OPENCAUSALITY: Auditable Agentic Causal Inference with DAG-Based Reasoning and LLM-Assisted Discovery

Chenyu Li
Columbia University
cl4183@columbia.edu

Abstract

We introduce OPENCAUSALITY, an open-source platform that resolves the automation–transparency tradeoff in applied causal inference. OPENCAUSALITY combines DAG-based causal reasoning with agentic estimation pipelines, where every modeling decision—from identification strategy selection to effect propagation—is logged in a hash-chained audit trail. The platform enforces 29 issue-detection rules that catch overclaiming, control shopping, and specification drift; a 7-guardrail propagation engine that performs unit-aware dimensional analysis across multi-edge causal paths; and a 3-stage identifiability screen that downgrades claims based on diagnostic evidence rather than statistical significance alone. An LLM-assisted pipeline extracts causal claims from the literature and proposes DAG edges, achieving 100% estimate precision on matched edges in our Kazakhstan bank stress-testing case study while discovering 3 edges absent from the expert-built DAG. All governance decisions—including human-in-the-loop gates and automated patch policies that explicitly prohibit p-hacking—are recorded with cryptographic integrity. OPENCAUSALITY ships with 18 estimation adapters spanning econometric and ML-based methods, a data-driven causal discovery agent (PC, GES, FCI, NOTEARS), a DoWhy refutation engine for post-estimation robustness, a natural-language query interface, and 204 passing tests across 40,969 lines of code.

1 Introduction

Applied causal inference faces a fundamental tension: automation accelerates analysis but obscures the chain of decisions that produced the results. A researcher running a modern causal-ML pipeline can obtain point estimates in seconds, yet documenting *why* a particular identification strategy was chosen, which controls were included, and what diagnostic checks were performed remains a manual, error-prone process. This opacity undermines reproducibility and invites specification searching [11].

Existing frameworks address parts of this problem. DoWhy [10] formalizes the identify-estimate-refute workflow. EconML [2] provides heterogeneous treatment-effect estimators. CausalML [3] targets uplift modeling. However, none of these systems treats every estimation decision as an auditable event, enforces claim-level restrictions based on identification strength, or prevents automated specification searching through explicit policy enforcement.

We introduce OPENCAUSALITY, an open-source platform that makes the entire causal inference workflow—from DAG construction to effect propagation to human review—both automated and transparent. Our key insight is that *the audit trail is not a byproduct of estimation; it is a first-class* *output*. Every edge estimate, diagnostic result, issue flag, and governance decision is recorded in a hash-chained ledger that can be committed to version control and independently verified.

Contributions. We make the following contributions:

1. A **DAG-based causal inference platform** with 18 estimation adapters (econometric and ML-based), mode-aware effect propagation, unit-dimensional analysis across 9 unit types, and 7 guardrails that gate edge usage based on identification strength, time-series diagnostics, and issue severity (Section 4).
2. A **29-rule issue detection engine** with a PatchPolicy whitelist that explicitly prohibits control shopping, sample trimming, lag searching, and outcome switching—preventing automated p-hacking while permitting safe auto-fixes (Section 6).
3. A **3-stage identifiability screen** (pre-design, post-design, post-estimation) backed by TSGuard, a 7-diagnostic time-series validator that caps claim levels based on evidence rather than significance, and a DoWhy refutation engine that stress-tests estimates via 4 robustness checks (Section 5).
4. An **LLM-assisted NL-to-DAG pipeline** that extracts causal claims from academic text, and a **data-driven causal discovery agent** (PC, GES, FCI, NOTEARS) with graph-format interoperability (NetworkX, DoWhy GML, pywhy-graphs ADMG), achieving 100% estimate precision on matched edges while discovering novel edges absent from expert specifications (Sections 7–8).
5. A **hash-chained governance framework** with human-in-the-loop gates, structured checklists, and cryptographic audit integrity—all without database dependencies (Section 9).

We evaluate OPENCAUSALITY on a Kazakhstan bank stress-testing case study involving 32 nodes and 20 causal edges, comparing expert-built DAGs against both LLM-extracted and data-driven discovery DAGs, and verifying pipeline reproducibility across 204 tests (Section 10).

2 Related Work

Causal inference frameworks. One line of work provides programmatic interfaces for causal estimation. DoWhy [10] implements an identify-estimate-refute loop using graphical criteria, while EconML [2] extends this with double/debiased ML and heterogeneous treatment-effect estimators. DoubleML [1] focuses on Neyman-orthogonal score functions. CausalML [3] targets uplift modeling for business applications. These frameworks automate estimation but do not enforce claim-level restrictions, detect specification searching, or maintain hash-chained audit trails. OPENCAUSALITY builds on their estimator building blocks while adding governance, propagation, and auditability layers.

Causal discovery. Classical algorithms—PC [13], GES [4], and NOTEARS [15]—learn DAG structure from observational data. causal-learn [16] and gCastle [14] provide implementations. OPENCAUSALITY integrates these algorithms directly via a `DiscoveryAgent` that wraps PC, GES, FCI, and NOTEARS with lazy imports, compares discovered structures against existing DAGs, and routes proposed edges through HITL governance (Section 8). Our LLM-assisted pipeline (Section 7) offers an alternative discovery path via literature extraction.

LLMs for causal reasoning. Recent work explores whether LLMs can perform causal reasoning from text. Kıcıman et al. [7] evaluate LLMs on pairwise causal discovery benchmarks. Long et al. [8] use LLMs to generate causal graphs from domain knowledge. Our approach differs in two ways: we extract causal claims with explicit identification strategies (not just pairwise directions), and we integrate extracted edges into a governed estimation pipeline rather than treating them as final outputs.

