ScrubID: Identifiability-aware auditing for mechanistic interpretability

Ali Uyar
Independent Researcher

January 2026

Abstract

Mechanistic interpretability often claims that a specific circuit explains a model behavior, but multiple distinct circuits can reproduce the same behavior under common intervention protocols. We introduce ScrubID, an identifiability-aware auditing framework that treats circuit explanations as estimators whose uniqueness depends on the intervention family. ScrubID defines scrubbed models that preserve a candidate circuit while resampling the rest of the network under a specified intervention family, and evaluates ϵ-faithfulness relative to a task-specific tolerance. To detect non-identifiability in practice, ScrubID introduces three diagnostics computed from discovered candidate sets: RR, a redundancy ratio measuring worst-case disagreement among near-optimal faithful circuits; SSS, a stability score measuring how consistent recovered circuits are across replicate discovery runs; and CC, a contradiction score measuring how inconsistent component necessity is across near-optimal faithful circuits. We validate these diagnostics on a deterministic synthetic suite with ground-truth equivalence class sizes and planted redundancy, and we apply ScrubID to real transformer case studies on indirect object identification, year-span greater-than comparison (yes/no), and induction-style next-token prediction. Our results motivate reporting identifiability diagnostics alongside mechanistic circuit claims and provide standardized artifacts for reproducible audits.

On the synthetic suite, RR and CC jump from 0 at planted redundancy factor 1 to RR $\approx$ 0.67 and CC $\approx$ 0.67–0.82 for redundancy factors $\geq$ 2, and audit certificates are emitted at rate 1.0 in these redundant regimes (Table T1). In GPT-2 case studies under activation patching, across N=5 suite seeds we compare three candidate generators (`G_ATTR_PATCH`, `G_MANUAL_SEED`, `G_RANDOM_K`). On the ID split, `G_ATTR_PATCH` recovers smaller ϵ-faithful circuits for IOI (mean 124.0/156) and induction (mean 92.8/156); IOI exhibits moderate redundancy/contradiction signals (RR_mean $\approx$ 0.36, CC_mean $\approx$ 0.10) while induction shows both redundancy and discovery instability (RR_mean $\approx$ 0.59, SSS_mean $\approx$ 0.41), producing WARN/FAIL verdicts (Table T2). On the OOD split, induction becomes highly unstable (SSS_mean $\approx$ 0.13) and exhibits RR non-identifiability in some seeds (e.g. RR=0.93 with |S_near|=2), producing audit certificates (Table T4).

0.1 Introduction and novelty

Mechanistic interpretability aims to explain neural model behavior in terms of internal components such as attention heads, MLPs, or learned features. A common workflow is to localize a circuit that is faithful to a behavior and then interpret the circuit. This approach has produced detailed case studies in transformer language models, including indirect object identification [Wang et al., 2023], a greater-than style mathematical behavior [Hanna et al., 2023], and induction-like in-context behavior [Olsson et al., 2022].

A critical gap remains: even if one faithful circuit can be found, it is often unclear whether the explanation is unique, stable to implementation choices, or artificially inflated in complexity. Recent work explicitly frames this issue as identifiability. Méloux et al. show that mechanistic explanations can be systematically non-identifiable even in small networks, with multiple circuits and multiple interpretations satisfying common criteria [Méloux et al., 2025].

ScrubID addresses the practical version of this problem for transformer circuit analyses. The key idea is to audit explanations rather than assuming uniqueness. ScrubID constructs a scrubbed model that preserves a proposed circuit while resampling the rest of the network under a specified intervention family. It then quantifies three aspects that are typically underreported:

- **Redundancy (RR):** how much near-optimal faithful circuits disagree in their component sets.
- **Stability (SSS):** how consistent the recovered circuit is across replicate discovery runs (rerunning the same discovery procedure with deterministically derived replicate seeds).
- **Contradiction (CC):** how inconsistent component necessity is across near-optimal faithful circuits.

Separately, ScrubID reports the complexity proxy MDL(C) for each candidate circuit and highlights the minimum-MDL faithful circuit.

ScrubID produces an audit certificate when at least one diagnostic verdict is FAIL; the certificate records which diagnostic(s) triggered emission.

0.1.1 Nearest-neighbor delta vs Méloux et al. (ICLR 2025)

Nearest neighbor: Méloux et al. [Méloux et al., 2025].

1. Méloux et al. primarily study identifiability by enumerating explanations in Boolean functions and small MLP settings, demonstrating that non-identifiability can arise at multiple stages.
2. ScrubID targets transformer circuit practice and defines an explicit audit pipeline whose interface supports multiple intervention families (activation patching, path patching, causal scrubbing); the numeric results in this paper use activation patching only, and we exclude non-activation-patching IDs from main claims (see the Method note on intervention IDs).
3. ScrubID introduces three quantitative diagnostics (RR/SSS/CC) designed to be computed from discovered candidate sets, instead of requiring enumeration over all explanations.
4. ScrubID defines a certificate artifact that records multiple faithful circuits, their MDL complexity, and their behavior distances under a chosen intervention family.
5. ScrubID includes a deterministic synthetic suite that labels ground-truth equivalence class size and planted redundancy, enabling direct benchmarking of diagnostics.
6. ScrubID evaluates identifiability properties in real transformer case studies and reports stability across replicate discovery runs, a practical concern not addressed by exhaustive enumeration in small networks.
7. ScrubID treats discovery instability and intervention-family sensitivity as first-class axes, motivated by patching pitfalls such as subspace intervention illusions [Makelov et al., 2024].
8. ScrubID is designed to integrate with automated and feature-level circuit discovery methods [Conmy et al., 2023, Syed et al., 2023, Kramár et al., 2024, Marks et al., 2025].

The result is a determinism-first auditing protocol that complements identifiability theory by providing an operational audit for transformer interpretability studies.

0.2 Related work

We position ScrubID at the intersection of mechanistic interpretability practice and causal identifiability.

0.2.1 Mechanistic interpretability and circuit discovery

Activation patching and its variants are widely used to localize behavior to internal components, but different intervention choices can yield different explanations. Recent methodological work has clarified best practices and pitfalls for activation patching and related procedures [Heimersheim and Nanda, 2024, Zhang and Nanda, 2024, Makelov et al., 2024].

Automated circuit discovery methods such as ACDC and feature-based circuit extraction have improved scalability, yet still leave open the question of when an explanation is uniquely supported by the intervention family [Conmy et al., 2023, Marks et al., 2025].

Path patching offers a more fine-grained localization primitive that can be viewed as edge-aware scrubbing [Goldowsky-Dill et al., 2023].

0.2.2 Causal identifiability and causal abstraction

From a causal perspective, a mechanistic explanation is only meaningful if it is identifiable under the interventions available. In causal discovery, interventional Markov equivalence classes formalize when multiple causal graphs are indistinguishable [Hauser and Bühlmann, 2012, Eberhardt, 2012].

