i against 0.596 for 0o12i); which interventional regime leads does not — the two trade places as the budget grows in every family except `GPT-5.2-high` at 6 nodes, so we treat them as indistinguishable.

The next two tables unpack the `GPT-5-mini` scaling runs behind Figure 10. At 4 nodes, observation-only budgets of six or more records give the best endpoint prediction (72–74%). At 6 nodes all accuracies sit near the floor (12–30%), extra observations lower accuracy, and at the ten-round matched budget the mixed regime beats observation-only (26% against 20%). In both tables the highest all-edge F_1 comes from budgets that include interventions.

Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
Pure observation					
3o0i	60	0.542	0.905/0.461/0.598	0.869	0.608
6o0i	72	0.545	0.930/0.486/0.625	0.901	0.727
12o0i	72	0.572	0.933/0.504/0.643	0.881	0.737
24o0i	74	0.542	0.898/0.460/0.595	0.864	0.737
Pure intervention					
0o3i	44	0.635	0.911/0.673/0.754	0.787	0.660
0o6i	46	0.509	0.873/0.595/0.682	0.773	0.703
0o12i	34	0.488	0.852/0.497/0.596	0.685	0.531
0o24i	32	0.453	0.870/0.509/0.613	0.664	0.549
Mixed intervention					
3o0i	60	0.542	0.905/0.461/0.598	0.869	0.608
3o3i	62	0.668	0.922/0.625/0.723	0.839	0.729
3o6i	64	0.716	0.937/0.616/0.721	0.877	0.737
3o12i	66	0.719	0.922/0.632/0.726	0.845	0.613
3o24i	58	0.749	0.919/0.634/0.730	0.843	0.609

Table 12: GPT-5-mini scaling on 4-node graphs. Rows are grouped into pure observation, pure intervention, and mixed intervention blocks. Task accuracy favors pure observation, while Full System Discovery favors settings that intervene: the best all-edge F_1 here is 0.754 under pure intervention and 0.730 under the mixed regime, against 0.643 for the best observation-only setting.

Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
Pure observation					
5o0i	30	0.214	0.853/0.168/0.265	0.622	0.310
10o0i	20	0.245	0.848/0.165/0.260	0.580	0.186
20o0i	14	0.271	0.812/0.174/0.272	0.584	0.165
Pure intervention					
0o5i	12	0.188	0.810/0.286/0.393	0.506	0.409
0o10i	12	0.165	0.774/0.295/0.392	0.500	0.374
0o20i	16	0.213	0.805/0.321/0.421	0.500	0.399
Mixed intervention					
5o0i	30	0.214	0.853/0.168/0.265	0.622	0.310
5o5i	26	0.378	0.868/0.344/0.434	0.661	0.495
5o10i	16	0.318	0.849/0.275/0.376	0.590	0.396
5o20i	26	0.342	0.841/0.320/0.428	0.676	0.435

Table 13: GPT-5-mini scaling on 6-node graphs. Rows are grouped into pure observation, pure intervention, and mixed intervention blocks. Both interventional regimes improve Full System Discovery over pure observation (best all-edge F_1 0.434 mixed and 0.421 pure intervention, against 0.272) even though all end-task accuracies remain low.

The stronger-model tables below expose the complementary failure mode. Even when `GPT-5.2-high` solves most endpoint predictions under observation-only budgets, the graph metrics still require interventions.

Figure 11 plots task accuracy for both models and graph sizes; read against the all-edge curves in Figure 12, it shows that better endpoint `frequency` prediction does not by itself certify Full System Discovery.

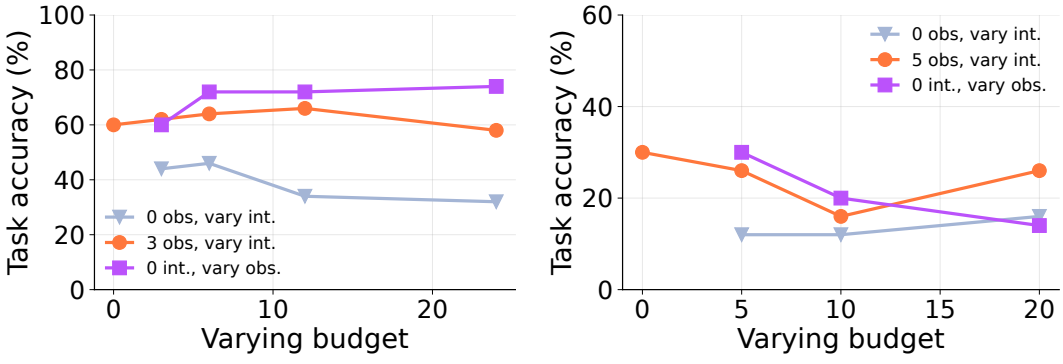
Finally, Figure 12 plots all-edge recovery directly for the same four suites.

Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
Pure observation					
3o0i	100	0.544	1.000/0.488/0.648	1.000	1.000
6o0i	100	0.557	1.000/0.488/0.648	1.000	1.000
12o0i	100	0.541	1.000/0.488/0.648	1.000	1.000
Pure intervention					
0o3i	100	0.973	0.958/0.969/0.960	1.000	1.000
0o6i	100	0.930	0.971/0.961/0.964	0.990	0.990
0o12i	98	0.920	0.960/0.951/0.951	0.993	0.993
Mixed intervention					
3o0i	100	0.544	1.000/0.488/0.648	1.000	1.000
3o3i	96	0.947	0.971/0.954/0.958	0.990	0.970
3o6i	96	0.947	0.980/0.935/0.949	0.987	0.987
3o12i	96	0.900	0.974/0.956/0.960	0.992	0.992

Table 14: GPT-5.2-high scaling on 4-node graphs. Rows are grouped into pure observation, pure intervention, and mixed intervention blocks. Observation-only settings already solve the reactor task but recover only the edges into `frequency` (all-edge precision 1.000, recall 0.488); both interventional regimes recover nearly the whole graph (best all-edge F_1 0.964 pure intervention and 0.960 mixed, against 0.648).

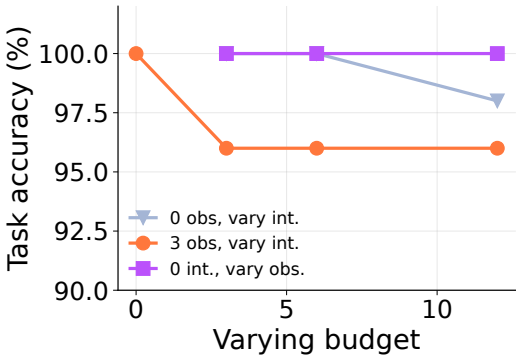
Setting	Acc.	Root F_1	All-edge P/R/F_1	Freq-edge F_1	Freq-weight F_1
Pure observation					
5o0i	90	0.309	0.990/0.316/0.469	1.000	1.000
10o0i	86	0.313	1.000/0.316/0.471	1.000	0.980
20o0i	92	0.329	1.000/0.316/0.471	1.000	1.000
Pure intervention					
0o5i	68	0.753	0.925/0.778/0.818	0.949	0.949
0o10i	76	0.640	0.868/0.805/0.819	0.910	0.894
0o20i	82	0.753	0.915/0.810/0.837	0.945	0.920
Mixed intervention					
5o0i	90	0.309	0.990/0.316/0.469	1.000	1.000
5o5i	74	0.731	0.885/0.767/0.796	0.957	0.952
5o10i	72	0.687	0.874/0.778/0.793	0.939	0.920
5o20i	80	0.767	0.889/0.778/0.799	0.957	0.957

Table 15: GPT-5.2-high scaling on 6-node graphs. Rows are grouped into pure observation, pure intervention, and mixed intervention blocks. Observation-only settings favor end-task success; intervention favors Full System Discovery, and here pure intervention leads it (best all-edge F_1 0.837, against 0.799 mixed and 0.471 observation-only).

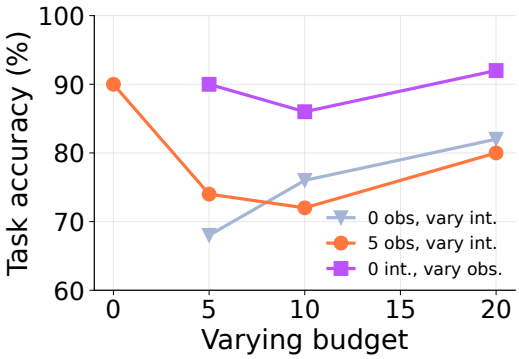


GPT-5-mini, 4 nodes.

