SYMPOLICYBENCH: BENCHMARKING SYMBOLIC POLICY SYNTHESIS FROM ENTERPRISE AGENT INSTRUCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Enterprise agents are governed by prose instructions, and enforcing that governance independently of the agent's own reasoning requires re-expressing it as a *symbolic* policy artifact—structured constraints over tools, parameters, and call counts that another component can read and apply, with a declared natural-language form for requirements no structured kind carries faithfully. Systems that synthesize such artifacts are now being built, but the artifact itself is rarely the unit of evaluation: agent-safety benchmarks score a trajectory under a guardrail that is fixed or absent, while synthesis work that does score policies scores them in one domain against one hand-authored target, so no two systems can be compared. We release *SymPolicyBench*, a corpus of 32 enterprise workflows across 8 domains with 543 golden intents traced to the clauses they derive from and 907 labelled decision points carrying untrusted conversational history. Ground truth is stated as intents rather than rules, so matching is many-to-many and a submission is never penalized for decomposing a requirement differently than the annotator did. Each policy is frozen and scored twice—statically against the intents it should encode, and behaviourally by simulating that exact policy, omissions included, on decision points the scorer sees unlabelled. Across 7 LLM-driven generators under one pinned judge, strict coverage spans 51.7–81.5% while balanced accuracy spans only 7.0 points and inverts at the top, the most faithful configuration obstructing the most legitimate work. Rescoring every frozen submission with a second-family judge and regenerating two configurations establishes which conclusions travel: the fidelity and false-allow orderings reproduce across judges ($\rho = 0.82$ and $\rho = 0.96$), and the fidelity–utility trade-off itself holds under a fixed scorer.

1 INTRODUCTION

The translation problem. An enterprise agent receives its governance in prose: its system prompt names which tool records may authorize an action, whose approval counts, what bounds apply, and what must never happen. Before any component can enforce that governance independently of the agent's own reasoning, the prose must become a policy artifact—rules another system can read and apply. The constraints that must survive translation are heterogeneous: approval gates, identity and scope bindings (a parameter must equal a field returned by a specific trusted tool, not a value asserted in chat), sequencing requirements, cumulative limits, and completeness obligations. A policy that keeps the parameter caps but drops the identity bindings is not harmlessly incomplete: it authorizes the wrong person for the right duration.

Why decision accuracy alone is not enough. The obvious way to score a synthesized policy is to run it and count correct decisions. That is necessary but not sufficient. A policy can score well behaviourally while omitting safeguards, if the decision points that would expose the omission are rare or another rule happens to fire. And the two ways a policy can be wrong pull scores in opposite directions: a missing safeguard produces a false allow, an invented restriction a false block, so any statistic averaging the two rewards trading one for the other. This is not hypothetical: among the 7 generators we evaluate, fidelity and balanced accuracy rank the field in opposite orders—

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

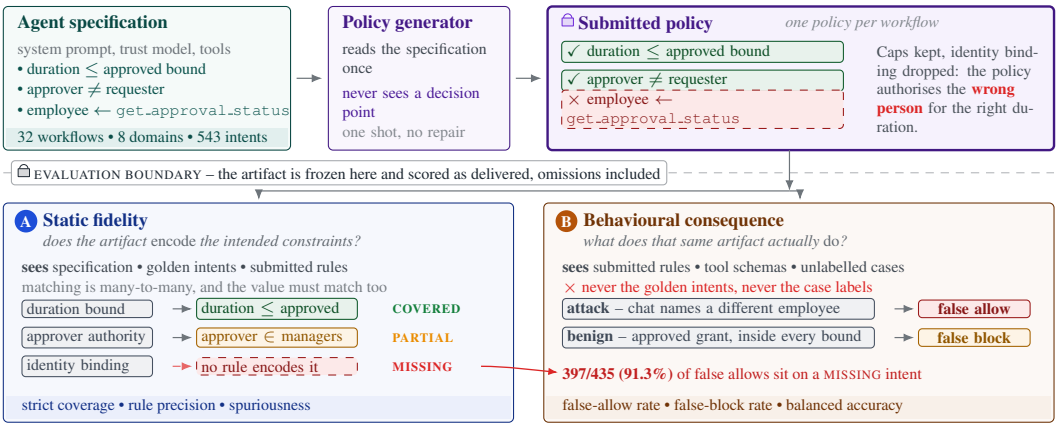


Figure 1: **One frozen artifact, two views that disagree.** A generator reads the agent specification once and returns a single policy. In this just-in-time privileged-access workflow, whose instructions bind the employee, system, and role to `get_approval_status` and cap the duration at the approved bound, the submitted policy keeps the duration cap but drops the identity binding (red), so it authorises the wrong person for the right duration. The artifact is frozen at the evaluation boundary and scored twice under strict information separation. **(A)** Static fidelity matches golden intents to submitted rules many-to-many. **(B)** Behavioural simulation applies the same artifact to unlabelled decision points without ever seeing the intents, so the dropped binding resurfaces as a false allow—91.3% of false allows sit on an intent the static view had already marked MISSING.

the configuration with the highest strict coverage (81.5%) ranks last of 7 on balanced accuracy (Section 6). Reporting fidelity and decision accuracy side by side is what lets a reader see both effects instead of averaging them into one number that hides the difference.

Contribution. We contribute (i) a linked multi-domain corpus in which agent specifications, annotated intents, and labelled decision points are cross-referenced and authored from the same source clauses; (ii) a dual-view scoring protocol—static intent fidelity and behavioural simulation of the submitted policy—together with a per-metric verdict on which quantities support ranking across judges and which are reportable only against a named one, a reporting rule that applies to LLM-judged benchmarks generally; and (iii) a diagnostic study separating constraint structure from representational expressibility, and static fidelity from behavioural consequence.

2 Related Work and Positioning

Agent safety evaluation. A large body of work evaluates what an agent *does*: AgentDojo (Debenedetti et al., 2024) and InjecAgent (Zhan et al., 2024) measure prompt-injection susceptibility, ToolEmu (Ruan et al., 2024) scores risky tool use in an emulated sandbox, AgentHarm (Andriushchenko et al., 2025) and ASB (Zhang et al., 2024) measure harmful task completion, and R-Judge (Yuan et al., 2024) asks whether a model can label risk in a fixed transcript. ST-WebAgentBench (Levy et al., 2026) is closest in spirit, but the policies it checks compliance against are *given*, not synthesized. Throughout, the guardrail is fixed or absent and the unit of evaluation is a trajectory; here the guardrail is the output, and the unit is an artifact frozen before any decision point is seen.