In mechanistic interpretability, causal abstraction provides a formal foundation for reasoning about interventions and explanations, and motivates certifying when multiple explanations are equally compatible with the same intervention family [Geiger et al., 2025, 2021].

The identifiability framing for mechanistic interpretability has recently been made explicit, including impossibility results under restricted interventions [Méloux et al., 2025].

0.2.3 Benchmarks and robustness across scale

Causal interpretability benchmarks aim to evaluate intervention methods under controlled task settings [Arora et al., 2024]. Empirically, some circuit analyses appear consistent across training and scale, but this can depend on methodology and task choice [Tigges et al., 2024, Lieberum et al., 2023].

0.2.4 ScrubID's contribution

ScrubID introduces a compact, implementation-ready protocol to:

- quantify worst-case redundancy among near-optimal circuits (RR),
- quantify solution stability across replicate discovery runs (SSS), and
- quantify contradictory necessity claims (CC),

and to emit an audit certificate when one or more diagnostics indicate the explanation is not reliably supported (e.g., non-identifiability or discovery instability).

0.3 Method

This section defines the ScrubID audit protocol end-to-end, including all numeric constants used in this paper. The intent is that a reader can reimplement the protocol from the paper alone.

0.3.1 Core objects

We fix:

- A model `f` (an autoregressive Transformer language model).

- A suite-defined dataset $D = (x_0, x_1, \ldots, x_{N-1})$ with deterministic ordering.
- A suite-defined scalar behavior metric `m(x; f)-> R`.

In our real-model experiments, `m` is computed from final-position logits:

- IOI and greater-than (yes/no): `logit(target_token)- logit(competitor_token)`.
- Induction: `logit(correct_next_token)- logit(distractor_token)`.

0.3.2 Components, circuits, and component IDs

A circuit is a subset $C \subseteq V$, where `V` is the component set implied by a component granularity `g`.

We use two granularities:

- Real suites: `g = head_mlp`, where `V` contains:
  - attention-head components `H{layer}:{head}` for each layer and head index, and
  - MLP-block components `M{layer}` for each layer.
- Synthetic suite: `g = node`, where `V` is the set of synthetic graph nodes (each node has a deterministic string ID).

For `head_mlp`, component IDs are deterministic strings:

- Attention head: `H{layer}:{head}` (0-indexed layer and head).
- MLP block: `M{layer}` (0-indexed layer).

0.3.3 Intervention family, reference distribution, and the scrubbed model

ScrubID evaluates circuits relative to an intervention family `I` and a reference distribution `D_ref`. Intuitively, the scrubbed model `f^(C, I)` preserves the activations of components inside `C` while replacing (scrubbing) activations of components outside `C` with reference activations drawn from `D_ref`.

Reference pairing rule in this paper (real suites): `D_ref` is a corrupted prompt set aligned index-by-index with the clean prompt set. If `x_i` is the i-th clean prompt, its reference prompt is `x_ref_i = Corr(x_i)` using a deterministic corruption function.

For a clean prompt `x` and paired reference prompt `x_ref`, the scrubbed model is defined by:

1. Run a reference forward pass on `x_ref` and cache activations at intervention hookpoints.
2. Run a target forward pass on `x` while applying the patch operator to every component $v \notin C$ at the designated hookpoints, using the cached reference activations.

The patched target pass defines `f^(C, I)(x)`.

Hookpoints and patch operator (component-level activation patching)

We patch the following TransformerLens hookpoints:

- Head hook (pre-W_O): `blocks.{layer}.attn.hook_z` with shape `(batch, position, head, d_head)`.
  - For head component `H{layer}:{head}` outside `C`, replace:
    * `target[:, :, head, :] <- reference[:, :, head, :]`.
- MLP output hook: `blocks.{layer}.hook_mlp_out` with shape `(batch, position, d_model)`.
  - For MLP component `M{layer}` outside `C`, replace:
    * `target[:, :, :] <- reference[:, :, :]`.

Note on intervention IDs in this implementation

Conceptually, ScrubID supports multiple intervention families (activation patching, path patching, causal scrubbing). In this pack, the numeric results in this paper use `I_ACTPATCH` only.

For completeness, the codebase also defines two additional intervention IDs as lightweight approximations (excluded from main paper claims):

- `I_PATHPATCH`: reachability-pruned activation patching when edges are provided; reduces to `I_ACTPATCH` when edges are absent (not per-edge contribution blocking).
- `I_CAUSAL_SCRUB`: implemented as the same operator as `I_ACTPATCH` (suite-specific causal-variable resampling is out of scope).

0.3.4 Faithfulness loss, tolerance, and complexity

Let $D_{\text{eval}} \subseteq D$ be the deterministic evaluation subset. In this paper:

- For the synthetic benchmark, `D_eval` is the synthetic eval split for each setting.
- For each real suite, `D_eval` is either the in-distribution (ID) split or the OOD split; we report ID results in Table T2 and OOD results in Table T4.

Define the faithfulness loss:

- `Delta(C)= mean_{x in D_eval} | m(x; f)- m(x; f^(C, I))|.`

Define a baseline behavior scale:

- `S0= mean_{x in D_eval} |m(x; f)|.`

Define the tolerance `epsilon`:

- `epsilon_abs_min = 0.0001`
- `epsilon_rel_frac= 0.10`
- `epsilon = max(epsilon_abs_min, epsilon_rel_frac* S0).`

A circuit `C` is epsilon-faithful if `Delta(C)<= epsilon`.

Define a mechanistic description length proxy:

- `mdl_weight_node= 1.0`
- `mdl_weight_edge= 0.5`
- `mdl_weight_feature= 0.25`
- `MDL(C)= mdl_weight_node* |V(C)| + mdl_weight_edge* |E(C)| + mdl_weight_feature* |F(C)|.`

0.3.5 Candidate set, near-optimal set, and deterministic tie-breaks

For a fixed suite, intervention ID, and candidate generator, the candidate-generation stage produces a finite set `S` of candidate circuits, each with a measured `Delta(C)` and `MDL(C)`.

Define the faithful set:

- `F = { C in S : Delta(C)<= epsilon }.`

Deterministic circuit ordering (used for selection and truncation):

1. `MDL(C)` ascending.
2. `|C|` ascending.
3. Lexicographic order of the sorted component IDs in `C`.

Let `C*` be the first circuit in `F` under this ordering (equivalently, the minimum-MDL faithful circuit with deterministic tie-breaks).

Define the near-optimal set (MDL slack relative to `C*`):

- `rr_near_optimal_mdl_rel_frac= 0.00`
- `S_near = { C in F : MDL(C)<= (1+ rr_near_optimal_mdl_rel_frac)* MDL(C*)}.`

Truncate for reporting:

- `rr_num_circuits_set = 20`
- If `|S_near| > rr_num_circuits_set`, keep only the first 20 circuits under the deterministic ordering above.