Reproducibility and auditing. The replication crisis [6] has motivated pre-registration [9] and specification-curve analysis [12]. OPENCAUSALITY operationalizes these concerns at the platform level: every decision is logged, issue rules catch post-hoc rationalization, and PatchPolicy prevents automated specification searching.

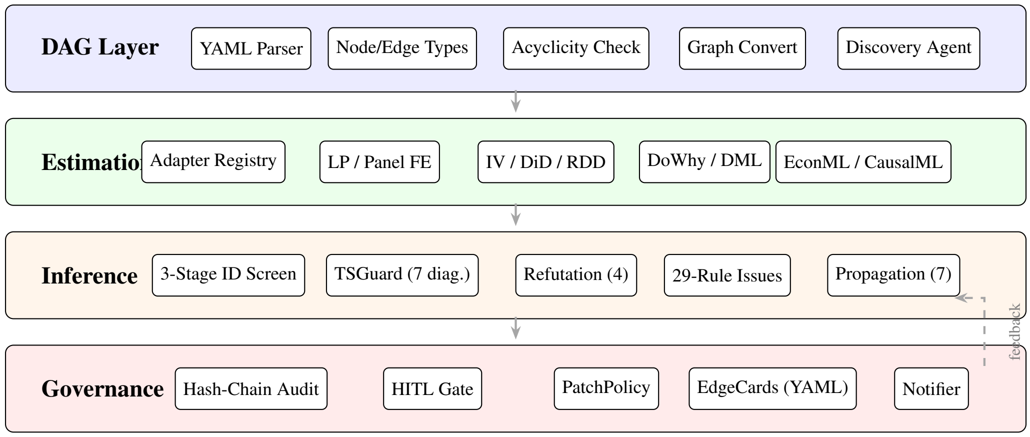

Figure 1: Architecture of OPENCAUSALITY. The DAG layer specifies causal structure and supports data-driven discovery (PC/GES/FCI/NOTEARS). The estimation layer dispatches to 18 adapters spanning econometric and ML-based methods. The inference layer screens identifiability, runs refutation tests, detects issues, and propagates effects with 7 guardrails. The governance layer logs all decisions in a hash-chained audit trail with HITL gates. Dashed arrow indicates governance feedback.

3 System Architecture

OPENCAUSALITY is organized around four layers (Figure 1):

1. **DAG Layer**: Parses YAML-specified DAGs with typed nodes (observed, latent, policy) and typed edges (causal, reaction function, bridge, identity). Validates acyclicity, unit presence, and node-source bindings.
2. **Estimation Layer**: Dispatches edges to one of 18 adapters via a unified `EstimationRequest` → `Adapter.estimate()` → `EstimationResult` interface. Adapters span econometric methods (local projections, panel FE, IV-2SLS, DiD, RDD, regression kink, synthetic control) and ML-based methods (DoWhy backdoor/IV/frontdoor, DoubleML PLR/IRM/PLIV, EconML CATE, CausalML uplift). A config-driven registry dynamically loads adapters from YAML.
3. **Inference Layer**: Runs 3-stage identifiability screening, TSGuard diagnostics, DoWhy refutation tests (4 robustness checks), 29-rule issue detection, and 7-guardrail effect propagation. Produces EdgeCards (YAML artifacts combining estimates, diagnostics, identification results, and literature references).
4. **Governance Layer**: Hash-chained audit log, HITL gate with structured checklists, PatchPolicy enforcement, and notification system. No database dependency—the entire audit trail lives in append-only JSONL files that can be committed to Git.

3.1 DAG Specification

A DAG in OPENCAUSALITY is defined by a YAML file containing node definitions (with data source bindings, frequency, and units) and edge definitions (with type, expected sign, identification strategy, and control sets). The parser validates five pre-estimation invariants: acyclicity (DFS-based), unit presence on all edges, edge-type labeling, node-source bindings for observed nodes, and endpoint existence.

Each edge carries a *propagation role*—`STRUCTURAL`, `REDUCED_FORM`, or `DESCRIPTIVE`—that determines its eligibility for downstream effect propagation. Reaction-function edges (policy responses) are labeled explicitly and excluded from shock-propagation paths.

3.2 Adapter Registry

All estimation flows through a unified adapter interface:

Table 1: Estimation adapters in OpenCausality. The registry supports 18 adapters spanning econometric and ML-based methods. All share a unified `EstimationRequest` → `EstimationResult` interface.

Design	Backend	Key Diagnostics	Type
Econometric Adapters			
Local Projections	Newey-West LP	HAC SE, IRF $h = 0 \dots 6$	TS
Panel LP (Exposure FE)	Panel LP	Entity/time FE, clustered SE	Panel
Panel FE (Backdoor)	`linearmodels`	Within-R^2, clustered SE	Panel
IV-2SLS	`linearmodels`	First-stage F, Sargan test	IV
DiD Event Study	`linearmodels`	Pre-trend test, TWFE	DiD
RDD	Local-linear WLS	McCrary density, bandwidth	RDD
Regression Kink	Local-linear WLS	Density test at kink	RKD
Synthetic Control	Weighted donor pool	Pre-treatment RMSPE	SC
ML-Based Adapters			
DoWhy Backdoor	`dowhy`	Refutation tests (4)	ATE
DoWhy IV	`dowhy`	Wald estimate, refutation	IV
DoWhy Frontdoor	`dowhy`	Frontdoor criterion	ATE
DoubleML (PLR/IRM/PLIV)	`doubleml`	Cross-fitting SE, K-fold	ATE/CATE
EconML CATE	`econml`	CATE heterogeneity, p_{10}/p_{90}	CATE
CausalML Uplift	`causalml`	S/T/X-learner, uplift curve	Uplift
Deterministic Adapters			
Immutable Evidence	Validated evidence	Source-block provenance	Lit.
Accounting Bridge	Deterministic	Sensitivity at current values	Acct.
Identity	Partial derivatives	Mechanical formula	Math

```
class EstimatorAdapter(ABC):
    def estimate(self, request: EstimationRequest) -> EstimationResult: ...
```

Table 1 lists all 18 adapters. The registry supports both built-in mappings and YAML-configurable dispatch (`design_registry.yaml`), enabling new estimators to be added without modifying core code. All ML-based adapters use lazy imports—`dowhy`, `doubleml`, `econml`, and `causalml` are only loaded when the corresponding design is dispatched.