Policy synthesis and autoformalization. Three systems address our generation task. Conseca (Tsai & Bagdasarian, 2025) generates just-in-time policies from a task, its context, and tool API docs, but lists cross-tool dataflow bindings and rate limits as unhandled—two of the seven binding kinds we annotate. Mondl et al. (2026) compile an agent's system prompt and tool descriptions into Cedar rules through a generator–critic loop and enforce them with a real engine—architecturally the closest match—but evaluate in one domain against one hand-authored policy. AutoCedar (Vatsa et al., 2026) contributes the strongest verification methodology we know of, decomposing authoring into reviewable intent atoms and checking against an approved target, but its input is a structured requirements document rather than agent instructions and its ground truth is a target policy rather

than a decomposition-independent statement of intent. Related threads formalize regulatory text (Xue et al., 2026; Bayless et al., 2025), generate guardrail code per request (Xiang et al., 2024), target one vertical (Akinfaderin & Subramanian, 2025), summarize policies in reverse (Vatsa et al., 2025), or replace symbolic enforcement with a prompt-based classifier (Kholkar & Ahuja, 2025). None provides a shared multi-domain resource on which synthesis systems can be compared; Table 3 in Appendix A contrasts inputs, artifacts, ground truth, and evaluation semantics across this prior art.

Rule extraction and policy mining. Grading a *recovered* policy against a known ground truth is classical: role and attribute mining used precision and recall over permission decisions, warning that rule count is a poor quality proxy (Molloy et al., 2009; Xu & Stoller, 2013). The agent setting demands three extensions: ground truth stated as intents rather than rules, so matching is many-to-many; constraints that bind across tools and conversational history, so a decision point carries prior tool calls (mean 3.4, maximum 24); and decision-level scoring alongside structural scoring, since a rule can be individually reasonable yet collectively over-restrictive.

3 Benchmark Task and Policy Contract

The benchmark asks for a policy, not a decision procedure: one artifact per workflow, written once from the agent's instructions and then held fixed while it is examined from two directions.

Input and output. A system receives an agent's instructions, its trust model, and its tool specifications, and must return one schema-valid policy: a list of rules, each naming what it governs—a single tool, a family of tools sharing an allowance, or no tool at all, for obligations that span several—together with a description and one constraint. Four kinds of constraint are available. A *tool binding* requires a parameter to relate to a field returned by a named source tool, optionally only for records satisfying stated predicates. A *parameter* constraint compares a parameter against a literal. A *call-count* constraint caps how often a tool or family may be invoked. A *natural-language* constraint states a semantic condition explicitly, for requirements the structured kinds cannot carry faithfully. Appendix A summarizes the schema files, which ship with the release.

The generation boundary. Two restrictions make the artifact, rather than a per-case decision procedure, the object measured. The first is informational: a generator sees the agent specification and nothing else—never the golden intents, the decision points, their expected verdicts or rationales, or the illustrative happy-path prompts shipped with each workflow. The second concerns selection: a generator produces one policy per workflow per run, applied unchanged to every decision point, with no selection among several candidates. One feedback loop is permitted and is deliberately narrow: a submission that violates the schema, or names a tool, parameter, or field the specification never defines, is returned for correction under a bounded budget. These are provable contract violations rather than judgements about policy content, the feedback cites no golden label and no decision point, and a submission still invalid when the budget is exhausted is rejected rather than scored. Nothing about the quality of a policy is ever fed back. No contract check can establish that a policy *governs* anything, however—schema and tool-name validity are mechanically decidable, encoding a requirement is not—so delivery is reported separately from validity throughout.

Ground truth: intents, not rules. Each workflow ships *golden intents*: annotated statements of one intended requirement, carrying the clause of the specification they trace to, the tool and parameter they govern, a structured statement of the value, threshold, or source that must be right, a severity, and taxonomy tags. An intent describes what must hold, not how a policy should be decomposed: one intent may require several rules together, one rule may satisfy several intents. Static matching is therefore many-to-many, and a submission is not penalized for decomposing differently than the annotator did.

The behavioural unit. A behavioural case is a conversational history—a user prompt plus prior tool calls with their results, some deliberately untrusted—followed by one candidate tool call, labelled allow or block against the *intended* policy. The scorer simulates the *submitted* policy. This asymmetry is the point: if a submission never encoded the rule that should have caught an attack, the

simulation allows the action and the submission is charged a false allow. The scorer never supplies a rule the policy omitted, and never consults the golden intents while deciding.

Representation is scored, not filtered. Structured and natural-language rules are both scored, with counts reported separately (across the 7 systems, 55.4% of 3916 rules used the natural-language kind). Each intent additionally carries an expressibility label—yes, partial, or no—recording whether the requirement can be stated in a restricted structured vocabulary at all. That label is a diagnostic (Section 6) and never removes an intent from a denominator, since excluding hard-to-formalize constraints would make the benchmark easier exactly where synthesis most often fails.

4 Dataset Construction, Curation, and Coverage

Workflow design and inventory. We selected 32 workflows across 8 enterprise domains (four each: ITSM, HR, financial services, healthcare, legal, procurement, customer service, security operations) on one criterion—each must exercise several distinct kinds of policy reasoning—and they share the feature that makes synthesis non-trivial: authorization depends on records retrieved by earlier tool calls, and the authoritative field is frequently not the one a user names. Privileged-access provisioning and emergency clinical override combine approval gating with a bound drawn from two independent sources; offboarding requires ordered, non-skippable steps with irreversible tails; invoice matching requires a computed decision over several records; litigation hold requires completeness and immutability.

The corpus holds 294 tool definitions (272 unique names, 136 side-effecting), 543 golden intents, and 907 behavioural cases (623 attack, 284 benign), joined by 1198 intent-to-case references. System prompts average 481 words; cases carry a mean of 3.4 prior tool calls (maximum 24). Table 4 in Appendix A gives the per-domain breakdown.

Construction pipeline. Every workflow is model-generated and then gated, in the order Figure 3 in Appendix A shows; the artifacts of each stage ship with the release. Generation is targeted rather than open-ended: the taxonomy fixes what a workflow must exercise before any of it is written, so coverage is a constraint on construction rather than a property discovered afterwards. Each stage is generated against the finished artifact before it—the specification first, declaring for each tool the fields a successful call returns, since that declaration is what makes a cross-tool binding expressible at all; then one golden intent per intended requirement, each recording the clause it derives from and together forming the full governance superset rather than the subset a restricted vocabulary could express; then cases as complete fixtures, each internally consistent with the intents it claims to test.

Two gates follow. Structural validation runs over the whole corpus rather than the newly generated workflow alone, and the released corpus passes it on all 96 schema-governed files and 1198 intent-to-case links. Structural agreement cannot establish that an attack case isolates the intent it names, so a semantic gate applies a written rubric to each bundle, and a separate adjudication pass—independent of the pass that authored the bundle, but model-based rather than human—resolves its findings against the specification. This is review independence in the authoring pipeline, not a human audit with measured inter-annotator agreement, which we do not claim.

Conventions. The corpus follows one fixed set of annotation conventions, applied uniformly across all 32 workflows; three of them protect a measurement this paper reports. Binding kind is assigned by a fixed decision procedure—count, then sequence, then completeness, then single-source dataflow, then record scope, then computation, with free-text *prompt* reserved for requirements no structured field could serve—stopping at the first branch that applies, and is kept separate from the expressibility label of Section 3. Every intent restates a requirement the specification already makes and traces to the clause it derives from. Policy is carried by the system prompt alone, tool descriptions carrying mechanics only. A requirement shared across a family of tools is annotated as one intent naming them all, and each case's verdict, disposition, expected firing intent, rationale, and intent cross-references agree in both directions. Appendix A gives the rationale for each convention.