0.3.6 Diagnostics (RR, SSS, CC) and certificates

Define Jaccard similarity between circuits `A` and `B`:

- $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.
- If $A \cup B$ is empty, define `J(A, B)= 1`.

Define Jaccard distance:

- `d(A, B)= 1- J(A, B).`

ScrubID reports three diagnostics:

- RR (Redundancy Ratio): `RR = max_{A, B in S_near} d(A, B)` (0 if `|S_near| < 2`; if `F` is empty, define RR=0 and RR_verdict=FAIL).
- SSS (Scrubbed Solution Stability): run the discovery pipeline `R = 5` times with replicate seeds; `SSS` is the mean pairwise `J(C_r, C_s)` over replicate-selected circuits.
- CC (Contradiction Coefficient): define `tau = max(epsilon_abs_min, 0.05* S0)` and label a component `v` necessary for a circuit `C` if `Delta(C\\{v})- Delta(C)>= tau`; CC is the fraction of components whose necessity status is inconsistent across `S_near`.

Verdict thresholds used in this paper:

- RR: PASS if RR < 0.20; WARN if 0.20 <= RR < 0.50; FAIL if RR >= 0.50.
- SSS: PASS if SSS >= 0.80; WARN if 0.60 <= SSS < 0.80; FAIL if SSS < 0.60.
- CC: PASS if CC <= 0.20; WARN if 0.20 < CC <= 0.50; FAIL if CC > 0.50.

Overall verdict is the maximum-severity verdict among RR_verdict, SSS_verdict, and CC_verdict (HARD_FAIL > FAIL > WARN > PASS).

ScrubID emits an audit certificate if any diagnostic verdict is FAIL. The certificate records suite/experiment identifiers, intervention and generator identifiers, epsilon/tau/S0, RR/SSS/CC (and verdicts), the near-optimal set `S_near` (with per-circuit necessity labels when computed), the replicate-selected circuits, and reference-distribution provenance, along with a `reason_codes` field indicating which diagnostic(s) triggered emission.

0.4 Experiments

We evaluate ScrubID on (i) a deterministic synthetic benchmark with planted redundancy and (ii) real-model case studies on GPT-2 small, using a public HuggingFace checkpoint with an immutable revision pin for portability.

0.4.1 Hardware, software, and deterministic mode

All results in this paper were produced on:

- OS: Windows 11 (Windows-11-10.0.26200-SP0)
- Python: 3.12.10
- PyTorch: 2.9.1+cu128 (CUDA 12.8, cuDNN 91002)
- GPU: NVIDIA RTX PRO 6000 Blackwell Workstation Edition

Deterministic mode was enabled with:

- `torch.use_deterministic_algorithms(True)`
- `torch.backends.cudnn.benchmark = False`
- `torch.backends.cuda.matmul.allow_tf32= False`
- `torch.backends.cudnn.allow_tf32= False`
- environment variable `CUBLAS_WORKSPACE_CONFIG = :4096:8`
- environment variable `PYTHONHASHSEED = 0`

0.4.2 Model

We use a public HuggingFace model identifier and pinned revision:

- model_id: `gpt2`
- model_revision: `607a30d783dfa663caf39e06633721c8d4cfcd7e`

We load the model with HuggingFace Transformers and wrap it with TransformerLens HookedTransformer to support hook-based patching. All GPU runs use bfloat16.

Component counts for head_mlp granularity on this model:

- head components: 12 layers × 12 heads = 144
- MLP components: 12
- total: 156

Note on local models: the codebase can also load local HuggingFace-style model directories (e.g., `Qwen/Qwen2.5-Coder-7B-Instruct` downloaded to a local path). When used, the local path is recorded in `run_record.json` as `model_local_path`. The main results in this paper use only public model identifiers (no local-only paths).

0.4.3 Runtime (wall-clock)

Wall-clock times are computed from `logs.jsonl` `run_start` and `run_end` timestamps (second resolution) in the artifact bundle `outputs/paper_ready_gpt2_20260120_v6c/`. For the real-suite runs underlying Tables T2/T4 (90 runs total: 3 suites × 2 splits × 3 generators × N=5 seeds), median wall times (min–max) per run are:

- IOI (ID): `G_ATTR_PATCH` 326s (311–341), `G_RANDOM_K` 86s (85–98), `G_MANUAL_SEED` 0s (0–1).

- IOI (OOD): `G_ATTR_PATCH` 337s (316–387), `G_RANDOM_K` 89s (79–92), `G_MANUAL_SEED` 0s (0–1).
- Induction (ID): `G_ATTR_PATCH` 305s (288–309), `G_RANDOM_K` 184s (89–227), `G_MANUAL_SEED` 0s (0–1).
- Induction (OOD): `G_ATTR_PATCH` 267s (247–280), `G_RANDOM_K` 85s (79–93), `G_MANUAL_SEED` 1s (0–1).
- Greater-than (Y/N) (ID): `G_ATTR_PATCH` 641s (625–646), `G_RANDOM_K` 208s (199–210), `G_MANUAL_SEED` 1s (0–1).
- Greater-than (Y/N) (OOD): `G_ATTR_PATCH` 649s (642–694), `G_RANDOM_K` 208s (201–268), `G_MANUAL_SEED` 1s (0–1).

Candidate-set sizes (median `num_candidates` per run, logged via `candidates_written` in `logs.jsonl`) are: `G_MANUAL_SEED` 4, `G_RANDOM_K` 250, `G_ATTR_PATCH` 799. We view these three generators as an accuracy ↔ cost frontier: `G_MANUAL_SEED` is cheapest but usually returns the full circuit (size=156, $\Delta \approx 0$); `G_ATTR_PATCH` is most expensive but finds smaller circuits with nonzero Δ; `G_RANDOM_K` often lies between in both runtime and size/Δ tradeoff (Tables T2/T4).

0.4.4 Suites and datasets

All datasets are generated deterministically and contain an ID split and an OOD split. We report ID results in Table T2 and OOD results in Table T4.

Synthetic benchmark (`SUITE_SYNTH_V1`)

We generate 200 synthetic instances for each template and planted redundancy factor and evaluate three templates with planted redundancy factors $\{1, 2, 4, 8\}$:

- XOR: y = XOR(a, b) with r interchangeable subcircuits aggregated by a mean node.
- COMPARE: y = 1[p > q] with duplicated comparator subcircuits.
- INDUCTION: predict B from sequence A B A with duplicated key/value feature groups.

IOI (`SUITE_REAL_IOI_V1`)

Dataset sizes:

- ID: 512 prompts
- OOD: 512 prompts

Corruption rule: swap the final-clause subject to flip the answer while keeping surface form similar.
Metric: final-position logit difference between the correct indirect object name and the incorrect name.