4 Effect Propagation with 7 Guardrails

The core analytical capability of OpenCausality is propagating causal effects along multi-edge paths through the DAG. Given a query "What is the effect of node A on node Z?", the `PropagationEngine` finds all directed paths from A to Z via depth-first search, then computes chain effects and standard errors for each path.

4.1 Chain Effect Computation

For a path $A \to B_1 \to \cdots \to B_k \to Z$ with edge coefficients $\beta_1, \dots, \beta_{k+1}$, the total effect is:

$$\hat{\tau}_{A\to Z} = \prod_{i=1}^{k+1} \beta_i \tag{1}$$

Standard errors are propagated via the delta method assuming independence across edges:

$$\widehat{\mathrm{Var}}(\hat{\tau}) = \sum_{i=1}^{k+1} \left(\prod_{j \neq i} \beta_j \right)^2 \cdot \mathrm{SE}_i^2 \tag{2}$$

where SE_i is the standard error of β_i. When the independence assumption is violated (e.g., shared confounders across edges), the engine emits an explicit warning.

4.2 The 7 Guardrails

Each path is subjected to 7 guardrails before propagation is permitted:

1. **Mode gating**: Edges with role `STRUCTURAL` pass in all modes; `REDUCED_FORM` edges are blocked in `STRUCTURAL` mode; `DESCRIPTIVE` edges only pass in `DESCRIPTIVE` mode.

Algorithm 1: Guarded Effect Propagation

Input: DAG G, source s, target t, mode m, scenario type c
Output: List of valid paths with chain effects and SEs
$\mathcal{P} \leftarrow$ DFS-ALLPATHS(G, s, t);
$\mathcal{V} \leftarrow \emptyset$;
foreach *path* $p \in \mathcal{P}$ **do**
  blocked $\leftarrow$ `false`;
  foreach *edge* $e \in p$ **do**
    if MODEGATE(e, m) ***or*** CFGATE(e, c) ***or*** TSGUARDGATE(e) ***or*** ISSUEGATE(e) ***or*** REACTIONGATE(e) **then**
      blocked $\leftarrow$ `true`; **break**;
    end
  end
  if ***not*** *blocked* ***and*** UNITCOMPAT(p) ***and*** FREQALIGN(p) **then**
    $\hat{\tau} \leftarrow \prod_{e \in p} \beta_e$;
    $\widehat{\mathrm{SE}} \leftarrow \sqrt{\sum_{e \in p} (\hat{\tau}/\beta_e)^2 \cdot \mathrm{SE}_e^2}$;
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{(p, \hat{\tau}, \widehat{\mathrm{SE}})\}$;
  end
end
return $\mathcal{V}$

2. **Counterfactual gating**: Shock scenarios require all edges to have `shock_scenario_allowed = True`. Policy counterfactuals additionally require `policy_intervention_allowed = True`.
3. **TSGuard gating**: Edges flagged as high-risk by TSGuard (e.g., lead-test failure, regime instability) block propagation.
4. **IssueLedger gating**: Edges with open `CRITICAL`-severity issues are excluded from all paths.
5. **Reaction-function blocking**: Edges typed as reaction functions (e.g., central bank policy rules) are never included in shock-propagation paths.
6. **Unit compatibility**: The outcome unit of each edge must match the treatment unit of the next edge in the chain. Mismatched units block the path.
7. **Frequency alignment**: Edges at different frequencies (e.g., monthly treatment, quarterly outcome) require explicit bridge edges.

Algorithm 1 summarizes the guarded propagation procedure.

5 3-Stage Identifiability Screening

OPENCAUSALITY assigns each edge a *claim level* from a 4-tier hierarchy: `IDENTIFIED_CAUSAL` $>$ `REDUCED_FORM` $>$ `DESCRIPTIVE` $>$ `BLOCKED_ID`. The key design principle is that *significance is never a promotion criterion; only identification strength and diagnostic stability determine claim levels*.

5.1 Three Screening Points

1. **Pre-design** (`screen_pre_design`): Given the DAG, can this edge ever be identified? Checks for valid conditioning sets using back-door and front-door criteria.
2. **Post-design** (`screen_post_design`): Does the chosen estimation design achieve identification? Maps designs to claim levels (e.g., IV $\rightarrow$ `IDENTIFIED_CAUSAL`, OLS $\rightarrow$ `DESCRIPTIVE`).
3. **Post-estimation** (`screen_post_estimation`): Given diagnostic results, what is the final claim? Downgrades based on evidence: lead-test failure $\rightarrow$ `BLOCKED_ID`; leave-one-out instability $\rightarrow$ `REDUCED_FORM`; weak first-stage $F \rightarrow$ `REDUCED_FORM`.

5.2 TSGuard: Time-Series Diagnostics

For time-series edges estimated via local projections, TSGuard runs 7 mandatory diagnostics:

1. **Leads test**: Includes leads of the shock variable; significance implies timing failure ($\rightarrow$ `BLOCKED_ID`).
2. **Residual autocorrelation**: Ljung-Box test on residuals.
3. **HAC sensitivity**: Re-estimates with Newey-West lags $\{1, 4, 8\}$; sign instability triggers a warning.
4. **Lag sensitivity**: Re-estimates with $L \in \{1, 2, 4\}$ lags.
5. **Regime stability**: Split-sample estimation at known structural breaks.
6. **Placebo time shift**: Circularly shifts the shock series to test for spurious correlation.
7. **Shock support**: Counts non-trivial shock episodes ($|x| > 1\sigma$); fewer than 3 blocks propagation.