GPT-5-mini, 6 nodes.

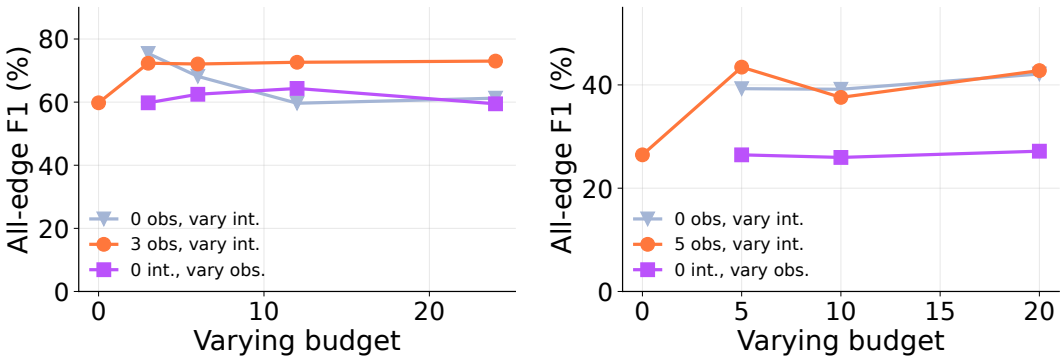


GPT-5.2-high, 4 nodes.



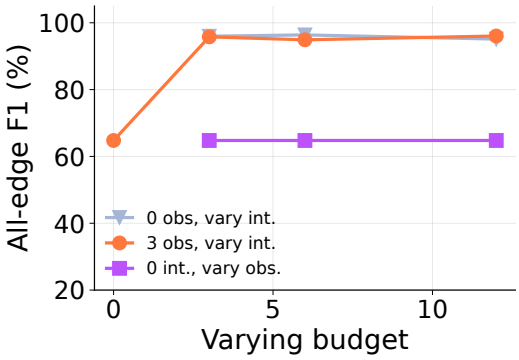
GPT-5.2-high, 6 nodes.

Figure 11: Observation/intervention scaling curves of `frequency` prediction score under different interaction modes, 50 graphs per point. Each legend entry names the count held fixed; the x-axis is the count that varies (interventions m on the pure-intervention and mixed curves, observations n on the pure-observation curve), so a mixed point's total budget is $x + 3$ at 4 nodes and $x + 5$ at 6 nodes. In the `GPT-5.2-high` 4-node panel the pure-intervention points at $x = 3$ and 6 (100%) lie under the pure-observation points. Exact settings and values are in Tables 12–15.

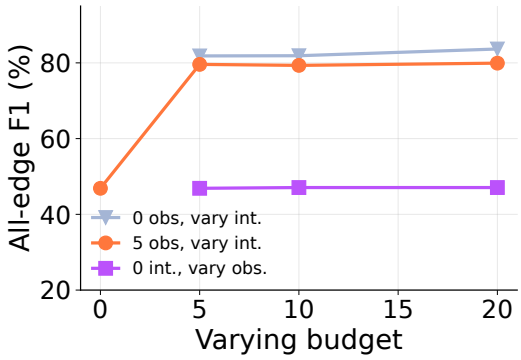


GPT-5-mini, 4 nodes.

GPT-5-mini, 6 nodes.



GPT-5.2-high, 4 nodes.