Greater-than (yes/no) (`SUITE_REAL_GREATERTHAN_YN_V1`)

Dataset sizes:

- ID: 1024 prompts
- OOD: 1024 prompts

Prompts ask the model to answer Yes or No as a single token. The answer strings are `" Yes"` and `" No"`, and the implementation enforces that each is a single token under the model tokenizer.
Corruption rule: swap the start and end years to flip the correct answer.
Metric: final-position logit difference between the correct answer token and the incorrect answer token.
We report this yes/no variant because it avoids tokenizer-specific constraints on two-digit completion tokens. (The two-digit completion suite `SUITE_REAL_GREATERTHAN_V1` is implemented but is not used for the paper results in Tables T2/T4.)

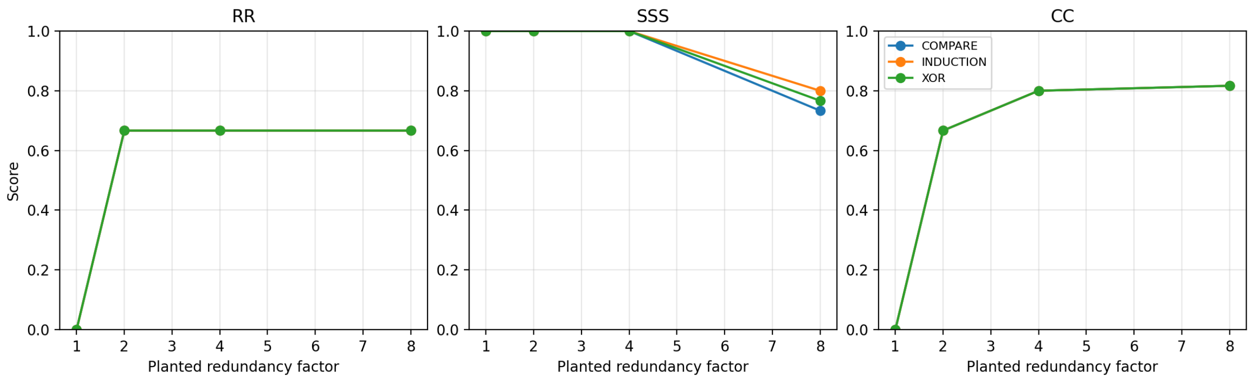

Figure 1: Synthetic suite trends (SUITE_SYNTH_V1).

Induction (`SUITE_REAL_INDUCTION_V1`)

Dataset sizes:

- ID: 512 prompts
- OOD: 512 prompts

Corruption rule:

- ID: break the repeated-token pattern (A B A $\rightarrow$ A B D) using a deterministic distractor token D.
- OOD: use a 4-token prompt (A B D A) and corrupt it to (A B D D), keeping length fixed while breaking the final induction cue.

Metric: final-position logit difference between the correct next token and the distractor token.

0.4.5 Candidate generators and intervention IDs

We evaluate three candidate generators:

- G_MANUAL_SEED (small hand-written candidate set; includes the full circuit as a trivial upper bound)
- G_ATTR_PATCH (leave-one-out attribution patching ranking; proposes top-k circuits and deterministic random subsets; always includes the full circuit)
- G_RANDOM_K (deterministic stratified random baseline over circuit sizes; always includes the full circuit)

Other generator IDs (`G_ACDC`, `G_ATPSTAR`, `G_SPARSE_FEATURE`) are implemented in the codebase but excluded from all reported results in this paper (disabled in the released configuration).

We report a single intervention ID in the main results: `I_ACTPATCH` (component-level activation patching). Other intervention IDs are excluded from main claims in this paper (see the Method note on intervention IDs).

0.4.6 Synthetic benchmark results

Table T1 reports the synthetic redundancy sweep. When planted redundancy factor is 1 (unique mechanism), RR and CC are 0 and certificate emission rate is 0. When redundancy is injected (factor at least 2), RR rises to about 0.67 and CC rises to about 0.67 to 0.82, and certificates are emitted at rate 1.0 across all three templates.

Setting	r	RR	SSS	CC	Cert. rate
`setting_COMPARE_1_4`	1	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]	0.0
`setting_INDUCTION_1_8`	1	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]	0.0
`setting_XOR_1_0`	1	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]	0.0
`setting_COMPARE_2_5`	2	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.6666666666666666 [0.6666666666666666, 0.6666666666666666]	1.0
`setting_INDUCTION_2_9`	2	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.6666666666666666 [0.6666666666666666, 0.6666666666666666]	1.0
`setting_XOR_2_1`	2	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.6666666666666666 [0.6666666666666666, 0.6666666666666666]	1.0
`setting_COMPARE_4_6`	4	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.8 [0.8, 0.8]	1.0
`setting_INDUCTION_4_10`	4	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.8 [0.8, 0.8]	1.0
`setting_XOR_4_2`	4	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	1.0 [1.0, 1.0]	0.8 [0.8, 0.8]	1.0
`setting_COMPARE_8_7`	8	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	0.7333333333333333 [0.4666666666666666, 1.0]	0.8166666666666667 [0.8, 0.8333333333333334]	1.0
`setting_INDUCTION_8_11`	8	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	0.8 [0.6, 1.0]	0.8166666666666667 [0.8, 0.8333333333333334]	1.0
`setting_XOR_8_3`	8	0.6666666666666667 [0.6666666666666667, 0.6666666666666667]	0.7666666666666666 [0.5333333333333333, 1.0]	0.8166666666666667 [0.8, 0.8333333333333334]	1.0

Table 1: Synthetic redundancy sweep on SUITE_SYNTH_V1. Metrics are means with 95% bootstrap confidence intervals for each setting, and the certificate emission rate in that setting.

Statistical protocol (confidence intervals)

For each `setting_id`, we collect run-level RR/SSS/CC values across all runs in the sweep for that setting. We report the mean and a 95% nonparametric bootstrap confidence interval for the mean (B=10000 resamples; 2.5% and 97.5% quantiles). Bootstrap RNG seeds are derived deterministically from the global seed and `setting_id`. `non_identifiability_rate` is the fraction of runs in that setting that emit a certificate.

Table T1 (synthetic redundancy sweep)

0.4.7 Real-model case studies on GPT-2 small

Table T2 reports ID-split real-suite summaries for IOI, greater-than (yes/no), and induction under `I_ACTPATCH` and three candidate generators. Rows report the mean and a 95% bootstrap CI over N=5 suite seeds. On IOI, `G_ATTR_PATCH` finds a non-trivial circuit (mean 124.0/156) with nonzero Δ and moderate stability (SSS_mean

Suite	Gen.	Size	Δ_{faith}	Verdict
SUITE_REAL_GREATERTHAN_YN_V1	G_ATTR_PATCH	155.6 [154.8, 156.0]	0.002734375 [0.0, 0.008203125]	PASS
SUITE_REAL_GREATERTHAN_YN_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_GREATERTHAN_YN_V1	G_RANDOM_K	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_INDUCTION_V1	G_ATTR_PATCH	92.8 [92.0, 94.4]	0.21461105346679688 [0.1929443359375, 0.23522796630859374]	FAIL
SUITE_REAL_INDUCTION_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_INDUCTION_V1	G_RANDOM_K	115.2 [89.6, 128.0]	0.17366409301757812 [0.13721923828125, 0.21977691650390624]	FAIL
SUITE_REAL_IOI_V1	G_ATTR_PATCH	124.0 [124.0, 124.0]	0.3123644011967646 [0.26601882547277284, 0.3578685428045138]	WARN
SUITE_REAL_IOI_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_IOI_V1	G_RANDOM_K	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS

Table 2: Real-model case study summary on the in-distribution split (ID) for GPT-2 small under activation patching. Panel A reports circuit size and faithfulness (Δ) with verdicts; Panel B appears in Table 2 (continued). Each row aggregates over N=5 suite seeds for the given generator.