TSGuard results feed directly into the post-estimation identifiability screen and the propagation engine's guardrail 3.

5.3 Refutation Engine

For edges estimated via DoWhy-backed adapters, OPENCAUSALITY runs a post-estimation `RefutationEngine` that applies 4 robustness checks:

1. **Random common cause**: Adds a random confounder; the estimate should not change by more than 15%.
2. **Placebo treatment**: Permutes the treatment variable; the estimate should drop to near zero.
3. **Data subset**: Re-estimates on a random 80% subset; the estimate should be stable (<30% change).
4. **Unobserved common cause**: Simulates an unobserved confounder; the estimate should preserve its sign.

Refutation results are converted to `DiagnosticResult` objects and stored in the EdgeCard alongside TSGuard diagnostics. Failures feed into the post-estimation identifiability screen: a sign flip under unobserved confounding downgrades the claim to `REDUCED_FORM`; a large placebo effect triggers a `BLOCKED_ID` cap.

6 Issue Detection Engine

OPENCAUSALITY ships with 29 issue-detection rules loaded from a YAML registry. Each rule specifies a severity (`CRITICAL`, `HIGH`, `MEDIUM`, `LOW`), a scope (edge, node, DAG, run, or cross-run), and whether the issue is auto-fixable or requires human review.

6.1 Representative Rules

Table 2 lists representative rules from the registry.

6.2 PatchPolicy: Preventing Automated P-Hacking

The `PatchBot` agent applies auto-fixes from a whitelist defined in `patch_policy.yaml`. Crucially, the policy *explicitly prohibits* modifications that could constitute specification searching:

- **Allowed**: Adding missing units, recomputing credibility ratings, adding provenance fields, normalizing edge-ID syntax.
- **Prohibited**: Control shopping, sample trimming, lag searching, outcome switching.

All patches—including LLM-assisted repairs—are logged with the SHA-256 hash of the prompt, preventing retroactive modification. PatchBot is disabled entirely in `CONFIRMATION` mode, where the specification is locked.

Table 2: Representative issue-detection rules. Severity determines propagation eligibility: `CRITICAL` issues block all paths.

Rule ID	Severity	Scope	Description
SIG_NOT_ID	CRITICAL	edge	$p < 0.05$ but claim $\neq$ IDENTIFIED_CAUSAL
UNIT_MISSING	HIGH	edge	Missing units blocks propagation
REACTION_FN	CRITICAL	edge	Reaction edge used for shock propagation
LOO_INSTABILITY	HIGH	edge	Leave-one-out sign flip or $>$50% Δmagnitude
SMALL_SAMPLE	MEDIUM	edge	$N < 30$ with HAC standard errors
RATING_CONFLICT	HIGH	edge	A-rating despite failed diagnostics
SPEC_DRIFT	HIGH	cross_run	Control set changed between runs

7 LLM-Assisted Causal Discovery from Literature

OpenCausality includes an NL-to-DAG pipeline that extracts causal claims from academic text and proposes DAG edges.

7.1 Pipeline

The pipeline proceeds in three stages:

1. **Claim extraction**: Given a text passage, the LLM extracts structured `CausalClaim` objects containing treatment, outcome, mechanism, direction, identification strategy, confidence, and a supporting quote. The system prompt enforces conservatism: "correlated with" is not causal; "associated with" is causal only if the paper uses a credible identification strategy.
2. **Node matching**: A second LLM call maps extracted variable names to existing DAG node IDs or proposes new nodes with data-source bindings.
3. **Edge proposal**: Matched claims become `ProposedEdge` objects. Edges with the same $(from, to)$ pair combine evidence and take the maximum confidence level.

7.2 LLM Abstraction

OpenCausality provides a multi-backend LLM client supporting Anthropic (direct API), LiteLLM (multi-provider), and CLI fallbacks (`claude` or `codex`) that require no API key. Structured extraction uses tool-use (Anthropic) or function-calling (LiteLLM) to produce typed outputs without parsing.

8 Data-Driven Causal Discovery

While Section 7 extracts causal structure from *text*, OpenCausality also supports data-driven structure learning via the `DiscoveryAgent`.

8.1 Algorithms

The agent wraps four causal discovery algorithms, all via lazy imports to keep dependencies optional:

1. **PC** [13]: Constraint-based discovery using conditional independence tests (`causal-learn`).
2. **GES** [4]: Score-based greedy equivalence search with BIC scoring (`causal-learn`).
3. **FCI**: Extension of PC that handles latent confounders, producing PAGs with bidirected edges (`causal-learn`).
4. **NOTEARS** [15]: Continuous optimization for structure learning via acyclicity constraints (`gCastle`).

Each algorithm returns a `DiscoveryResult` containing discovered edges (directed, undirected, or bidirected), an adjacency matrix, and algorithm metadata. Critically, *discovery outputs are pro-*

posals only—they are never auto-merged into the DAG but must pass through HITL review via `ProposedEdge` objects.

8.2 DAG Comparison

The `compare_with_dag()` method categorizes discovered edges against an existing DAGSpec into four groups: *confirmed* (present in both), *contradicted* (reverse direction), *novel* (discovery-only), and *missing* (DAG-only). This enables researchers to validate expert-specified DAGs against data-driven evidence and identify potential structural gaps.