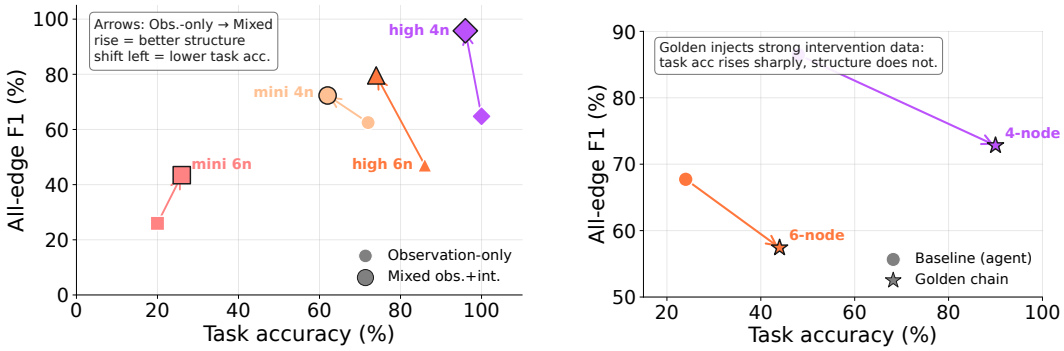
GPT-5.2-high, 6 nodes.

Figure 12: Observation/intervention scaling curves of all-edge recovery F_1 score under different interaction modes, 50 graphs per point. Axes and legends follow Figure 11: the x-axis is the varied count, and a mixed point's total budget is $x + 3$ at 4 nodes and $x + 5$ at 6 nodes.

F.2 Intervention-Trace Controls

The Golden follow-up tests whether better intervention traces alone are enough to recover the full SCM. Each Golden episode adds a pre-generated trace to the initial observations, and the agent makes no online interventions. A trace holds two passive records followed by 3–6 intervention records on 4-node graphs and 6 on 6-node graphs, except for 14 of the 50 six-node graphs, where the trace search found no qualifying chain and the trace holds only the two passive records. The traces target a singleton interventional Markov equivalence class (MEC), but only 10 of the 50 four-node traces are verified to reach one. Table 16 gives the numbers behind the main-text claim that these traces greatly improve endpoint prediction, yet do not produce matching gains in all-edge recovery. Figure 13 sets this shift (right) beside each scaling family's observation-only-to-mixed shift from Appendix F.1 (left).

Several limits apply to this comparison. The baselines are the main-suite runs, which come from an earlier run batch (see the batch note in Appendix E), so part of the all-edge gap may be a batch effect; their agents also had an online budget of 12 (4 nodes) or 20 (6 nodes) interventions. At 6 nodes the aggregate all-edge drop comes from the 14 episodes without injected interventions (21.4% accuracy, 0.186 all-edge F_1); on the other 36 graphs Golden reaches 52.8% and 0.725, against 27.8% and 0.685 for the baseline on the same graphs. Finally, the first intervention of each trace is computed from the first passive record rather than the second, so the first listed transition is not a valid adjacent-state change.



(a) Observation-only to mixed.

(b) Baseline to Golden.

Figure 13: Arrows in (task accuracy, all-edge F_1) space. Left: each arrow runs from a scaling family's observation-only setting to its mixed setting at the matched budget of Figure 3 (six rounds at 4 nodes, ten at 6); the mixed regime gains 10–33 points of all-edge F_1, costing 4–12 accuracy points except for `GPT-5-mini` at 6 nodes, where accuracy also rises by 6. Right: baseline to Golden (`GPT-5-mini`, 4 and 6 nodes); baselines use the default budget (2 initial observations; 12 interventions at 4 nodes, 20 at 6), Golden runs get 2 initial observations plus the injected trace and make no online interventions; injected interventional Markov equivalence class (MEC)-minimizing traces improve prediction but lower aggregate all-edge recovery; at 6 nodes the drop comes from the 14 traces that hold no intervention.

Suite	Setting	n	Acc.	All-edge F_1	Freq-edge F_1	Freq-weight F_1
4-node main	Baseline	50	48.0	0.865	0.883	0.622
4-node main	Golden	50	90.0	0.728	0.743	0.720
6-node main	Baseline	50	24.0	0.677	0.718	0.473
6-node main	Golden	50	44.0	0.574	0.715	0.543