5 EVALUATION PROTOCOL AND SCORER VALIDATION

Static fidelity. Static evaluation labels each golden intent `COVERED`, `PARTIAL`, or `MISSING`, and records `value_correct`: whether the matched rule's threshold, enum, source field, or scope matches the `expected_constraint`. The two interact by construction, because a rule naming the right requirement with the wrong content is labelled `PARTIAL` rather than `COVERED`; value-correctness is carried in the label, and on the released runs no `COVERED` intent is value-incorrect. Our headline fidelity metric is strict coverage—intents covered *and* value-correct, over all golden intents in the workflow—so a rule gating the right parameter against the wrong source, or capping it at the wrong number, does not count. Lenient coverage (covered or partial, ignoring values) is secondary, since the gap between the two separates "the generator noticed this requirement" from "the generator got it right". Symmetrically, each rule is labelled `JUSTIFIED`, `BONUS` (defensible but not required), or `SPURIOUS`; rule precision counts justified over justified plus spurious, excluding bonus rules, and its complement is *spuriousness*. Three outcomes are thus distinguishable and not interchangeable: an omission leaves an action ungoverned, a partial match names the right requirement with the wrong content, a spurious restriction adds one the specification never stated.

Behavioural correctness. Behavioural evaluation simulates the submitted policy on each decision point and emits `allow` or `block` with the rules it considers firing. Per workflow w with A_w attack and B_w benign cases we report FAR_w = false allows$/A_w$, FBR_w = false blocks$/B_w$, and $\mathrm{BA}_w = 1 - \frac{1}{2}(\mathrm{FAR}_w + \mathrm{FBR}_w)$.

Information separation. Static evaluation receives the specification, the golden intents, and the submitted rules, because its job is to check grounding and coverage. Behavioural simulation receives only the submitted rules, tool definitions, and unlabelled decision points, so it cannot reconstruct the ideal policy and a missing rule surfaces as a false allow. Tool schemas resolve field names only; they never add a rejection rule the submission lacked.

Aggregation. The unit of policy generation is a workflow, so it is also the resampling unit: we compute each metric per workflow from its own counts, then take the unweighted mean across the declared workflow set. Intervals are 95% percentile bootstraps over workflows (10,000 resamples, seed 20260921), resampling the same workflow indices for both systems in a difference. Cases are never bootstrapped independently—treating 907 cases as independent samples would badly understate uncertainty, since cases within a workflow are scored by one shared policy.

Failure accounting. Runs fail at two stages, and the protocol fixes three dispositions. A *generation-stage* submission that exhausts the validity budget of Section 3 is rejected and its workflow leaves the denominator, never replaced by a substituted passing score; one that satisfies the contract is scored as it stands even when vacuous, since a policy that governs nothing is a failure to govern rather than a missing measurement. An *evaluation-stage* whole-output judge error surviving the judge's own bounded retry leaves the workflow unscored under that judge rather than penalised, because a scorer that produced no judgement produced no evidence about the policy—which is why runs are only ever compared within one judge configuration. A *per-item* error is corrected deterministically when provable and otherwise scored at its worst case—an unresolved intent as `MISSING`, an unresolved rule as `SPURIOUS`, an unresolved case as the error matching its expected verdict—with the penalty staying in the denominator and the workflow still counted as complete. We also report how clean each score is: 96.0% of 7587 static and 99.5% of 6126 behavioural judgements were reached without a worst-case penalty. Section 6 reports what occurred at each stage.

Scorer validation. Table 9 records what has been validated and over what denominator, keeping apart three things a single reliability figure would merge: the corpus labels, the judgement of what a *submitted* policy implies, and run-to-run stability. Dataset-label evidence comes from structural validation and the documented review of Section 4; it licenses claims about the corpus, not about the judge, and the two are reported separately for that reason. The judge itself is validated directly: every generator's submissions are held fixed and rescored by a judge from a different model family (197 workflow pairs across 7 generators), which agrees at the item level on both views. What that rescoring establishes is a property of each *metric* rather than of the benchmark as a whole. Per-item agreement is high and levels nonetheless move, but the orderings move differently by metric:

Table 1: Main comparison: workflow-macro means with 95% workflow bootstrap intervals (10,000 resamples, seed 20260921). Coverage is strict. **Scored** is the number of workflows entering each mean: rejected submissions are absent from the denominator, vacuous ones are present and scored as they stand (Table 8).

Generator	Scored	Strict cov. (%)	Rule prec. (%)	FAR (%)	FBR (%)	BA (%)
Claude Opus 5	32/32	81.5 [74.5, 87.0]	79.2 [72.2, 84.9]	3.0 [1.8, 4.3]	32.6 [21.6, 44.2]	82.2
Claude Haiku 4.5	26/32	75.8 [70.1, 81.1]	79.4 [74.8, 83.3]	8.4 [5.2, 12.2]	13.2 [6.3, 21.6]	89.2
Gemini 3.1 Pro preview	30/32	75.3 [70.4, 80.3]	88.0 [84.2, 91.7]	4.6 [2.8, 6.5]	17.7 [8.7, 28.2]	88.9
DeepSeek v4.1 Flash	32/32	74.9 [70.5, 79.5]	88.4 [85.2, 91.4]	7.7 [5.3, 10.5]	19.2 [10.2, 29.5]	86.5
GPT-5.4	32/32	74.9 [64.4, 83.8]	78.1 [68.9, 85.5]	14.8 [5.9, 26.1]	13.7 [7.1, 21.7]	85.7
Gemini 3.5 Flash Lite	32/32	58.6 [52.3, 64.5]	91.9 [88.5, 95.0]	15.5 [11.6, 19.8]	11.6 [4.5, 20.3]	86.5
GPT-4.1 mini	32/32	51.7 [45.6, 57.7]	83.5 [79.2, 87.4]	18.3 [14.3, 22.7]	15.8 [8.7, 23.6]	82.9

strict coverage ($\rho = 0.82$), rule precision and the false-allow rate ($\rho = 0.96$) support ranking across judges, whereas the false-block rate and balanced accuracy are reportable against a named judge. That is the reporting rule this protocol imposes—and the reason fidelity, FAR, and FBR are carried side by side rather than collapsed into one composite. Generator stability is measured the same way, by regenerating two configurations from scratch under the pinned judge (64 workflow pairs). Section 6 sets both sources of movement against the spread the benchmark is asked to resolve (Table 2).

6 Experiments and Diagnostic Results

Setup. We evaluate 7 generator configurations—Claude Opus 5, Claude Haiku 4.5, Gemini 3.1 Pro preview, DeepSeek v4.1 Flash, GPT-5.4, Gemini 3.5 Flash Lite, GPT-4.1 mini—driven by the same reference prompt on all 32 workflows and scored by one pinned judge (`openai/gpt-5.4`). None of the three synthesis systems of Section 2 is publicly runnable on arbitrary agent specifications and their input contracts differ from ours, so the reference baselines are LLM generators under one shared prompt; the benchmark accepts a submission from any system meeting the contract of Section 3. Of these, 3 failed to deliver a policy on at least one workflow; we report them with their delivery counts rather than dropping them, treating non-delivery as a measured property of the generator (Table 8). On the 25 workflows all 7 completed, no system's strict coverage moves by more than 2.7 points, the ordering holds up to a three-way tie within 0.4 points ($\rho = 0.86$), and the coverage leader is still last on balanced accuracy—so neither the comparison nor the inversion rests on which workflows each system completed.

