COUNTEREXAMPLE-GUIDED SYNTHESIS OF SKILL APPLICABILITY CONTRACTS WITH LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents reusing tool-calling skills need to know when they succeed and how they change the environment. We synthesize such applicability contracts from a supplied DSL and execution feedback, with the reference contract withheld. Our counterexample-guided inductive synthesis (CEGIS) loop partitions reachable states by the proposed contract's accept/reject predictions, executes the skill on both sides, and revises the contract from structured counterexamples.

We evaluate synthesized contracts exhaustively over four core domains, three synthetic and one built on third-party tool code; a run counts as exact if its final contract is correct on every reachable state. Gemma-4-31B is exact in 20 of 20 seeds on Deployment and Calendar, against 0 and 3 for direct generation. On a finite τ-bench retail fixture it is exact in 19 of 20, against 0 for direct generation and 3 for fixed-pool refinement. A common-loop factorial study of selection and termination retains the partitioning advantage for Gemma under both stopping rules after multiplicity correction; Qwen3.8 shows weaker, non-significant differences. Relational coverage has mixed effects across conditions. A Shopping difficulty study distinguishes recovery on a deliberately relaxed variant from failure when hidden rules become independently identifiable. Muse-Glimmer-30B is also exact in 20 of 20 seeds on Deployment and Calendar under its native low-reasoning setting. For local models, evidence chosen without the candidate fails mostly by false accepts; in local and hosted cohorts alike, the partitioned loop's residual failures are mostly false rejects hidden in the candidate's large predicted-reject region.

1 INTRODUCTION

An agent using a tool-calling skill must know when it succeeds and what state change it produces: its *applicability contract*. Prior work encodes applicability conditions, avoidance rules, repairable pre/postconditions, or logic-grounded control flow (Lin et al., 2026; Lu et al., 2026; Shao et al., 2026), typically from authored specifications or observed trajectories. We study inferring a withheld contract through new executions.

Our question is: **can an LLM synthesize a closure-exact applicability contract from execution feedback and a supplied DSL?** Several domains supply named valid sets, and Calendar and retail additionally expose clause-level predicates. The model sees neither the reference contract nor the environment source. It observes execution outcomes; a separate evaluator checks its final contract on every reachable state.

We formulate this as counterexample-guided inductive synthesis (CEGIS) (Solar-Lezama et al., 2006). Prior specification synthesis uses formal verification (Leite et al., 2026) or execution-based checks (Yamamoto & Katsurai, 2026). Our current candidate partitions reachable states by predicted acceptance; executions on both sides test its predictions, and structured before/after differences guide revision. Recomputing the partition makes execution selection responsive to each candidate.

Contributions:

- An execution-only formulation with evaluator-only full-closure ground truth across four core domains (Section 5).

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.

- A candidate-partitioned selector, evaluated in a common-loop factorial study of selection and termination (Section 4).
- Controlled comparisons and failure diagnostics separating counterexample exposure, revision, and validation, with paired tests and Holm correction (Sections 6–7; Appendix T.1).

Across six- to ten-clause references, candidate partitioning is the selector component that matters: for the original models, removing it lowers exact recovery wherever the full selector recovers any contract.

2 Related work

Skills and applicability. Agent libraries store executable skills (Wang et al., 2024; 2025a), workflow memories (Wang et al., 2025b), or trace-derived logic (Shao et al., 2026). ContractSkill repairs pre/postconditions (Lu et al., 2026); SkillCommit expands applicability through replay (He & Yang, 2026); BASM stores applicability, risk, and avoidance information (Lin et al., 2026); SkillGen combines contrastive induction with downstream validation (Ma et al., 2026b). Contract2Tool infers tool preconditions and effects from metadata, documentation, and traces for downstream tool filtering (Babu & Iyer, 2026). We isolate applicability synthesis in a typed DSL and select new executions against the current candidate. Our literature-inspired controls isolate selected mechanisms; their adaptation scope is documented in Appendix F.