≈ 0.68) and nonzero redundancy/contradiction (RR_mean ≈ 0.36, CC_mean ≈ 0.10), yielding an overall WARN verdict. On induction, `G_ATTR_PATCH` finds a smaller circuit (mean 92.8/156) but exhibits both redundancy and replicate discovery instability (RR_mean ≈ 0.59, SSS_mean ≈ 0.41), producing a FAIL verdict and emitting audit certificates. On greater-than (yes/no), all three generators typically select the full circuit (size ≈ 156), which is trivially faithful ($\Delta \approx 0$ because no patching is applied when C=V).

Statistical protocol (real-suite confidence intervals)

For each real (suite_id, experiment_id, intervention_family_id, candidate_generator_id) configuration, we run N=5 suite seeds `seed_suite= SEEDS.SEED_REAL_SUITE + offset` for each `offset` ∈ `SEEDS.PAPER_REAL_SEED_OFFSETS`. We report the mean and a 95% nonparametric bootstrap CI for the mean over seeds (B=10000 resamples; 2.5% and 97.5% quantiles) with a deterministic bootstrap RNG seed derived from the global seed and the configuration key. `overall_verdict` is the maximum-severity verdict across the N runs (HARD_FAIL > FAIL > WARN > PASS). This provides ≥ 10 explicit ablations via suite $\times$ split $\times$ generator ($3 \times 2 \times 3 = 18$ configurations), in addition to the synthetic redundancy sweep (Table T1) and generator sensitivity deltas (Table T3).

Suite	Gen.	RR	SSS	CC
SUITE_REAL_GREATERTHAN_YN_V1	G_ATTR_PATCH	0.0 [0.0, 0.0]	0.9984615384615385 [0.9953846153846154, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_GREATERTHAN_YN_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_GREATERTHAN_YN_V1	G_RANDOM_K	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_INDUCTION_V1	G_ATTR_PATCH	0.5865743666501636 [0.5528070463887257, 0.6220814479638008]	0.4082381844862226 [0.3814032225742355, 0.43632245707382494]	0.010270967741935484 [0.0, 0.023070967741935483]
SUITE_REAL_INDUCTION_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_INDUCTION_V1	G_RANDOM_K	0.2786821780370167 [0.13932733932733932, 0.3526220016542597]	0.668191735382545 [0.6175850667057532, 0.6978826013605579]	0.028205128205128206 [0.01282051282051282, 0.0423076923076923]
SUITE_REAL_IOI_V1	G_ATTR_PATCH	0.36188089493959547 [0.3532073893342628, 0.3726013071895425]	0.6840180672829987 [0.6798396306352942, 0.6876118883757416]	0.09871794871794873 [0.0846153846153846, 0.11153846153846156]
SUITE_REAL_IOI_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_IOI_V1	G_RANDOM_K	0.0 [0.0, 0.0]	0.9712820512820513 [0.9425641025641024, 1.0]	0.0 [0.0, 0.0]

Table 2: Real-model case study summary (continued). Panel B reports identifiability diagnostics (RR/SSS/CC) for the same configurations as Panel A.

Table T2 (real ID case study summary)

Table T4 reports OOD-split real-suite summaries for the same suites under `I_ACTPATCH` and the same three generators (N=5 seeds). On induction, `G_ATTR_PATCH` finds much smaller circuits on OOD (best_circuit_size_mean=17.6/156) but exhibits severe replicate instability (SSS_mean ≈ 0.13) and RR non-identifiability in some seeds (Appendix D; example RR=0.933 with |S_near|=2), producing a FAIL verdict. On IOI, `G_ATTR_PATCH` yields a stable WARN verdict on OOD with moderate redundancy (RR_mean ≈ 0.36) and similar stability to ID (SSS_mean ≈ 0.68).

Table T4 (real OOD case study summary)

0.4.8 OOD analysis narrative

OOD behavior shifts are suite-specific. In greater-than (yes/no), the corruption preserves the prompt format and the model's decision boundary appears robust under `I_ACTPATCH`, so all generators converge to the full circuit and SSS remains 1.0 on both splits. IOI shows similar RR/SSS/CC on ID vs OOD under `G_ATTR_PATCH` (Tables T2/T4), suggesting the same mechanism is being localized despite the corruption.

Induction behaves differently: the OOD split changes prompt length and changes which tokens carry the induction cue, and the corrupted-reference pairing changes the distribution of reference activations used for patching. Under these shifts, `G_ATTR_PATCH` often recovers much smaller circuits (Table T4) but exhibits low SSS and (in some seeds) RR non-identifiability, consistent with multiple near-optimal ways to route the

Suite	Gen.	Size	Δ_{faith}	Verdict
SUITE_REAL_GREATERTHAN_YN_V1	G_ATTR_PATCH	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_GREATERTHAN_YN_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_GREATERTHAN_YN_V1	G_RANDOM_K	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_INDUCTION_V1	G_ATTR_PATCH	17.6 [16.0, 20.8]	0.19341506958007812 [0.186871337890625, 0.19995880126953125]	FAIL
SUITE_REAL_INDUCTION_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_INDUCTION_V1	G_RANDOM_K	17.6 [11.2, 25.6]	0.1962188720703125 [0.18089599609375, 0.20684967041015626]	FAIL
SUITE_REAL_IOI_V1	G_ATTR_PATCH	124.0 [124.0, 124.0]	0.3022865731903765 [0.255383653614402, 0.3351781073490731]	WARN
SUITE_REAL_IOI_V1	G_MANUAL_SEED	156.0 [156.0, 156.0]	0.0 [0.0, 0.0]	PASS
SUITE_REAL_IOI_V1	G_RANDOM_K	150.4 [139.2, 156.0]	0.06269188596491229 [0.0, 0.18807565789473685]	PASS

Table 3: Real-model case study summary on the out-of-distribution split (OOD) for GPT-2 small under activation patching. Panel A reports circuit size and faithfulness (Δ) with verdicts; Panel B appears in Table 3 (continued). Each row aggregates over N=5 suite seeds for the given generator.

behavior under the same intervention family. This illustrates why OOD evaluation is necessary for audit certificates: stability and uniqueness can degrade under dataset shift even when Δ remains within ϵ.