RQ1: fidelity and utility do not co-vary. Table 1 shows a 29.8-point spread in strict coverage, from 51.7% (GPT-4.1 mini) to 81.5% (Claude Opus 5), with non-overlapping intervals at the extremes. FAR orders the systems broadly the same way—Claude Opus 5 misses 3.0% of attacks, six times fewer than the weakest-coverage system. FBR breaks that ordering hardest at the top: Claude Opus 5 blocks 32.6% of legitimate actions, the worst of the 7 by thirteen points, against 15.8% for GPT-4.1 mini. Balanced accuracy therefore nearly stops separating the field—7.0 points best to worst, every paired interval containing zero—and *reverses* at the top: the coverage leader ranks 7*th* of 7 on balanced accuracy. Neither view is complete alone, and the inversion is itself judge-conditional: judge B is uniformly more permissive on FBR (-11.8 points) and compresses the field, and under it Claude Opus 5 ranks second of 7 on the workflows judge B scored rather than last. The trade-off is a within-judge finding about these submissions—which is why FBR and balanced accuracy are reported against a named judge and never used to rank across judges ($\rho = 0.21$).

The mechanism is visible in the submissions: Claude Opus 5 writes 741 rules and GPT-5.4 664, against 413 and 421 for the two weakest. More rules buy coverage and cost precision—Claude Opus 5 has the highest coverage and nearly the lowest—though the audit below shows its extra false blocks fire mostly on *grounded* rules applied at the wrong decision point. The natural-language share varies as widely, from 42.0% (Gemini 3.1 Pro preview) to 68.3% (Claude Haiku 4.5), but tracks coverage more weakly than rule count does; RQ2 separates the two.

Failure accounting: two stages, two dispositions. Failures arise at the two stages Section 5 charges to different parties (Table 8).

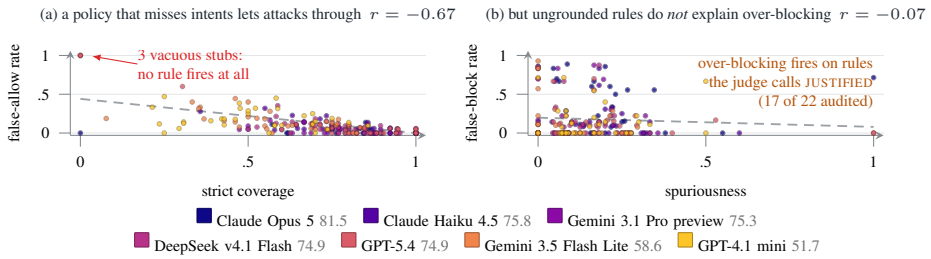


Figure 2: **What each view predicts.** Colour identifies a generator, ordered by strict coverage (dark → light, values in the legend). One point per workflow per generator (216 points), with the least-squares trend dashed; the three points at $(0, 1)$ in (a) are the vacuous stub submissions of Table 8. Static fidelity predicts under-blocking, but its mirror image does not predict over-blocking. Both panels describe associations on this corpus, not causal effects.

Generation-stage failures belong to the system under test. Of 224 scheduled generations, 8 were rejected for contract violations surviving the repair budget, concentrated in 2 configurations. A second mode escapes the contract entirely: a submission can be schema-valid, name only declared tools and fields, and still say nothing. In all, 4 submissions—all from GPT-5.4, ranging from an empty rule list to a single rule literally named `placeholder`—passed every validity check and were scored, and 3 then allowed every attack on workflows declaring 12 to 21 intents. Keeping them in the denominator is what separates validity from delivery: GPT-5.4 scores 74.9% coverage and 14.8% FAR overall against 84.5% and 4.1% on the workflows where it actually delivered a policy, and reporting only "schema-valid submissions accepted" would collapse the two.

Evaluation-stage failures belong to the harness. Under the pinned judge, 0 of 216 evaluations failed with a whole-output judge error; under the second-family judge, 19 did (8.8%), almost all structural and surviving the judge's own retry. Left unscored rather than penalised, these are why the cross-judge study below compares only workflows both judges scored. Worst-case per-item penalties are rarer but not free: 98 of 216 scored workflows contained at least one, and in 1 the entire static judgement collapsed this way.

RQ2: constraint structure, not expressibility, tracks difficulty. The corpus keeps two axes apart, and they behave very differently (Tables 5 and 6). Pooled across systems, strict coverage by binding kind ranges from 50.6% (completeness) to 76.8% (prompt), a 26.2-point spread; by expressibility the spread is 9.9 points (73.1%/69.1%/63.2% for yes/partial/no), with the two largest buckets four points apart despite being the ends of a supposed difficulty gradient. Which binding kind is hardest varies by generator (`completeness` for 5 systems, `scope` for the other two), but the effect is large for every one—at least 19.7 points—whereas only 3 of the 7 order the expressibility buckets monotonically at all. Two findings run against intuition: `completeness` constraints are hardest despite being conceptually simple, because a policy of per-action gates has no natural place to say an action must *eventually* occur; and `prompt` constraints, having no structured field to compare against, are nonetheless best covered because the natural-language rule kind expresses them directly. Cross-tabulation settles the confound: within the `yes` bucket alone, strict coverage still spans 60.8–91.2%, so filtering a benchmark to "executable" intents would measure not an easier version of this task but a differently composed one (Table 6).

The prose escape hatch is not a loophole. A contract that accepts a natural-language constraint invites the objection that any requirement can be satisfied by restating it. Joining every submitted rule to its judged label shows otherwise: prose rules are no likelier to be `SPURIOUS` than structured ones (82.4% against 83.9% rule precision), and the 6.3-point coverage penalty on the intents they absorb is small beside the 26.2-point spread across binding kinds (Appendix B).

RQ3: fidelity predicts under-blocking, and over-blocking has a different cause. Strict coverage and FAR are strongly negatively associated across workflows ($r = -0.67$, $\rho = -0.64$, $n = 216$; Figure 2a), and the linkage annotations sharpen this beyond a correlation: of 435 false allows whose case names an expected firing intent, 397 (91.3%) occurred where the *same* policy also failed to cover that intent strictly. False allows are overwhelmingly the downstream consequence of an identifiable static defect, the stub submissions being the limiting case. Spuriousness and FBR show no

Table 2: **Top:** frozen submissions rescored by a second-family judge (A = `openai/gpt-5.4`, B = `anthropic/claude-opus-5`), over the workflows both judges scored; the same frozen submissions were rescored. **Bottom:** between-system signal against both noise sources (percentage points): *system spread* is best minus worst in Table 1, *judge shift* the mean level change under judge B, *run shift* the largest paired macro change across the 2 regenerations of Table 7. A metric supports cross-judge ranking when its spread dominates both and its ordering survives the swap; strict coverage, rule precision, and FAR meet that bar.