Synthesis and specification inference. CEGIS alternates proposals with verifier-produced counterexamples (Solar-Lezama et al., 2006); neural variants learn the synthesizer (Polgreen et al., 2020; Si et al., 2020). Closely related DbC-GPT synthesizes postconditions using symbolic counterexamples from a reference implementation (Leite et al., 2026). Other LLM systems generate and refine specifications (Ma et al., 2025), refine intrusion rules (Yamamoto & Katsurai, 2026), or translate requirements into temporal logic (Cosler et al., 2023). SpecSyn strengthens specifications through program-mutation discrimination (Ma et al., 2026a), whereas our mutation checks are offline audits, not synthesis feedback. Here the proposer has neither source access nor a symbolic verifier: selected executions can falsify a candidate but cannot certify closure-wide correctness.

Active learning from executions. Query-based system learning dates to L^* (Angluin, 1987); L*LM (Vazquez-Chanlatte et al., 2025) uses an LLM to answer membership queries, whereas here the environment answers and the LLM proposes. Daikon tests invariant templates against execution traces (Ernst et al., 2007). Class-balanced active learning reweights samples using model-predicted scores (Fairstein et al., 2024); our rule allocates execution checks across a contract's Boolean acceptance partition to expose false accepts and rejects.

Self-refinement. Self-Refine (Madaan et al., 2023) motivates our control for an additional critique-and-revision call without new execution feedback.

3 Problem setting

An environment is a finite deterministic transition system $(\mathcal{S}, \mathcal{A}, \delta, s_0)$ with a distinguished target skill $a^\star \in \mathcal{A}$. Executing $a^\star$ in state s either *succeeds*, yielding $\delta(s, a^\star) = s'$, or is *rejected*, leaving s unchanged. A *contract* for $a^\star$ is a pair $(\mathrm{pre}, \mathrm{post})$ where $\mathrm{pre} : \mathcal{S} \to \{0, 1\}$ predicts acceptance and $\mathrm{post} : \mathcal{S} \times \mathcal{S} \to \{0, 1\}$ constrains the successor. A *reference* contract $(\mathrm{pre}^\star, \mathrm{post}^\star)$ is withheld from the model; its agreement with the environment is checked on the reachable set.

Given a candidate contract c and the reachable set $\mathcal{R} \subseteq \mathcal{S}$, the evaluator reports three counts:

$$\mathrm{FA}(c) = |\{s \in \mathcal{R} : \mathrm{pre}_c(s) \wedge \neg \mathrm{pre}^\star(s)\}| \qquad \text{(false accepts)}$$
$$\mathrm{FR}(c) = |\{s \in \mathcal{R} : \neg \mathrm{pre}_c(s) \wedge \mathrm{pre}^\star(s)\}| \qquad \text{(false rejects)}$$
$$\mathrm{PV}(c) = |\{s \in \mathcal{R} : \mathrm{pre}_c(s) \wedge \mathrm{pre}^\star(s) \wedge \neg \mathrm{post}_c(s, \delta(s, a^\star))\}| \qquad \text{(postcondition violations)}$$

A contract is *exact* iff $\mathrm{FA} = \mathrm{FR} = \mathrm{PV} = 0$: its applicability agrees with the reference on every reachable state and its effects hold on all admitted successful transitions. Postcondition equivalence and fault rejection are separate properties (Appendix T).