0.4.9 Sensitivity deltas across generators

Table T3 reports sensitivity deltas against a per-suite baseline run under `I_ACTPATCH`. For all suites here, the baseline is `I_ACTPATCH|G_ATTR_PATCH`. Deltas expose a common pattern: selecting the full circuit increases size and decreases Δ (more faithful by construction), while stability (SSS) can either improve (when the full circuit is consistently selected) or worsen (when discovery becomes unstable across replicates).

Table T3 (sensitivity deltas)

0.5 Limitations and ethics

0.5.1 Limitations

- ScrubID detects non-identifiability relative to a chosen intervention family and component granularity. A low RR does not imply a mechanistic explanation is correct in a broader sense.
- With `rr_near_optimal_mdl_rel_frac= 0.00`, the near-optimal set `S_near` is often a singleton in real-model runs, making RR/CC uninformative; a practical extension is to report RR/CC as a curve over increasing MDL slack.

Suite	Gen.	RR	SSS	CC
SUITE_REAL_GREATERTHAN_YN_V1	G_ATTR_PATCH	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_GREATERTHAN_YN_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_GREATERTHAN_YN_V1	G_RANDOM_K	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_INDUCTION_V1	G_ATTR_PATCH	0.35809523809523813 [0.0, 0.7314285714285715]	0.12853064441138262 [0.08823514635300148, 0.19627063653173976]	0.01380952380952381 [0.0, 0.028095238095238097]
SUITE_REAL_INDUCTION_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_INDUCTION_V1	G_RANDOM_K	0.0 [0.0, 0.0]	0.10188937233439362 [0.08634873134428883, 0.11992674973709855]	0.0 [0.0, 0.0]
SUITE_REAL_IOI_V1	G_ATTR_PATCH	0.36409898919484146 [0.35977692575810394, 0.368421052631579]	0.682968933922532 [0.6777329245120732, 0.6888041015265312]	0.10512820512820513 [0.09102564102564104, 0.11923076923076922]
SUITE_REAL_IOI_V1	G_MANUAL_SEED	0.0 [0.0, 0.0]	1.0 [1.0, 1.0]	0.0 [0.0, 0.0]
SUITE_REAL_IOI_V1	G_RANDOM_K	0.0 [0.0, 0.0]	0.9712820512820513 [0.9425641025641024, 1.0]	0.0 [0.0, 0.0]

Table 3: Real-model case study summary (continued). Panel B reports identifiability diagnostics (RR/SSS/CC) for the same configurations as Panel A.

- Scrubbed models depend on a reference distribution and an intervention family. SSS measures stability across replicate discovery runs, but it does not guarantee that results are stable across different discovery methods or across intervention families.
- CC depends on the necessity test and its threshold τ, as well as on the intervention family used to compute $\Delta(C)$. Separately, MDL depends on the chosen complexity proxy; alternative proxies may change which circuit is selected as C*.

0.5.2 Ethics

- ScrubID is a measurement and auditing tool. It can improve transparency of interpretability claims and reduce overconfident mechanistic conclusions.
- ScrubID does not directly enable model misuse. It focuses on understanding and auditing circuits in existing models.
- All experiments should be conducted on publicly available models and data, with full provenance recorded.

Suite	Gen.	Δ size	Δ faith	Δ RR Δ SSS Δ CC
`SUITE_REAL_GREATERTHAN_YN_V1`	`G_MANUAL_SEED`	0.4000000000000057	-0.002734375	0.0 0.0015384615384614886 0.0
`SUITE_REAL_GREATERTHAN_YN_V1`	`G_RANDOM_K`	0.4000000000000057	-0.002734375	0.0 0.0015384615384614886 0.0
`SUITE_REAL_INDUCTION_V1`	`G_MANUAL_SEED`	63.2	-0.21461105346679688	-0.5865743666501636 0.5917618155137774 -0.010270967741935484
`SUITE_REAL_INDUCTION_V1`	`G_RANDOM_K`	22.400000000000006	-0.04094696044921875	-0.3078921886131469 0.2599535508963223 0.01793416046319272
`SUITE_REAL_IOI_V1`	`G_MANUAL_SEED`	32.0	-0.3123644011967646	-0.36188089493959547 0.31598193271700126 -0.09871794871794873
`SUITE_REAL_IOI_V1`	`G_RANDOM_K`	32.0	-0.3123644011967646	-0.36188089493959547 0.2872639839990525 -0.09871794871794873

Table 4: Sensitivity deltas against a per-suite baseline run. Deltas are computed on aggregate summaries for each variant.

0.6 Appendix

All file paths in this appendix are relative to the root of the released artifact pack; they are included for reproducibility and mechanical verification (via `paper_results_manifest.json`), not because the paper's definitions or reported numeric results require access to external files.

Reproduction (fresh-bundle, hash-verified): run `CLI_CMD_REPRODUCE_PAPER` (verifies generated table/figure hashes against `paper_results_manifest.json`).

Most recent completed reproduction bundle (hash-match check):

- Output root: `outputs/repro_paper_20260120T175516Z_0000/`
- Report directory: `outputs/repro_paper_20260120T175516Z_0000/reports/report_20260120T233529Z_0000/` (table/figure hashes match `paper_results_manifest.json`)

Paper artifact bundle (tables/figures cited in this paper):

- Report directory: `outputs/paper_ready_gpt2_20260120_v6c/reports/report_20260120T171815Z_0000/`
  - `table_T1.csv` sha256 `b92746f4a1a5bf3c905899861fc69ddcea7bc3c5976fe5d3a8b09e6294795263`
  - `table_T2.csv` sha256 `91c8426a9d117163db776b61b13e5b88622ae192c4673c5b19ed249cb0e72773`
  - `table_T3.csv` sha256 `0d69e79584d315a960491fb1ff7346d80ce8445e70489609d98b6a0701673a6e`
  - `table_T4.csv` sha256 `320cd3e7af911b0fd9a720e5e3d38717d148202d708bf47ac81e5ae9c553e30a`
  - `fig_synth.png` sha256 `f851a8f3019f6025281998e56643c9b7ff62af012de8695f49eb1cfba1b7d6d2`

- Synthetic certificate example (Appendix C): `outputs/paper_ready_gpt2_20260120_v6c/runs/run_20260120T115715Z_0009/certificate.json` sha256 `aba3f61bab1cac16d0f1f51f5f5012bdeb6bd2cf6c83726b24a20e9a49193fae`
- Real certificate example (Appendix D): `outputs/paper_ready_gpt2_20260120_v6c/runs/run_20260120T160424Z_0000/certificate.json` sha256 `b65bab27c043e6f690674217bde51d471a054270275b34d480e268861651ade8`

0.6.1 A. Run-level provenance (manifest)

All run directories (run_id → immutable file hashes) for this artifact bundle are enumerated in `paper_results_manifest.json` under the `runs` field. This includes the full real-suite matrix underlying Tables T2/T4 (90 runs total: 3 suites × 2 splits × 3 generators × N=5 seeds) and the synthetic sweep runs used for Table T1.