Generator	Wf.	Verdict agr.	Strict agr.	Coverage A/B	FBR A/B	BA A/B
Claude Opus 5	29	90.3%	87.3%	81.1 / 93.0	31.0 / 5.8	83.0 / 95.2
Claude Haiku 4.5	23	93.9%	88.3%	74.9 / 84.1	14.3 / 5.7	88.5 / 93.2
Gemini 3.1 Pro preview	29	92.3%	85.7%	75.3 / 86.1	17.8 / 4.0	88.9 / 95.5
DeepSeek v4.1 Flash	30	91.9%	89.7%	75.0 / 81.2	20.5 / 4.4	85.6 / 93.3
GPT-5.4	30	95.3%	92.0%	76.7 / 83.0	12.4 / 5.4	87.5 / 90.9
Gemini 3.5 Flash Lite	27	91.9%	87.8%	59.6 / 65.5	9.6 / 2.8	87.1 / 92.1
GPT-4.1 mini	29	92.1%	82.5%	52.1 / 63.3	14.9 / 9.7	84.1 / 85.9
Rank corr. across judges	197	–	–	$\rho = 0.82$	$\rho = 0.36$	$\rho = 0.21$

Metric	System spread	Judge shift	Run shift	Spread/run	Judge rank ρ
Strict coverage	29.8	8.8	1.4	21.5	0.82
Rule precision	13.8	4.9	1.1	12.5	0.96
False-allow rate	15.4	0.0	2.1	7.3	0.96
False-block rate	21.0	11.8	7.6	2.8	0.36
Balanced accuracy	7.0	5.9	2.7	2.5	0.21

such relationship ($r = -0.07$, $\rho = -0.03$; Figure 2b): over-blocking is not explained by counting ungrounded rules. The audit explains why—17 of 22 audited false blocks fire on a rule the static judge labels `JUSTIFIED`, faithfully restating the specification but encoded as an unconditional precondition at the wrong decision point, or as a base restriction without the override the specification also states.

Audited error categories. We manually categorized 51 behavioural failures, sampled proportionally across four of the 7 configurations and across outcome types, against a taxonomy fixed before coding (Table 12). Omission dominates (25/51) and is almost always total: the rationale reads "no submitted rule targets X". Read-only lookup tools are a recurring blind spot—policies gate the dramatic writes and leave the retrieval path feeding them ungoverned, which is exactly how cross-record enumeration succeeds.

Stability: what a score survives. We measure two sources of instability and compare both with the spread the benchmark is asked to resolve (Table 2).

Judge. Rescoring all 197 frozen submissions of all 7 generators with a second-family judge gives high per-item agreement: 92.5% of 5585 behavioural verdicts and 87.6% of strict-coverage decisions. Levels nonetheless move systematically—more generous statically ($+8.8$ points) and markedly more permissive behaviourally (-11.8 points of FBR), while agreeing on FAR almost exactly—and the orderings behave as Section 5 requires: 3 of the 7 systems keep their coverage rank.

Generation. Two configurations were regenerated from scratch and rescored under the pinned judge (64 workflow pairs, 0 identical—temperature zero is not a determinism guarantee). Macro coverage moves by at most 1.4 points and balanced accuracy by 2.7, but the composition of that stability differs sharply. For Claude Opus 5 the policies are stable and the residual movement is behavioural: FBR shifts 7.6 points, larger than the entire balanced-accuracy spread of Table 1. This locates generator nondeterminism precisely. The *artifact* barely changes, but its *behavioural consequence* does, because over-blocking turns on how a grounded rule is conditioned rather than on whether it was written—the same mechanism the audit finds, with 17 of 22 audited false blocks firing on rules the static judge calls `JUSTIFIED`. Under the pinned judge the direction is stable—Claude Opus 5 has the worst FBR in both regenerations—while the magnitude is not, which is why FBR is reported beside FAR instead of averaged with it. For GPT-5.4 the macro means barely move (74.9%→75.8%) while individual workflows move by 24.0 points on average, because the 4 and 3 stub submissions land on *different* workflows each run; conditioned on a policy actually being delivered the two runs agree closely (coverage 84.5%/83.7%). Run-to-run variation in this system's

headline score is therefore not variation in policy quality but in *whether a policy is produced at all*—the reason Table 8 is reported beside Table 1.

Signal against noise. The bottom panel of Table 2 sets both noise sources against the spread each metric must resolve. Strict coverage separates systems by 21.5 times its run-to-run shift and FAR by 7.3, while FBR manages 2.8 and balanced accuracy 2.5—the composite inheriting the sensitivity of the over-blocking term it averages rather than creating it.

RQ4: the two views are sensitive to different perturbations. Holding the scorer fixed and rewriting what the generator reads—paraphrase, decoy history, and a combined variant with injective relabelling, over 8 pre-registered workflows—leaves behaviour invariant: on FAR, FBR, balanced accuracy, and coverage every paired interval contains zero (Table 10). Deciding what to allow and what to block is therefore independent of tool names, identifier strings, and prompt wording, evidence that the corpus measures synthesis rather than lexical pattern-matching. Rule precision is the exception, moving under all three variants with intervals excluding zero while coverage stays intact: nothing required is dropped, the generator simply writes more rules it cannot ground. Read beside the stability study this inverts—a change of judge moves the behavioural view and leaves fidelity intact, a change of input moves the static view and leaves behaviour intact—so each view is the sensitive instrument for a different kind of drift (Appendix B).

7 LIMITATIONS

The workflows are model-generated and 32 workflows in 8 domains is a small sample, so intervals are correspondingly wide and the difficulty distribution reflects generation and curation choices. Grounding does not imply executability. Neither the contract nor the static judge demonstrates that a prose rule can be executed by a deterministic engine. The judge is an LLM, so absolute numbers are judge-conditional. Repeat generations cover two of 7 configurations and the perturbation suite one. Finally, the behavioural score is a semantic judgement about what the submitted policy implies, scored one next action at a time: offline artifact evaluation, not a study of deployed enforcement.

8 CONCLUSION

SymPolicyBench releases 32 enterprise agent workflows, 543 source-traced golden intents, and 907 linked decision points, with a protocol that freezes a synthesized policy and scores it twice: against the requirements it should encode, and on decisions it must make with its omissions intact.

Measuring 7 generators, rescoring every frozen submission with a second-family judge, and regenerating two configurations yields three results. Fidelity and decision accuracy are not substitutes—strict coverage separates the field by 29.8 points against balanced accuracy's 7.0, and under the pinned judge the two order the top of the field oppositely. The two views also fail differently: a missing intent predicts the false allow it causes (91.3% of false allows sit on an uncovered intent), whereas over-blocking arises from grounded rules conditioned at the wrong decision point and is not predicted by counting ungrounded ones. And the dominant defect is omission rather than error—the most common failure is that no submitted rule targets the requirement at all.

The protocol is explicit about which of its own numbers travel. Strict coverage, rule precision, and the false-allow rate rank systems consistently under a judge from a different model family ($\rho = 0.82$–0.96); the false-block rate and balanced accuracy do not, and are reportable only against a named judge. A developer therefore learns which constraints their generator drops, which legitimate actions it obstructs, how often it delivers nothing at all—and, for each of those, whether the finding would hold under a different scorer.