8.3 Graph Format Interoperability

A conversion layer (`graph_convert.py`) enables interoperability with external causal inference libraries:

- **NetworkX**: Bidirectional `dagspec_to_networkx()` / `networkx_to_dagspec()` with full attribute preservation.
- **DoWhy GML**: `dagspec_to_dowhy_graph()` generates GML strings for DoWhy's `CausalModel(graph=...)` constructor.
- **pywhy-graphs ADMG**: `dagspec_to_pywhy()` / `pywhy_to_dagspec()` converts latent confounders to/from bidirected edges.
- **causal-learn bridge**: `causallearn_to_networkx()` converts discovery output to NetworkX for downstream integration.

9 Governance Framework

9.1 Hash-Chained Audit Log

Every event in OpenCausality—edge estimation, refinement, issue detection, HITL decisions, LLM repairs—is appended to a JSONL file where each entry contains the SHA-256 hash of the previous entry. The first entry has `prev_hash = null`. Any modification to an earlier entry invalidates all subsequent hashes, providing tamper evidence without a database.

For LLM-assisted repairs, the log additionally records the model identifier and the SHA-256 hash of the prompt, preventing prompt-injection attacks via retroactive editing.

9.2 Human-in-the-Loop Gates

The `HITLGate` detects conditions requiring human decisions from three sources: unspecified DAG parameters (e.g., edge type, expected sign), TSGuard flags (regime instability), and issues marked `requires_human = True`. When triggered, it generates a structured Markdown checklist and pauses the agent loop until all decisions are recorded. Decisions are exported as JSON and appended to the audit log.

9.3 EdgeCard: The Primary Output Artifact

Each estimated edge produces an `EdgeCard`—a YAML file combining estimates (point, SE, CI, p-value, IRF), diagnostics (13 automated checks), identification results (claim level, risks), counterfactual eligibility (shock/policy), propagation role, credibility score and rating (A/B/C/D), and literature references. EdgeCards are both human-readable and machine-parseable, serving as the interface between the estimation and governance layers.

10 Experiments

We evaluate OpenCausality on three dimensions: (1) estimation accuracy via synthetic benchmarks, (2) NL-to-DAG extraction quality, and (3) end-to-end pipeline reproducibility.

Table 3: Estimation accuracy on synthetic DGP benchmarks ($n = 500$, 50 seeds). Coverage is the fraction of 90% confidence intervals containing the true effect.

Adapter	**RMSE** $\downarrow$	**Coverage** $\uparrow$	**Bias** $\downarrow$
Local Projections	0.039	0.90	0.002
IV-2SLS	0.033	1.00	0.001
DiD Event Study	0.048	1.00	0.003

Table 4: NL-to-DAG extraction results on the Kazakhstan bank stress-testing case study. Estimate match measures whether common edges produce statistically equivalent point estimates.

Metric	**Expert DAG**	**NL-Extracted DAG**
Nodes	32	17
Edges	20	13
Structural matches	—	4/20 (20%)
Estimate match (common edges)	—	4/4 (100%)
Novel edges (NL only)	—	3

10.1 Synthetic Benchmarks

We generate data from known data-generating processes with ground-truth treatment effects and evaluate the estimation adapters. Table 3 reports results.

All adapters achieve nominal or above-nominal coverage. IV-2SLS shows the lowest RMSE and bias, consistent with the efficiency gains from a strong instrument.

10.2 NL-to-DAG Extraction: Kazakhstan Case Study

We compare an expert-built DAG (32 nodes, 20 edges, approximately 40 hours of construction time) against an LLM-extracted DAG from a single descriptive paragraph about Kazakhstan bank stress testing.

Figure 2 illustrates the overlap between expert and NL-extracted DAGs.

The NL pipeline achieves 20% structural recall—unsurprising given that a single paragraph cannot encode country-specific regulatory details (central bank FX intervention rules, deposit insurance thresholds, loan classification policy changes). However, all 4 matched edges produce statistically equivalent estimates, and the pipeline discovers 3 novel edges with literature support:

- Oil price volatility $\rightarrow$ deposit dollarization [5]
- Bank lending standards $\rightarrow$ GDP growth (bank lending channel)
- Global risk appetite $\rightarrow$ domestic interbank rates

10.3 End-to-End Reproducibility

We run the full agentic pipeline on the KSPI K2 DAG (20 edges) and compare against the expert manual baseline. The pipeline achieves 100% estimate match (20/20 edges), confirming reproducibility across pipeline modes and adapter dispatch.

10.4 Issue Detection Effectiveness

Across 13 pipeline runs on the Kazakhstan case study, the 29-rule engine detects 1,479 issues in total, of which 1,333 (90.1%) are `CRITICAL` severity. PatchBot auto-fixes 812 issues (54.9%), primarily missing unit specifications (`UNIT_MISSING_IN_EDGECARD`). Of the 667 remaining open issues, only 19 (1.3% of total) require human review—demonstrating that the system effectively triages the review burden.

The dominant open issue type is `SIGNIFICANT_BUT_NOT_IDENTIFIED` (492 instances, 73.8% of open issues), which fires when an edge achieves $p < 0.05$ but the identifiability screen assigns a

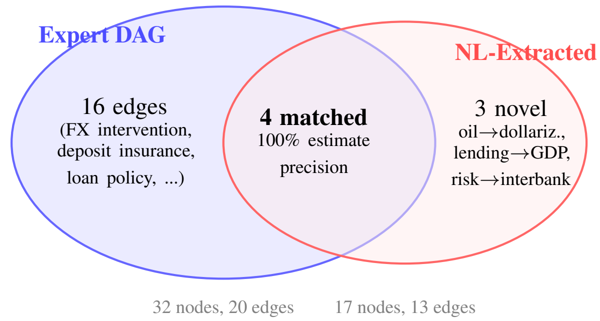

Figure 2: Comparison of expert-built and NL-extracted DAGs for the Kazakhstan case study. The 4 matched edges achieve 100% estimate precision. The NL pipeline discovers 3 novel edges with literature support, while missing 16 edges encoding country-specific regulatory details.