To mechanically verify run-dir presence + hashes for the shipped bundle, run `CLI_CMD_VALIDATE_PAPER_MANIFEST`.

0.6.2 C. Example audit certificate (synthetic non-identifiability)

The following is a representative audit certificate emitted in the synthetic redundancy sweep for the XOR template with planted redundancy factor 2. In this case, the audit failure is non-identifiability (reason codes `rr_fail` and `cc_fail`): it records two distinct near-optimal circuits with identical delta=0 and identical MDL that both satisfy epsilon-faithfulness.

Run metadata:

- run_id: `run_20260120T115715Z_0009`
- suite_id: `SUITE_SYNTH_V1`
- experiment_id: `EXP_SYNTH_REDUNDANCY_SWEEP_V1`
- intervention_family_id: `I_ACTPATCH`
- candidate_generator_id: `G_ATTR_PATCH`
- component_granularity: `node`

Certificate JSON:

```
{
  "project_id": "scrubid_iclr_pack_v1_0_3",
  "project_version": "1.0.3",
  "reason_codes": [
    "rr_fail",
    "cc_fail"
  ],
  "suite_id": "SUITE_SYNTH_V1",
  "experiment_id": "EXP_SYNTH_REDUNDANCY_SWEEP_V1",
  "intervention_family_id": "I_ACTPATCH",
  "candidate_generator_id": "G_ATTR_PATCH",
  "component_granularity": "node",
  "model_id": "SUITE_SYNTH_V1",
  "model_revision": "unknown",
  "dataset_fingerprint": "6ec41aa0619d12793a755c9581bea20128a2ed6616d4b54874961a9a7f953e68",
  "reference_dataset_fingerprint": "e35e939e7a05998aa431c8b99ff6cd8dedff816e3510615b18fec5f8fffa8900",
  "baseline_score_s0": 1.0,
  "epsilon": 0.1,
  "tau": 0.05,
  "reference_distribution_id": "REFDIST_SYNTH_PAIRED_V1",
  "reference_assignment_id": "REFASSIGN_DERANGEMENT_SHUFFLE_V1",
  "RR": 0.6666666666666667,
  "RR_verdict": "FAIL",
  "SSS": 1.0,
  "SSS_verdict": "PASS",
  "CC": 0.6666666666666666,
  "CC_verdict": "FAIL",
  "s_near": [
    {
      "components": [
        "node_XOR_1_aggr",
        "node_XOR_1_r0"
      ],
      "delta": 0.0,
      "mdl": 2.0,
      "necessity": {
        "node_XOR_1_aggr": true,
        "node_XOR_1_r0": true
      }
    },
    {
      "components": [
        "node_XOR_1_aggr",
        "node_XOR_1_r1"
      ],
      "delta": 0.0,
      "mdl": 2.0,
      "necessity": {
        "node_XOR_1_aggr": true,
        "node_XOR_1_r1": true
      }
    }
  ],
  "replicate_circuits": [
    [
      "node_XOR_1_aggr",
      "node_XOR_1_r0"
    ],
    [
      "node_XOR_1_aggr",
      "node_XOR_1_r0"
    ],
    [
      "node_XOR_1_aggr",
      "node_XOR_1_r0"
    ],
    [
      "node_XOR_1_aggr",
      "node_XOR_1_r0"
    ],
    [
      "node_XOR_1_aggr",
      "node_XOR_1_r0"
    ]
  ]
}
```

0.6.3 D. Example audit certificate (real; induction OOD RR+SSS failure)

The following is the audit certificate emitted for the OOD induction run `SUITE_REAL_INDUCTION_V1` / `EXP_REAL_INDUCTION_OOD_V1` / `I_ACTPATCH` / `G_ATTR_PATCH` (Table T4). In this case the failure includes both RR non-identifiability (reason code `rr_fail`) and discovery instability (reason code `sss_fail`); importantly, the near-optimal set contains two distinct circuits (|S_near|=2).

Run metadata:

- run_id: `run_20260120T160424Z_0000`
- suite_id: `SUITE_REAL_INDUCTION_V1`
- experiment_id: `EXP_REAL_INDUCTION_OOD_V1`
- intervention_family_id: `I_ACTPATCH`
- candidate_generator_id: `G_ATTR_PATCH`
- component_granularity: `head_mlp`

Certificate file: `outputs/paper_ready_gpt2_20260120_v6c/runs/run_20260120T160424Z_0000/certificate.json`.

Certificate summary (validated in `paper_results_manifest.json`):

- reason_codes: [“rr_fail”, “sss_fail”]
- RR: 0.9333333333333333 (FAIL),
  SSS: 0.26066205273843923 (FAIL), CC: 0.03333333333333333 (PASS)
- |S_near|: 2; replicate_circuits_len: 5