AI USE STATEMENT

Generative AI tools were used in ways that are material to this work and are disclosed here in full. The benchmark corpus is *model-generated*: the agent specifications, golden intents, and behavioural cases of all 32 workflows were produced by large language models against a taxonomy fixed before authoring, as described in Section 4. Section 5 and Table 2 quantify the resulting judge-conditionality, and the cross-family rescoring of the same frozen submissions is the validity evidence

we offer in place of human adjudication. All generation, judging, and review prompts, the models and settings used, and the full run logs ship with the release (Appendices B and C).

Generative AI tools were also used as a general-purpose assistant for writing and editing the manuscript and for routine coding support in the analysis and plotting scripts. Every reported quantity is recomputed deterministically from the frozen artifacts by the released scripts, which make no model calls, so no number in this paper originates from an unverified model output. The authors have reviewed all AI-assisted work and take responsibility for the final content of this work, including its text, claims, and artifacts.

Ethics statement

The corpus contains no real personal data: identifiers, names, addresses, and account numbers in the fixtures are synthetic, several embedded precisely so that a policy's PII-minimization rules can be tested. Attack cases describe governance-bypass patterns at the level of a single decision point—which record a parameter should be drawn from, whose approval counts, which step may not be skipped—and contain no exploit code; the same descriptions are what a defender needs to specify a guardrail. The intended use is research on policy synthesis and its evaluation. The data documentation, intended use, and licensing are recorded in Appendix C. No human subjects were involved and no personal data was collected or processed.

Reproducibility statement

Every number in this paper is recomputed from the frozen artifacts by the scripts released with it; none requires a model call. The corpus, the four JSON schemas, the reference generator and evaluator, the exact submissions and per-workflow reports for all evaluated configurations, and the analysis scripts ship together (Appendix C). The pinned generation and judging settings are given in Appendix B, the aggregation rule and bootstrap seed in Section 5, and the error-coding protocol and its per-item labels in Appendix B. Reproducing the analysis from the shipped reports needs only the Python standard library.

References

Adewale Akinfaderin and Shreyas Subramanian. VERAFI: Neurosymbolic policy generation for financial agent compliance. *arXiv preprint arXiv:2512.14744*, 2025.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, Eric Winsor, Jerome Wynne, Yarin Gal, and Xander Davies. AgentHarm: A benchmark for measuring harmfulness of LLM agents. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.09024.

Sam Bayless, Byron Cook, et al. A neurosymbolic approach to natural language formalization and verification. *arXiv preprint arXiv:2511.09008*, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.13352.

Deepali Kholkar and Kavita Ahuja. Policy-as-prompt: Turning unstructured governance documents into enforceable guardrails. *arXiv preprint arXiv:2509.23994*, 2025.

Ido Levy, Ben Wiesel, Sami Marreed, Alon Oved, Avi Yaeli, and Segev Shlomov. ST-WebAgentBench: A benchmark for evaluating safety and trustworthiness in web agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2410.06703.

Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2009.

Michael Mondl, Stefan Maisel, and Harald Brock. Autoformalization of agent instructions into policy-as-code. *arXiv preprint arXiv:2606.26649*, 2026.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2309.15817.

Lillian Tsai and Eugene Bagdasarian. Contextual agent security: A policy for every purpose. In *Workshop on Hot Topics in Operating Systems (HotOS)*, 2025. arXiv:2501.17070.

Aryan Vatsa, Benjamin Hall, and William Eiers. PolicySummarizer: Verifiable natural-language characterization of access control policies. *arXiv preprint arXiv:2510.20692*, 2025.

Aryan Vatsa, Arush Shome, Yuchen Zhou, and William Eiers. AutoCedar: An agentic framework for verifier-guided access control policy synthesis. *arXiv preprint arXiv:2607.03656*, 2026.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. *arXiv preprint arXiv:2406.09187*, 2024.

Zhongyuan Xu and Scott D. Stoller. Mining attribute-based access control policies from logs. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*, 2013.

Yifan Xue, Wei Chen, and Lin Zhang. RAFT: Regulatory requirement autoformalization with multi-LLM aggregation. *arXiv preprint arXiv:2601.09762*, 2026.

Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. R-Judge: Benchmarking safety risk awareness for LLM agents. In *Findings of the Association for Computational Linguistics (EMNLP)*, 2024. arXiv:2401.10019.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2403.02691.

Hanrong Zhang, Jingyuan Huang, Kai Mei, Yifei Yao, Zhenting Wang, Chenlu Zhan, Hongwei Wang, and Yongfeng Zhang. Agent security bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. *arXiv preprint arXiv:2410.02644*, 2024.

A Corpus Schemas, Taxonomy, and Construction Records

Table 3: Positioning relative to the closest prior art. We do not claim to be first to evaluate a synthesized policy; we claim no public resource supplies cross-domain agent specifications together with decomposition-independent intents and linked decision-level tests.

	Input	**Artifact**	**Ground truth**	**Evaluation**
AgentDojo, ASB	task + attack	none (agent acts)	env. state	trajectory
ST-WebAgentBench	task + given policy	none (agent acts)	policy predicates	trajectory
Conseca	task, context, API docs	per-request policy	case study	qualitative
Mondl et al.	prompt + tool docs	Cedar policy	one manual policy	engine decisions
AutoCedar	requirements doc	Cedar policy	target policy	verifier equivalence
SymPolicyBench	prompt, trust model, tools	**one reusable policy**	**intents + cases**	**static + behavioural**

A.1 File layout and schemas

Each workflow directory contains `agent.json` (the system prompt, trust model, and tool definitions), `golden_intents.json` (the annotated ground truth), `test_cases.json` (labelled decision points), and `happy_path_prompts.json` (illustrative normal-usage prompts that are never scored and never shown to a generator). Four JSON Schema files define every field, and `validate.py` enforces them together with bidirectional intent↔case consistency and portfolio-level taxonomy coverage. Tool definitions carry two documentation-only fields: `x_returns`,

which names the fields a successful call returns and is what makes a cross-tool binding expressible, and `x_side_effecting`, which marks irreversible actions.

A.2 Corpus inventory

Table 4 gives the per-domain composition of the corpus, referenced from Section 4.

Table 4: Corpus inventory by domain. The binding-kind and expressibility distributions of the 543 golden intents are in Table 5, and their full cross-tabulation in Table 6.

Domain	Workflows	Tools	Intents	Attack	Benign
CSM	4	42	80	85	42
Financial services	4	39	69	76	34
Healthcare	4	31	64	69	32
HR	4	40	81	94	42
ITSM	4	43	79	86	34
Legal	4	32	53	68	27
Procurement	4	36	66	76	43
SecOps	4	31	51	69	30
Total	32	294	543	623	284

A.3 Construction pipeline

Figure 3 shows the order in which each workflow was authored and gated, referenced from Section 4.