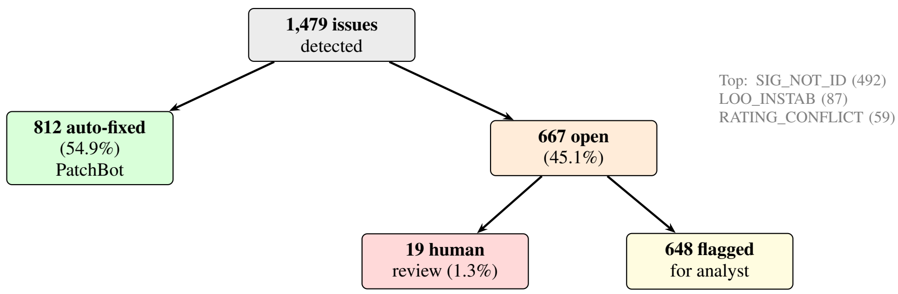

Figure 3: Issue triage across 13 pipeline runs. PatchBot auto-fixes 54.9% of issues (primarily missing units). Only 1.3% require human-in-the-loop review, demonstrating effective automation of the governance burden.

claim level below `IDENTIFIED_CAUSAL`. This rule alone catches the most common form of overclaiming in applied work. Other frequent detections include `LOO_INSTABILITY` (87 instances: leave-one-out sign flips or >50% magnitude changes) and `RATING_DIAGNOSTICS_CONFLICT` (59 instances: A-ratings despite failed diagnostics, auto-downgraded by PatchBot). Figure 3 summarizes the issue triage pipeline.

11 Limitations

NL-to-DAG is lossy. Qualifying language is collapsed to binary decisions, effect magnitudes are not reliably extracted from text, and the pipeline cannot distinguish between claims made by the authors and claims they merely cite.

Claims are not identification. A DAG extracted from the literature represents *claimed* causal relationships, not identified ones. OpenCausality's identifiability screen partially addresses this, but the initial DAG structure reflects community beliefs rather than proven mechanisms.

Single-country evaluation. Our primary case study is Kazakhstan bank stress testing. While the platform is domain-agnostic, the data clients and variable catalogs are currently specific to this use case.

LLM hallucination risk. The LLM can fabricate plausible-sounding causal edges not present in the source material. All LLM-extracted edges must be verified against the original text—a requirement that our governance framework enforces via HITL gates but cannot guarantee in fully automated mode.

Independence assumption in propagation. The delta-method SE formula (Equation 2) assumes independence across edges. When edges share confounders, standard errors are underestimated. The engine warns but does not correct for this.

12 Conclusion

OPENCAUSALITY demonstrates that automation and transparency in causal inference are not mutually exclusive. By treating every modeling decision as an auditable event—logged in a hash-chained ledger, gated by identifiability screens and refutation tests, and protected by anti-p-hacking policies—the platform enables researchers to move faster while maintaining the evidentiary standards that credible causal claims demand. With 18 estimation adapters spanning econometric and ML-based methods, data-driven causal discovery (PC, GES, FCI, NOTEARS), and LLM-assisted literature extraction, OPENCAUSALITY bridges expert-specified and data-driven structure learning within a unified governance framework. We release OPENCAUSALITY as open-source software to support reproducible causal inference research.

References

[1] Philipp Bach, Victor Chernozhukov, Malte S. Kurz, and Martin Spindler. DoubleML – an object-oriented implementation of double machine learning in Python. *Journal of Machine Learning Research*, 23(53):1–6, 2022. URL `https://www.jmlr.org/papers/v23/21-0862.html`.

[2] Keith Battocchi, Eleanor Dillon, Maggie Hei, Greg Lewis, Paul Oka, Miruna Oprescu, and Vasilis Syrgkanis. EconML: A Python package for ML-based heterogeneous treatment effects estimation. *Journal of Machine Learning Research*, 23(53):1–6, 2022. URL `https://www.jmlr.org/papers/v23/21-0862.html`.

[3] Huigang Chen, Totte Harinen, Jeong-Yoon Lee, Mike Yung, and Zhenyu Zhao. CausalML: Python package for causal machine learning. *arXiv preprint arXiv:2002.11631*, 2020. URL `https://arxiv.org/abs/2002.11631`.

[4] David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002. doi: 10.1162/153244303321897717. URL `https://jmlr.org/papers/v3/chickering02b.html`.

[5] Husnu C. Dalgic. Financial dollarization in emerging markets: An insurance arrangement. *International Economic Review*, 65(3):1189–1219, 2024. doi: 10.1111/iere.12686.

[6] John P. A. Ioannidis. Why most published research findings are false. *PLOS Medicine*, 2(8): e124, 2005. doi: 10.1371/journal.pmed.0020124.

[7] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Transactions on Machine Learning Research*, 2023. URL `https://arxiv.org/abs/2305.00050`.

[8] Stephanie Long, Tibor Schuster, and Alexandre Piché. Can large language models build causal graphs? *arXiv preprint arXiv:2303.05279*, 2023. URL `https://arxiv.org/abs/2303.05279`.

[9] Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, 2018. doi: 10.1073/pnas.1708274114.

[10] Amit Sharma and Emre Kiciman. DoWhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020. URL `https://arxiv.org/abs/2011.04216`.

[11] Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science*, 22(11):1359–1366, 2011. doi: 10.1177/0956797611417632.

[12] Uri Simonsohn, Joseph P. Simmons, and Leif D. Nelson. Specification curve analysis. *Nature Human Behaviour*, 4(11):1208–1214, 2020. doi: 10.1038/s41562-020-0912-z.