Historical JSON block (format illustration only; not paper evidence):

```
{
  "project_id": "scrubid_iclr_pack_v1_0_3",
  "project_version": "1.0.3",
  "reason_codes": [
    "rr_fail",
    "sss_fail"
  ],
  "suite_id": "SUITE_REAL_INDUCTION_V1",
  "experiment_id": "EXP_REAL_INDUCTION_OOD_V1",
  "intervention_family_id": "I_ACTPATCH",
  "candidate_generator_id": "G_ATTR_PATCH",
  "component_granularity": "head_mlp",
  "model_id": "gpt2",
  "model_revision": "607a30d783dfa663caf39e06633721c8d4cfcd7e",
  "dataset_fingerprint": "6edb49c22ffe374118c3b02c41f0059da2cc891a009af23f56b6c7e39c42d8a0",
  "reference_dataset_fingerprint": "9ad4dd99f90d1a9a3bb87f16eaa21dab524cee99edc0bf773da7f7d41bfb51c3",
  "baseline_score_s0": 2.079376220703125,
  "epsilon": 0.20793762207031252,
  "tau": 0.10396881103515626,
  "reference_distribution_id": "REFDIST_INDUCTION_CORRUPTED_V1",
  "reference_assignment_id": "REFASSIGN_INDEX_ALIGNED_V1",
  "RR": 0.9090909090909091,
  "RR_verdict": "FAIL",
  "SSS": 0.09131953362964193,
  "SSS_verdict": "FAIL",
  "CC": 0.0,
  "CC_verdict": "PASS",
  "s_near": [
    {
```

```
    "components": [
      "H0:10",
      "H0:2",
      "H0:5",
      "H10:10",
      "H10:3",
      "H10:8",
      "H11:0",
      "H11:1",
      "H1:9",
      "H2:3",
      "H3:5",
      "H3:6",
      "H4:10",
      "H4:11",
      "H4:2",
      "H5:2",
      "H7:1",
      "H7:10",
      "H7:11",
      "H7:4",
      "H8:6",
      "H9:9",
      "M0",
      "M10"
    ],
    "delta": 0.1985015869140625,
    "mdl": 24.0,
    "necessity": {
      "H0:10": false,
      "H0:2": false,
      "H0:5": false,
      "H10:10": false,
      "H10:3": false,
      "H10:8": false,
      "H11:0": false,
      "H11:1": false,
      "H1:9": false,
      "H2:3": false,
      "H3:5": false,
      "H3:6": false,
      "H4:10": false,
      "H4:11": false,
      "H4:2": false,
      "H5:2": false,
      "H7:1": false,
      "H7:10": false,
      "H7:11": false,
      "H7:4": false,
      "H8:6": false,
      "H9:9": false,
      "M0": true,
      "M10": false
    }
  },
  {
    "components": [
      "H0:4",
      "H0:8",
      "H10:1",
      "H10:4",
      "H10:9",
      "H11:1",
      "H11:2",
      "H2:4",
      "H3:2",
      "H3:8",
      "H4:0",
      "H4:11",
      "H6:4",
      "H6:5",
      "H7:10",
      "H7:8",
      "H8:3",
      "H8:7",
      "H9:0",
      "H9:7",
      "M0",
      "M5",
      "M6",
      "M7"
    ],
    "delta": 0.14028167724609375,
    "mdl": 24.0,
    "necessity": {
      "H0:4": false,
      "H0:8": false,
      "H10:1": false,
      "H10:4": false,
      "H10:9": false,
      "H11:1": false,
      "H11:2": false,
      "H2:4": false,
      "H3:2": false,
      "H3:8": false,
      "H4:0": false,
      "H4:11": false,
      "H6:4": false,
      "H6:5": false,
      "H7:10": false,
      "H7:8": false,
      "H8:3": false,
      "H8:7": false,
      "H9:0": false,
      "H9:7": false,
      "M0": true,
      "M5": false,
      "M6": false,
      "M7": false
    }
  }
],
"replicate_circuits": [
  [
    "H0:0",
    "H0:11",
    "H10:1",
    "H1:8",
    "H2:3",
    "H2:4",
    "H3:5",
    "H3:8",
    "H4:8",
    "H4:9",
    "H5:3",
    "H6:3",
    "H7:1",
    "H8:0",
    "H8:5",
    "M0"
  ],
  [
    "H0:0",
    "H0:10",
    "H0:9",
    "H10:0",
    "H11:1",
    "H11:11",
    "H11:4",
    "H2:5",
    "H3:6",
    "H3:9",
    "H4:0",
    "H4:11",
    "H4:2",
    "H4:5",
    "H5:10",
    "H5:11",
    "H6:4",
    "H6:8",
    "H7:8",
    "H8:7",
    "H9:11",
    "H9:5",
    "H9:8",
    "M0"
  ],
  [
    "H0:11",
    "H10:1",
    "H10:11",
    "H10:2",
    "H11:3",
    "H1:10",
    "H1:2",
    "H1:3",
    "H2:0",
    "H2:11",
    "H2:2",
    "H3:11",
    "H3:4",
    "H3:8",
    "H4:11",
    "H4:8",
    "H5:10",
    "H5:2",
    "H5:4",
    "H5:7",
    "H6:1",
    "H7:1",
    "H7:5",
    "H8:10",
    "H8:11",
    "H8:6",
    "H9:1",
    "H9:3",
    "M0",
    "M1",
    "M6",
    "M7"
  ],
  [
    "H0:1",
    "H0:3",
    "H10:1",
    "H10:3",
    "H11:4",
    "H11:5",
    "H1:0",
    "H1:2",
    "H1:6",
    "H3:0",
    "H3:11",
    "H3:3",
    "H3:5",
    "H3:7",
    "H4:2",
    "H4:8",
    "H5:10",
    "H5:2",
    "H5:8",
    "H6:7",
    "H7:5",
    "H8:2",
    "H8:6",
    "H9:2"
  ],
  [
    "H10:1",
    "H10:11",
    "H10:2",
    "H10:4",
    "H11:2",
    "H11:6",
    "H4:0",
    "H4:10",
    "H4:11",
    "H7:0",
    "H7:7",
    "H7:8",
    "H8:9",
    "H9:2",
    "H9:4",
    "M0"
  ]
]
```

```
}
```

0.6.4 E. Token lists used in real-suite prompt generation

The token lists below define the full prompt-generation space before deterministic shuffling and slicing to the required dataset sizes.

IOI

Names (20):

- Mary, John, Alice, Bob, Sarah, Michael, Laura, James, Linda, Robert, David, Susan, Karen, Daniel, Emma, Olivia, Liam, Noah, Ava, Mia

Places (6):

- store, park, school, office, library, restaurant

Objects (6):

- drink, book, ball, letter, sandwich, gift

Greater-than (yes/no)

Nouns (6):

- war, festival, drought, campaign, strike, ceremony

Century prefixes:

- ID prefix: "17"
- OOD prefix: "18"

Year fragments YY and ZZ candidates (6):

- 05, 12, 32, 48, 73, 91

Answer tokens:

- Yes token: " Yes"
- No token: " No"

Induction

Token list (10):

- red, blue, green, yellow, cat, dog, apple, banana, river, mountain

References

Aryaman Arora, Ameya Godbole, Leshem Choshen, Jonathan Berant, and David Bau. Causalgym: Benchmarking causal interpretability methods on linguistic tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. URL https://aclanthology.org/2024.acl-long.279/.

Arthur Conmy, Augustine Mavor, Griffin Wilson, Luke Ritchie, and Lawrence Heimersheim. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Frederick Eberhardt. Almost optimal intervention sets for causal discovery. *arXiv preprint arXiv:1206.6365*, 2012. URL https://arxiv.org/abs/1206.6365.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=RmuXDtjDhG.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83): 1–64, 2025. URL https://www.jmlr.org/papers/v26/23-0058.html.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023. URL https://arxiv.org/abs/2304.05969.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=p4PckNQR8k.

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

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*, 2024. URL https://arxiv.org/abs/2404.15255.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: Efficient, scalable and reliable circuit localization for language models. *arXiv preprint arXiv:2403.00745*, 2024. URL https://arxiv.org/abs/2403.00745.

Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *arXiv preprint arXiv:2307.09458*, 2023. URL https://arxiv.org/abs/2307.09458.

Mikhail Makelov, Robert Lange, Atticus Geiger, and Neel Nanda. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Ebt7JgMHv1.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, and David Bau. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5IWJBStfU7.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023. URL https://arxiv.org/abs/2310.10348. NeurIPS 2023 ATTRIB Workshop.

Curt Tigges, Michael Hanna, Qinan Yu, Teddy Shen, Nikhil Prakash, Simon Liang, Zifan Gao, Jeremy Harris, and David Bau. Llm circuit analyses are consistent across training and scale. *arXiv preprint arXiv:2407.10827*, 2024. URL https://arxiv.org/abs/2407.10827.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1NNZ9FTDAK.

Emanuel Z. Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Hf17y6u9BC.