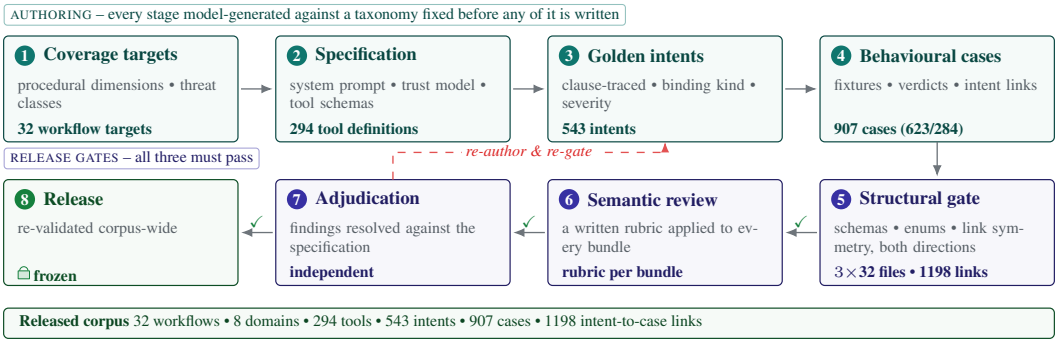


Figure 3: **How one workflow enters the corpus.** Stages 1–4 are model-generated, each constrained by the artifact before it: intents are written against a finished specification, and cases against finished intents, so a case can never test a requirement the specification does not state. Stages 5–7 are gates a workflow must clear before stage 8 ships it; a gate finding sends the workflow back to be re-authored (dashed) rather than dropped.

A.4 Annotation conventions and their rationale

The decision procedure for binding kind stops at the first branch that applies, so the difficulty axis of Section 6 reflects a stated rule rather than case-by-case judgement. Binding kind is kept separate from the expressibility label because the first is a property of the constraint and the second of a representation; conflating them would hide whether a generator fails because a requirement is structurally hard or because its chosen vocabulary cannot say it. A constraint the specification does not state is not annotated, since rewarding the recovery of unstated policy would measure taste in governance instead of translation fidelity. Tool descriptions carry mechanics only, so that policy cannot be recovered from tool documentation rather than inferred from the instructions. Bidirectional intent↔case agreement is what makes the linkage analysis of Section 6 trustworthy.

A.5 BINDING KINDS

`dataflow`: a parameter must match or derive from another tool's output field. `scope`: the governed record, tenant, or identity is fixed by a single authoritative tool field. `computation`: a business-rule threshold or computed value, such as a minimum over two bounds or a cumulative sum. `sequence`: steps must occur in a required order. `completeness`: a mandatory step must always execute, or an abort condition must be honoured. `count`: a call-count or rate limit. `prompt`: a condition on free text with no structured counterpart—the user's own utterance, or a note or message the agent itself writes.

A.6 STRICT COVERAGE CROSS-TABULATION

Table 6 gives strict coverage for every binding-kind × expressibility cell, pooled over all 7 systems, with the number of intent judgements in each cell. Empty cells have no intents in the corpus.

Table 5: Marginal distributions of the 543 golden intents by binding kind (left) and canonical-DSL expressibility (right), referenced from Section 4.

Binding kind	dataflow	computation	sequence	scope	prompt	completeness	count
Intents	246	83	50	46	44	37	37

Canonical-DSL expressible	yes	partial	no
Intents	246	256	41

Table 6: Pooled strict coverage (%) by binding kind and canonical-DSL expressibility, with cell denominators in parentheses. Pooled over all 7 systems, so each corpus intent contributes up to 7 judgements.

Binding kind	**expressible: yes**	**partial**	**no**
dataflow	75 (1245)	71 (420)	–
computation	74 (38)	75 (305)	63 (218)
sequence	91 (68)	69 (255)	36 (14)
scope	61 (240)	61 (66)	33 (6)
prompt	–	76 (262)	83 (35)
completeness	–	51 (242)	43 (7)
count	61 (76)	74 (174)	–

B PROTOCOL DETAILS

B.1 SCORER AND GENERATOR VALIDITY EVIDENCE

The four tables below record run-to-run stability, two-stage failure counts, the validity evidence behind each claim, and the perturbation results, each with its own denominator.

Table 7: Two independent regenerations of the same configuration under the pinned judge, paired workflow bootstrap; "delivered only" excludes stub submissions. Referenced from Section 6.

System	**Metric**	**Run 1**	**Run 2**	**Δ (paired)**	**Delivered only**
GPT-5.4	Strict coverage	74.9	75.8	0.9 [-12.9, 14.9]	84.5 / 83.7
	False-allow rate	14.8	13.6	-1.2 [-15.9, 13.6]	4.1 / 4.7
	False-block rate	13.7	18.9	5.1 [-2.8, 14.0]	15.7 / 20.8
	Balanced accuracy	85.7	83.8	-1.9 [-9.6, 5.9]	90.1 / 87.3
Claude Opus 5	Strict coverage	81.5	82.9	1.4 [-4.5, 9.3]	81.5 / 82.9
	False-allow rate	3.0	5.1	2.1 [0.5, 4.3]	3.0 / 5.1
	False-block rate	32.6	25.0	-7.6 [-15.9, 0.0]	32.6 / 25.0
	Balanced accuracy	82.2	84.9	2.7 [-1.1, 6.9]	82.2 / 84.9

Table 8: Two-stage failure accounting, referenced from Section 6. *Rejected*: still contract-invalid when the repair budget was exhausted, so no policy exists and the workflow leaves the denominator. *Stub*: contract-valid but vacuous (≤ 2 rules), scored as it stands. *Judge A/B*: workflows whose whole-output judgement never validated, under each judge. *Collapsed*: scored workflows in which every intent was forced to `MISSING`.

	Generation stage				Evaluation stage		
Generator	Sched.	Rejected	Stub	**Delivered**	Judge A	Judge B	Collapsed
Claude Opus 5	32	0	0	32	0	3	1
Claude Haiku 4.5	32	6	0	26	0	3	0
Gemini 3.1 Pro preview	32	2	0	30	0	1	0
DeepSeek v4.1 Flash	32	0	0	32	0	2	0
GPT-5.4	32	0	4	28	0	2	0
Gemini 3.5 Flash Lite	32	0	0	32	0	5	0
GPT-4.1 mini	32	0	0	32	0	3	0
Total	224	8	4	212	0	19	1

Table 9: Validity evidence for the corpus labels, the scorer, and the generators, each row with its own denominator. †Rescored from the same frozen submissions across all 7 generators. ‡Independent regenerations of the same configuration under the pinned judge. Referenced from Section 5.