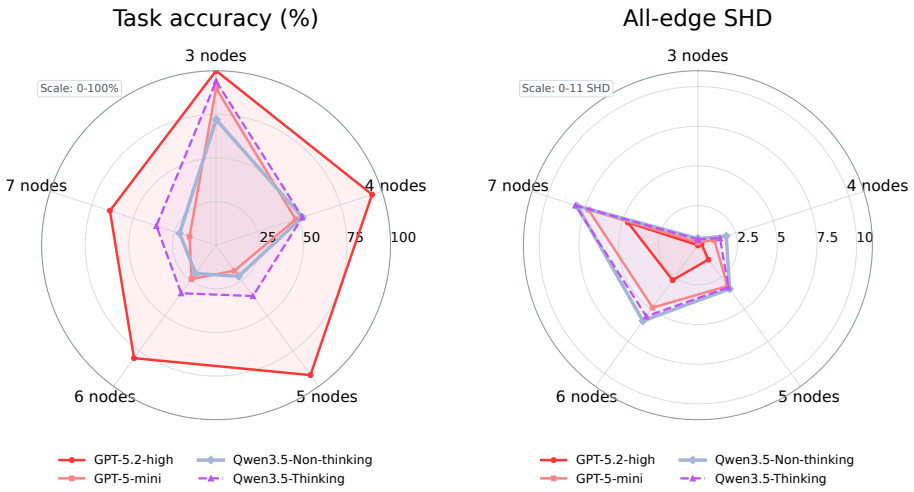
Table 16: Golden follow-up under exact-context re-evaluation (GPT-5-mini). Injected low-MEC intervention traces strongly improve frequency prediction but do not improve aggregate all-edge recovery. Baselines use the default budget (2 initial observations; 12 interventions at 4 nodes, 20 at 6); Golden runs receive 2 initial observations plus the injected trace and make no online interventions.

G RQ3 Details: Model Family and Graph-Size Scaling

This appendix supports §5.4: the GPT and Qwen sweeps over 3–7 node graphs, and the `DeepSeek-V4` family on the 6-node suite.

G.1 GPT and Qwen across Graph Sizes

These tables give the full numeric support for the model-family and graph-size claims in RQ3. Table 17 combines the 3–7 node GPT and Qwen sweeps behind the model-family radar plots, while Figure 14 shows the four-model radar comparison, Figure 16 shows the corresponding GPT absolute metric trajectories, and Figure 15 decomposes the capability gap by metric. Both Qwen configurations are `Qwen3.5-35B-A3B`; `Qwen3.5-Thinking` runs with thinking enabled and a 4,096-token thinking budget (24,576 max output tokens, against 8,192 for `Qwen3.5-Non-thinking`).



(a) Task accuracy.

(b) Directed all-edge SHD.

Figure 14: Four-model comparison across the 3–7 node main suites at the default budget (2 initial observations, $4(k-1)$ interventions). Each vertex fixes graph size and compares `GPT-5.2-high`, `GPT-5-mini`, `Qwen3.5-Thinking`, and `Qwen3.5-Non-thinking`. Lower SHD is better; a reversed edge counts as one directed SHD error. Task values are endpoint reactor accuracies.