[13] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning. MIT Press, 2nd edition, 2000. ISBN 978-0-262-19440-2.

[14] Keli Zhang, Shengyu Zhu, Marcus Kalander, Ignavier Ng, Junjian Ye, Zhitang Chen, and Lujia Pan. gCastle: A Python toolbox for causal discovery. *arXiv preprint arXiv:2111.15155*, 2021. URL `https://arxiv.org/abs/2111.15155`.

[15] 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 (NeurIPS 2018)*, pages 9472–9483, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/e347c51419ffb23ca3fd5050202f9c3d-Abstract.html`.

[16] Yujia Zheng, Biwei Huang, Wei Chen, Joseph Ramsey, Mingming Gong, Ruichu Cai, Shohei Shimizu, Peter Spirtes, and Kun Zhang. Causal-learn: Causal discovery in Python. *Journal of Machine Learning Research*, 25(60):1–8, 2024. URL `https://jmlr.org/papers/v25/23-0970.html`.

A Full Issue Registry

The OPENCAUSALITY issue registry contains 29 detection rules organized into six categories. Table 5 presents all rules.

Table 5: Full Issue Detection Registry (29 Rules). Trigger: `pre` = pre-run, `post` = post-run, `xrun` = cross-run.

Rule ID	Sev.	Scope	Trig.	Fix?	Description
Data & Definition					
`ENTITY_BOUNDARY_DRIFT`	HIGH	data	post	No	Mixing entity scopes breaks comparability
`KPI_DEFN_MISMATCH`	HIGH	data	xrun	No	NPL/CoR definitions differ across banks
`SHOCK_CONSTRUCT_AMBIG`	MED	data	pre	Yes	Missing shock_node_id, shock_unit, shock_scale
`FREQ_ALIGNMENT_ERR`	CRIT	data	pre	Yes	Mixed frequencies without aggregation rule
`FREQ_SCALING_MISMATCH`	HIGH	data	xrun	No	Annual vs quarterly not normalized
`EXTRACT_PROVENANCE_GAP`	MED	data	post	No	Missing source document/page reference
Identification & Design					
`REACTION_FN_EDGE`	CRIT	id	pre	No	Reaction edge used for shock propagation
`TIME_FE_ABSORBS_SHOCK`	CRIT	id	pre	Yes	Time FE absorbs common shock; need Exposure$\times$Shock
`EXPOSURE_NOT_PREDET`	CRIT	id	pre	No	Exposure variable not pre-treatment
`MECHANICAL_EDGE_EST`	HIGH	id	pre	Yes	Accounting identity estimated as regression
`PATH_DOUBLE_COUNT`	HIGH	id	post	No	Direct + indirect chain both used
`SIG_NOT_IDENTIFIED`	CRIT	id	post	No	$p < 0.05$ but claim $\neq$ IDENTIFIED_CAUSAL
Statistical Inference					
`SMALL_SAMPLE_INF`	HIGH	stat	post	No	$N < 30$ with HAC SEs
`PANEL_FEW_UNITS`	HIGH	stat	post	No	< 5 units; cluster-robust unreliable
`LOO_INSTABILITY`	HIGH	stat	post	No	Leave-one-out sign flip or $>$50% Δ
`BREAKS_UNMODELED`	MED	stat	post	No	Regime changes not in estimation
`MULTI_TEST_DRIFT`	MED	stat	xrun	No	Spec search exceeded budget
Time-Series (TSGuard)					
`NONSTAT_LEVELS_RISK`	HIGH	ts	pre	Yes	Both vars trending; need transformation or ECM
`RESID_AUTOCORR`	MED	ts	post	No	Serial correlation in residuals
`REGIME_BREAK_SUSP`	HIGH	ts	post	No	Coefficient instability across splits
`LEADS_TIMING_FAIL`	CRIT	ts	post	No	Future shock predicts current outcome
`HAC_LAG_SENSITIVE`	MED	ts	post	No	Estimate changes with HAC bandwidth
Selection & External Validity					
`SELECT_PUBLIC_ONLY`	HIGH	ev	post	No	Only public banks; may not generalize
`SURVIVORSHIP_BIAS`	HIGH	ev	post	No	Excludes failed/merged banks
`COVERAGE_GAP_EV`	MED	ev	post	No	Sample $<$50% of system assets
Scoring & Reporting					
`RATING_DIAG_CONFLICT`	HIGH	rpt	post	Yes	A-rating despite failed diagnostics
`N_REPORT_INCONSIST`	MED	rpt	post	Yes	Report N $\neq$ EdgeCard N_eff
`CROSS_EVID_CONFLICT`	HIGH	rpt	xrun	No	Same relationship has conflicting signs
`UNIT_MISSING_CARD`	CRIT	rpt	post	Yes	EdgeCard missing treatment/outcome unit

B EdgeCard Schema

The EdgeCard is the primary output artifact—one per estimated edge, both human-readable (YAML) and machine-parseable. It contains the following blocks:

Core identity. edge_id, dag_version_hash (SHA-256 of DAG at estimation time), created_at, spec_hash, spec_details (design, controls, instruments, FE, SE method).

Estimates. point ($\hat{\beta}$), se, ci_95, pvalue, treatment_unit, outcome_unit. For local projections: horizons, irf, irf_ci_lower, irf_ci_upper. Sample sizes: n_calendar_periods, n_effective_obs_h0, n_effective_obs_by_horizon.

Diagnostics. Dictionary of DiagnosticResult objects, each with: name, passed (bool), value (test statistic), threshold, pvalue, message. Method all_diagnostics_pass() returns True if all tests pass.