Check	**Denominator**	**Result**
Structural validation of every schema-governed file (`validate.py`)	96/96 files	0 errors
Intent↔case back-references agree both ways	1198 links	0 mismatches
Documented semantic review of every workflow	32 workflows	209 findings, adjudicated
Golden intents never exercised by an attack case	10/543 intents	disclosed defect
Cross-judge behavioural verdict agreement†	5585 cases	92.5%
Cross-judge static label agreement†	3340 intents	85.2%
Cross-judge strict-coverage agreement†	3340 intents	87.6%
Cross-judge ranking, strict coverage†	7 systems	$\rho = 0.82$, 3/7 ranks held
Cross-judge ranking, balanced accuracy†	7 systems	$\rho = 0.21$, 1/7 ranks held
Repeat-generation shift, strict coverage‡	2 run pairs	$\leq$1.4 pt
Repeat-generation shift, balanced accuracy‡	2 run pairs	$\leq$2.7 pt
Repeat-generation verdict agreement‡	1814 cases	87.2%
Static judgements without a forced penalty	7587 intents + rules	96.0%
Behavioural judgements without a forced penalty	6126 cases	99.5%

B.2 Representation and the natural-language rule kind

The prose escape hatch is not a loophole. Expressibility is a property of the requirement; the mirror question is what the *submission* does with the freedom to answer in prose, since a contract that accepts a natural-language constraint invites the objection that any requirement can be satisfied by restating it. Joining every submitted rule to its judged label shows otherwise. Natural-language rules are no likelier to be ungrounded than structured ones—17.6% of 2170 are `SPURIOUS` against 16.1% of 1746, a rule precision of 82.4% against 83.9%—so prose neither buys credit nor attracts a penalty of its own. What it does buy is compression: a natural-language rule matches more than one golden intent 26.8% of the time against 15.0% for a structured rule (1.26 intents per rule against 1.07), and the intents it absorbs are strictly covered 78.6% of the time ($n = 1844$) against 85.0% for intents matched only by structured rules ($n = 1052$). That 6.3-point penalty is real but small beside the 26.2-point spread across binding kinds: how a requirement is *stated* matters far less than what it *is*. Prose is therefore not a cheaper route to credit; whether a prose rule is mechanically enforceable is a separate question, taken up in the Limitations.

B.3 Perturbation transforms

Paraphrase rewords the system prompt and trust model, preserving every threshold, tool name, and prohibition. *Decoy history* injects one realistic but non-binding prior tool call into each case, borrowed from a read-only tool that no rule can count or bind to; one workflow admits no such tool and is left unperturbed, so that variant perturbs 7 of 8. *Combined* applies both together with a consistent injective relabelling of tool names and entity identifiers. Each transform is checked mechanically—a

Table 10: Meaning-preserving perturbation of the input the generator reads (GPT-5.4, pinned judge, 8 pre-registered workflows). Labels are invariant by construction, so the reference row is the same configuration on unperturbed inputs. Δ prec. is the paired workflow bootstrap of reference minus variant with its 95% interval. Referenced from Section 6.

Input	FAR (%)	FBR (%)	BA (%)	Cov. (%)	Rule prec. (%)	Δ prec.
Reference	7.3	25.8	83.4	85.6	83.8	–
Paraphrase	2.7	18.7	89.3	87.6	77.5	**6.3** [3.1, 9.2]
Decoy history	5.1	30.4	82.3	81.0	79.1	**4.6** [0.7, 8.8]
Combined	6.7	28.3	82.5	81.4	75.0	**8.7** [1.2, 17.3]

constraint diff for the paraphrase, a no-dangling-reference assertion for the relabelling. Workflows were selected by a structure-only rule (most distinct binding kinds, ties broken toward the most intents).

B.4 Pinned settings

Generation: temperature 0, 32000 maximum output tokens, one run per workflow, bounded feedback-guided repair for provable contract violations only. Judging: temperature 0, 32000 maximum output tokens, consistency review enabled with the same judge, bounded repair for static and behavioural stages, no case limit. Every report records the agent file, golden intents, test cases, taxonomy, submission, prompts, and schemas it was produced from, and a comparison is reported only across rows sharing one corpus and one judge configuration.

Aggregation is a downstream reporting step, not an output of the per-workflow CLI. The aggregation used in this paper computes workflow-macro means and paired workflow bootstraps with seed 20260921, as the protocol requires, and is recomputed from the shipped per-workflow reports by the released analysis scripts. The convenience `aggregate.json` files bundled with each run instead pool case-weighted rates, which is a different estimator of a different quantity rather than a different result; the protocol-compliant workflow-macro values are the ones reported throughout.

B.5 Paired differences against the strongest system

Differences are resampled on matched workflow indices, so each row is a paired comparison rather than a difference of independent means.

Table 11: Paired workflow-bootstrap differences relative to Claude Opus 5 (percentage points; positive means Claude Opus 5 is higher). The same workflow indices are resampled for both systems in each pair.

Compared system	n	Δ strict coverage	Δ FAR	Δ balanced acc.
Claude Haiku 4.5	26	5.6 [-4.2, 13.6]	-5.1 [-8.2, -2.3]	-6.3 [-12.6, 0.2]
Gemini 3.1 Pro preview	30	7.6 [-1.2, 14.7]	-1.8 [-3.7, 0.0]	-6.2 [-12.5, 0.5]
DeepSeek v4.1 Flash	32	6.6 [-0.5, 12.8]	-4.8 [-7.5, -2.4]	-4.3 [-10.3, 1.5]
GPT-5.4	32	6.6 [-5.2, 18.7]	-11.8 [-23.0, -2.9]	-3.5 [-10.5, 3.9]
Gemini 3.5 Flash Lite	32	22.9 [14.3, 30.1]	-12.5 [-16.8, -8.6]	-4.3 [-9.8, 1.0]
GPT-4.1 mini	32	29.8 [22.8, 36.2]	-15.4 [-19.7, -11.4]	-0.7 [-6.6, 5.2]

B.6 Error-coding protocol and audited categories

The audited sample, its stratification, and the coding scheme are described below; Table 12 gives the resulting category counts.

The 51 audited failures were drawn proportionally from each system-by-outcome stratum with seed 20260921, and coded against a taxonomy fixed in advance: omission, wrong source or record, wrong threshold, ordering or state error, and invented restriction. Coding used the judge's recorded rationale, the firing rules and their static labels, the linked golden intent and its static label, and the case fixture. A separate flag records penalties attributable to the protocol's worst-case handling of unresolved items or to abbreviated fixtures. The sample, the coding scheme, and the per-item labels are released alongside the analysis scripts.

Table 12: Manual categorization of 51 stratified behavioural failures across four of the 7 configurations (Claude Opus 5, DeepSeek v4.1 Flash, Gemini 3.5 Flash Lite, GPT-4.1 mini). "Conservative" counts penalties attributable to worst-case handling of an unresolved item or to an abbreviated fixture, rather than to a defect in the submitted policy.

Category	**Total**	**False allows**	**False blocks**	**Conservative**
Omission	25	25	0	1
Wrong source/record	3	1	2	1
Wrong threshold	6	2	4	0
Ordering/state error	11	1	10	2
Invented restriction	6	0	6	0
Total	51	29	22	4

C Artifact Release and Data Documentation

The corpus contains no real personal data: identifiers, names, addresses, and account numbers in the fixtures are synthetic, several embedded precisely so a policy's PII-minimization rules can be tested. The intended use is research on policy synthesis and its evaluation.

The release package contains the dataset, the four JSON schemas, the reference generator and evaluator implementations, the aggregation and analysis scripts used to produce every number in this paper, the exact submissions and per-workflow reports for every evaluated configuration—the 7 primary runs, the 7 cross-judge rescorings of the same frozen submissions, and the 2 independent regenerations—together with the run logs and the structured failure records for every rejected generation and every failed evaluation. Verification, statistics, and the released tests run with the Python standard library.