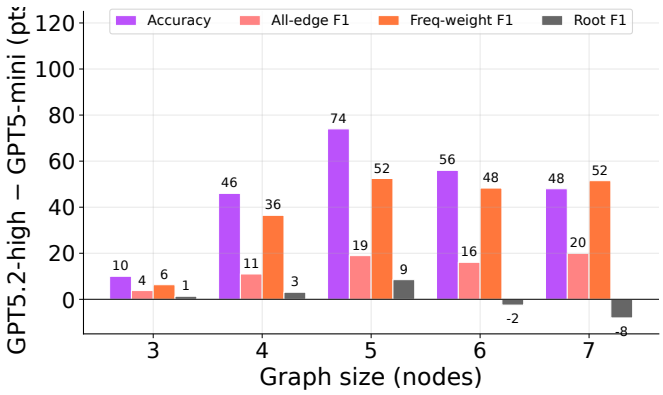
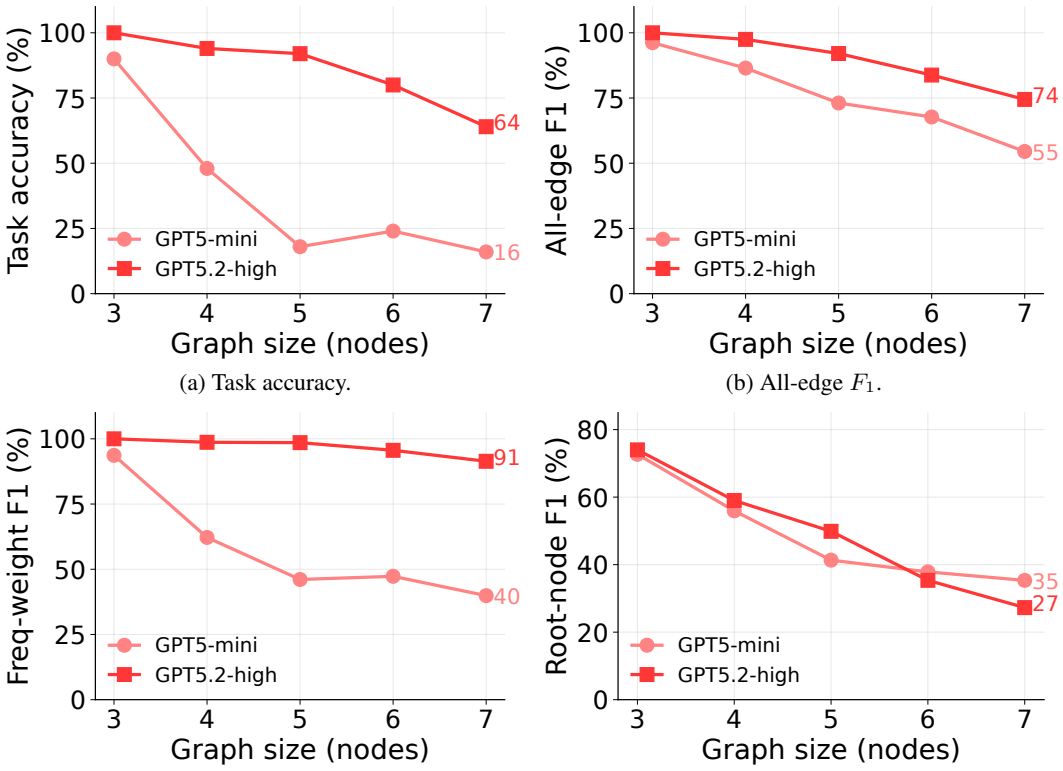


Figure 15: Capability gap (`GPT-5.2-high` − `GPT-5-mini`) in percentage points on the 3–7 node main suites at the default budget (2 initial observations, $4(k-1)$ interventions). Scaling concentrates in accuracy and `frequency`-weight F_1. The root-node bars turn negative at 6–7 nodes (-2 and -8 points), but we do not read that as the larger model stalling: root-node F_1 scores a separately elicited root list rather than the roots implied by the recovered graph, so for `GPT-5-mini` it partly measures an answer convention (see Appendix G.2). The all-edge F_1 bars, together with the SHD column of Table 17 and Figure 14, carry the scaling claim in §5.4.

Model	Nodes	Rows	Acc.	Root F_1	All-edge P/R/F_1	SHD↓	Freq-edge F_1	Freq-weight F_1
`GPT-5-mini`	3	50	90.00	0.727	0.983/0.950/0.962	0.180	0.960	0.937
`GPT-5-mini`	4	50	48.00	0.559	0.909/0.846/0.865	1.080	0.883	0.622
`GPT-5-mini`	5	50	18.00	0.413	0.863/0.660/0.731	3.180	0.707	0.461
`GPT-5-mini`	6	50	24.00	0.378	0.794/0.620/0.677	4.860	0.718	0.473
`GPT-5-mini`	7	50	16.00	0.353	0.733/0.472/0.545	7.260	0.585	0.399
`GPT-5.2-high`	3	50	100.00	0.740	1.000/1.000/1.000	0.000	1.000	1.000
`GPT-5.2-high`	4	50	94.00	0.590	0.969/0.984/0.975	0.240	0.987	0.987
`GPT-5.2-high`	5	50	92.00	0.499	0.954/0.904/0.921	1.120	0.985	0.985
`GPT-5.2-high`	6	50	80.00	0.354	0.878/0.840/0.838	2.720	0.956	0.956
`GPT-5.2-high`	7	50	64.00	0.273	0.855/0.703/0.745	4.640	0.921	0.914
`Qwen3.5-Non-thinking`	3	50	72.00	0.853	0.987/0.850/0.900	0.420	0.973	0.870
`Qwen3.5-Non-thinking`	4	50	52.00	0.769	0.901/0.663/0.737	1.880	0.830	0.691
`Qwen3.5-Non-thinking`	5	50	22.00	0.668	0.894/0.613/0.696	3.400	0.741	0.513
`Qwen3.5-Non-thinking`	6	50	20.00	0.459	0.806/0.474/0.556	5.900	0.737	0.493
`Qwen3.5-Non-thinking`	7	50	22.00	0.514	0.725/0.418/0.495	8.000	0.678	0.508
`Qwen3.5-Thinking`	3	50	94.00	0.905	1.000/0.850/0.905	0.367	0.986	0.983
`Qwen3.5-Thinking`	4	50	52.00	0.813	0.942/0.750/0.801	1.460	0.875	0.739
`Qwen3.5-Thinking`	5	50	36.00	0.719	0.917/0.621/0.707	3.280	0.841	0.614
`Qwen3.5-Thinking`	6	50	34.00	0.473	0.831/0.468/0.556	5.540	0.790	0.572
`Qwen3.5-Thinking`	7	50	36.00	0.520	0.734/0.376/0.454	8.120	0.708	0.568