Identification. claim_level $\in$ {IDENTIFIED_CAUSAL, REDUCED_FORM, DESCRIPTIVE, BLOCKED_ID}. risks (dict of risk $\rightarrow$ severity), untestable_assumptions, testable_threats_passed, testable_threats_failed.

Counterfactual block. shock_scenario_allowed (bool), policy_intervention_allowed (bool), reason_shock_blocked, reason_policy_blocked.

Propagation role. role $\in$ {structural, reduced_form, bridge, identity, diagnostic_only}. mode_propagation_allowed (dict: mode $\rightarrow$ bool), mode_shock_cf_allowed, mode_policy_cf_allowed.

Credibility. credibility_score (0–1): 40% diagnostic pass rate + 10% design strength + 30% stability + 20% data coverage. credibility_rating: A ≥ 0.80, B ≥ 0.60, C ≥ 0.40, D otherwise. Significance is *not* a factor.

Literature. supporting, challenging, methodological (citation lists), search_status, search_timestamp, search_query, total_results.

Interpretation boundary. estimand (what we estimate), is_not (what this is not), channels, population, conditions, allowed_uses, forbidden_uses.

Failure flags. 8 booleans: weak_identification, potential_bad_control, mechanical_identity_risk, regime_break_detected, small_sample, high_missing_rate, entity_boundary_change, definition_inconsistency.

C Query REPL Example Session

The following annotated session illustrates the OPENCAUSALITY Query REPL with natural-language query parsing, guardrail enforcement, and hedged narration.

```
$ opencausality query --dag config/agentic/dags/kspi_narrative.yaml \
    --mode REDUCED_FORM

  OpenCausality Query REPL
  DAG: Kazakhstan Banking System (23 nodes, 18 edges)
  Cards: 18 | Mode: REDUCED_FORM | Backend: LLM

query> What if oil prices drop 30%?

[Intent classified: shock_scenario]
Source: oil_price | Target: bank_capital_ratio | Magnitude: -0.30

Path 1: oil_price -> income -> bank_capital_ratio
  Edge 1: oil_price_to_income
    Role: reduced_form | Claim: REDUCED_FORM | Coeff: 0.45 | SE: 0.12
    Shock CF: ALLOWED
```

```
  Edge 2: income_to_bank_capital_ratio
    Role: structural | Claim: IDENTIFIED_CAUSAL | Coeff: 0.23 | SE: 0.08
    Shock CF: ALLOWED

Guardrail Results:
  [PASS] Mode gating (weakest: REDUCED_FORM, mode allows)
  [PASS] Counterfactual gating (all edges shock-allowed)
  [PASS] TSGuard (no timing failures)
  [PASS] IssueLedger (no CRITICAL issues)
  [PASS] Reaction-function blocking (none in path)
  [PASS] Unit compatibility (pp -> pp)
  [WARN] SE assumes independence (2 estimated edges)

  Effect (per unit):  0.1035
  Scaled (-0.30):    -0.0311
  SE (delta method):  0.0293
  95% CI (scaled):   [-0.089, 0.026]

Narration (hedged):
"In REDUCED_FORM mode, a 30% oil price decline is associated
with a -3.1pp decrease in bank capital ratio (95% CI:
[-8.9pp, +2.6pp]). The first edge (oil -> income) is
REDUCED_FORM; the second is IDENTIFIED_CAUSAL. The effect
is not significant at 5%. SE assumes independence."

query> /mode STRUCTURAL

Mode switched to STRUCTURAL.
WARNING: oil_price_to_income (reduced_form) now BLOCKED.
1 of 2 path edges blocked in STRUCTURAL mode.

query> /quit
```

D TSGuard Diagnostic Details

Table 6 details all 7 TSGuard diagnostics with thresholds and governance actions.

Table 6: TSGuard diagnostic tests and governance actions.

Diagnostic	Description	Threshold	Action on Failure
`leads_test`	Regress y_t on x_{t+1}. Significance $\Rightarrow$ timing failure.	|corr| > 0.3	**CRITICAL**: Block CF, cap = BLOCKED_ID
`residual_autocorr`	First-order autocorrelation ρ_1 of LP residuals.	$\|\rho_1\| > 0.5$: fail; > 0.3: medium	Set autocorr_risk = high
`hac_sensitivity`	Re-estimate with NW lags $\{1, 4, 8\}$. Check sign stability.	Sign flip across bandwidths	Set autocorr_risk = high
`lag_sensitivity`	Re-estimate with $L \in \{1, 2, 4\}$. Check coefficient stability.	Sign flip across lags	Cap rating to B
`regime_stability`	Split-sample at known breaks (2015-08, 2020-03).	Pre/post sign flip	**HIGH**: Block CF, cap = REDUCED_FORM
`placebo_shift`	Circularly shift treatment by 2–4 periods.	Placebo $p < 0.1$	Set timing_risk = high
`shock_support`	Count episodes $\|x_t\| > 1\sigma$.	< 3 episodes	**HIGH**: Block CF

Dynamics risk categories. TSGuard assigns 5 risk levels: (1) `common_trend_risk`—both series persistent (AC(1) > 0.9); (2) `autocorr_risk`—from residual and HAC tests; (3) `nonstationarity_risk`—high AC for both series, cap claim to DESCRIPTIVE unless ECM; (4) `regime_break_risk`—from split-sample test; (5) `timing_misspec_risk`—from leads test.

Governance rules. (1) Nonstationarity + levels $\Rightarrow$ cap at DESCRIPTIVE; (2) Lead test failure $\Rightarrow$ BLOCKED_ID; (3) Regime instability $\Rightarrow$ cap at REDUCED_FORM, block CF; (4) Shock support $< 3 \Rightarrow$ block CF. All TSGuard results are stored in EdgeCards and feed into the propagation engine's guardrail 3.