Table 17: GPT and Qwen model comparison across the 3–7 node main suites at the default budget (2 initial observations, $4(k-1)$ interventions). Acc. is endpoint reactor accuracy in percent; each P/R/F_1 cell reports precision, recall, and F_1 for the corresponding recovery target. SHD is directed all-edge structural Hamming distance, with reversed edges counted as one error. Accuracy is always averaged over all 50 sampled graphs in a row, so an episode that ends without a recorded outcome counts as a miss rather than being dropped (6 and 10 `GPT-5-mini` episodes at 6 and 7 nodes, 1 and 6 `Qwen3.5-Thinking` episodes at 3 and 4 nodes). The recovery metrics average over all 50 episodes in every row except `Qwen3.5-Thinking` at 3 nodes, where the one episode that stopped on API connection errors before writing its outcome was excluded from re-evaluation (49). Root F_1 is not comparable across families: the GPT rows score a root list elicited in a separate call (Appendix G.2), while the Qwen rows were re-evaluated without API calls and score the parentless ancestors of `frequency` in the final hypothesis graph. Scored that way, `GPT-5.2-high` reaches 1.000, 1.000, 0.960, 0.813 and 0.730 at 3–7 nodes and `GPT-5-mini` 0.973, 0.913, 0.783, 0.679 and 0.575, above both Qwen variants at every size.



(a) Task accuracy.

(b) All-edge F_1.

(c) `frequency-weight` F_1.

(d) Root-node F_1.

Figure 16: Absolute trajectories of task accuracy and three recovery metrics across graph sizes for `GPT-5-mini` and `GPT-5.2-high` on the main suites. Every size uses the default budget (2 initial observations, $4(k-1)$ interventions), so the intervention budget grows with graph size. Task accuracy and `frequency-weight` F_1 separate the two models cleanly; root-node F_1 shows the smallest gap and both models still drop on 6–7 node graphs.

G.2 DeepSeek-V4 Family Results

Table 18 reports the `DeepSeek-V4` evaluation on the matched 6-node main suite referenced in RQ3, extending the model-family comparison to six configurations across three families. On the same graphs, `DeepSeek-V4-Pro` improves task success by 0.200 (95% CI $[0.060, 0.340]$) and `frequency`-weight F_1 by 0.169 ($[0.064, 0.282]$) over `DeepSeek-V4-Flash`. We evaluated this family on the 6-node suite given time constraints and will extend the evaluation to other graph sizes in future work. In Figure 4, which plots all six configurations, the Structure axis is $1 - \mathrm{SHD}/8.82$, where 8.82 is the mean true edge count of a 6-node graph (Table 4).

Both configurations were run and re-evaluated on 2026-07-14, before either model ID was repointed, so they are the 2026-04-24 preview builds that the model IDs `deepseek-v4-pro` and `deepseek-v4-flash` served at the time. They ran in thinking mode (`reasoning_effort=high`, 24,576 max output tokens) with no sampling temperature set. Those IDs were later repointed to newer releases, the re-post-trained `DeepSeek-V4-Flash-0731` (public beta, 2026-07-31) and the general-availability `DeepSeek-V4-Pro` (2026-08-13), both of which report substantially stronger agentic benchmark scores than the preview builds; the `deepseek-v4-flash` ID was routed to `DeepSeek-V4.1-Flash` again on 2026-09-10. The numbers below therefore characterize the preview builds and are not reproducible by calling those model IDs today.

Root-node F_1 is reported here for completeness but is not carried into the cross-family comparison in Figure 4, because it is not scored the same way for every family and for the GPT models it partly measures an answer convention. This family scores 0.780–0.887 on these graphs where the GPT and Qwen configurations score 0.354–0.473, and the spread has different causes. `GPT-5-mini` emits a mean of 4.58 root names and on 42 of the 50 graphs lists every non-target variable, answering which variables influence `frequency` rather than which of them are parentless. That pins its precision at 0.248 ($1/5$ on the 36 single-root graphs) while its recall reaches 0.910. Truncated to the size of the true set, its list comes close to `DeepSeek-V4-Pro` (exact on 40 against 43 of 50 graphs), and its first name is a true root on 44 of 50 graphs, the same as `Pro`. `GPT-5.2-high`'s 0.354 is also largely an elicitation effect: the roots implied by its own final graphs score 0.813. Against `Qwen3.5` the gap is not an artifact. The Qwen rows score the roots implied by each final graph (Table 17), and scored the same way `DeepSeek-V4-Pro` and `DeepSeek-V4-Flash` recover the true set exactly on 43 and 36 of 50 graphs (root F_1 0.910 and 0.807) against 15 and 13 for `Qwen3.5-Thinking` and `Qwen3.5-Non-thinking`, with sets of similar size (1.30 and 1.58 against 1.80 and 1.86 names).

Model	Task success	All-edge F_1	Root F_1	SHD↓	Freq-edge F_1	Freq-weight F_1
`DeepSeek-V4-Flash`	0.640	0.777	0.780	3.520	0.889	0.777
`DeepSeek-V4-Pro`	0.840	0.886	0.887	1.900	0.980	0.946

Table 18: `DeepSeek-V4` results on the 6-node main suite (50 graphs, same graphs as the other model families; default budget of 2 initial observations and 20 interventions).

H RQ4 Details: Early-Commitment Diagnostics

Figure 5 gives the three `GPT-5-mini` diagnostics behind the RQ4 early-commitment analysis in §5.5. Panel (a) compares the intervention budget, $4(k-1)$, with the mean number of interventions that successful and failed runs of the main 4- and 5-node suites use. Panel (b) takes each run's last recorded hypothesis H_t and reports the share of the run's own data $\mathcal{D}_{\leq t}$ whose recorded `frequency` the hypothesis equation reproduces within 0.01 Hz, averaged over runs (33 successful and 67 failed runs, pooled over 4 and 5 nodes). Of the 67 failed runs, 61 end with at least one data point that their final hypothesis contradicts.

The budget-scaling numbers in §5.5 come from Appendix F.1. The 6-node mixed-regime change of $+0.003$ all-edge F_1, from 5 to 20 interventions with 5 observations, is for `GPT-5.2-high` (Table 15); under the same change `GPT-5-mini` goes from 0.434 to 0.428 (Table 13). The 4-node pure-intervention drop from 0.754 to 0.596, from 3 to 12 interventions, is for `GPT-5-mini` (Table 12).

Panel (c) is a prompt ablation. It compares the `GPT-5-mini` 4-node main run (24/50 tasks solved) with a rerun on the same 50 graphs at the same budget (2 observations, 12 interven-

tions). The rerun adds three lines to the baseline run's Phase 1 controller prompt: on every step, the agent must check the `frequency` equation and coefficients of its current hypothesis against every entry of its collected data, state any inconsistency, and revise the hypothesis (`react_simple_memory_prompt_dsl_validate.txt` in the supplementary material). Nothing enforces the check: 17 of the rerun's 20 failed runs still end with a hypothesis that contradicts at least one of their own data points. The rerun solves 30/50 tasks. Paired by graph, 15 tasks flip to success and 9 to failure (exact McNemar $p \approx 0.31$), so the difference is not statistically significant; each arm is a single run.