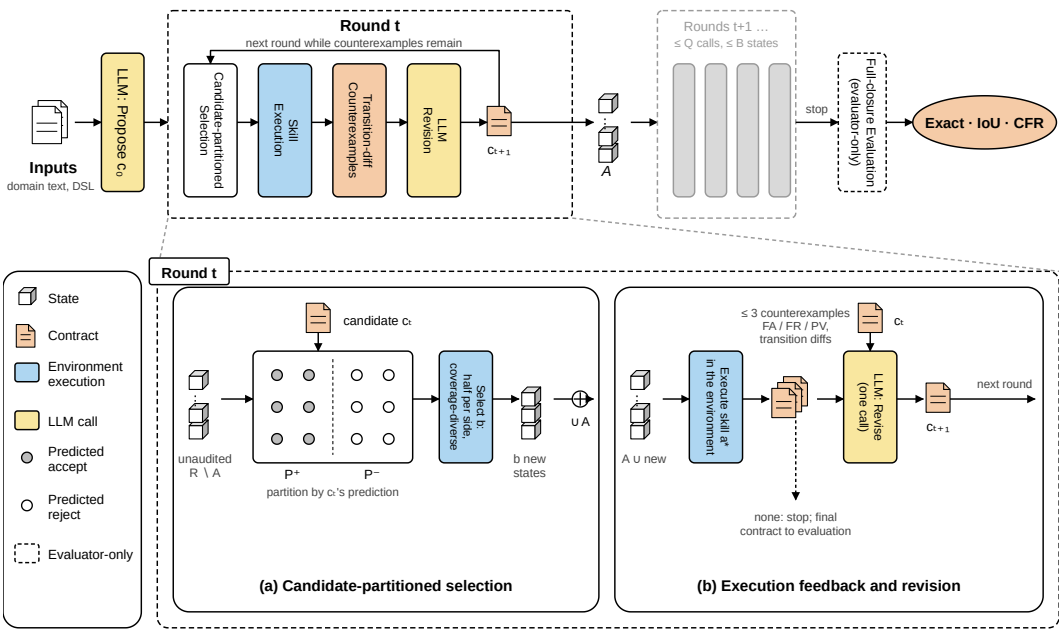

Figure 1: Execution-guided contract synthesis. Top: the bounded loop (at most Q model calls and B feedback states); bottom: one round expanded. The candidate c_t partitions the unaudited states, a batch drawn from both sides is executed, and up to three counterexamples from the accumulated audit set A drive one revision. Full-closure evaluation runs only after synthesis and is never shown to the model.

Partial recovery. Let $\mathcal{R}^+ = \{s \in \mathcal{R} : \mathrm{pre}^\star(s)\}$ be the executable states and $N_+ = |\mathcal{R}^+| > 0$. Applicability IoU measures precondition recovery; contract fulfillment (CFR) also includes observed effect consistency:

$$\mathrm{AppIoU}(c) = 100\frac{N_+ - \mathrm{FR}(c)}{N_+ + \mathrm{FA}(c)}, \qquad \mathrm{CFR}(c) = 100\frac{N_+ - \mathrm{FR}(c) - \mathrm{PV}(c)}{N_+ + \mathrm{FA}(c)}. \tag{1}$$

CFR is 100% iff recovery is exact. Main tables average IoU and CFR, assigning zero to attempts without an evaluable final contract. Table 14 gives exact counts and Wilson intervals; the detailed supplement reports valid yield and PVR. Hypothesis tests use exact recovery; postcondition strength is audited separately.

Information and access. The model receives state fields, configuration options, the DSL, and target skill name. Price-aware Shopping hides accepted options, coupon rules, and payment limits; other main-domain prompts name valid/rejected sets and give effect examples. Order-neutral retail removes the postcondition example block (Appendix S). Source code, the reference contract, and full-closure scores remain evaluator-only. Budget B caps distinct states allocated to feedback; repeated audits, target executions, and offline enumeration are accounted for separately (Appendix D).

Contract DSL. Typed JSON trees combine Boolean predicates, comparisons, multiset operations, and frame conditions over before/after variables and public sets. Calendar and retail expose clause-level predicates such as `no_room_conflict` and `items_in_order` (Appendix S). A decoding schema constrains JSON structure; remaining type errors trigger one budgeted repair call.

4 METHOD

Active-CEGIS runs a bounded CEGIS loop with query budget Q and state budget B (Appendix D). Algorithm 1 shows immediate stopping on a clean audit. Historical domain runners differ in termination and batch schedules; a common-loop study crosses selection with termination (Appendix K).

Algorithm 1 Active-CEGIS: schematic immediate-stop variant

$c_0 \leftarrow$ PROPOSE(domain description); $A \leftarrow \emptyset$ ▷ audited states
$b \leftarrow \lceil B / \max(Q - 1, 1) \rceil$ ▷ batch size
for $t = 0, 1, \ldots$ while calls $< Q$ and $|A| < B$ **do**
  $X_t \leftarrow$ SELECT$(c_t, \mathcal{R} \setminus A, b)$ ▷ Section 4.1
  $A \leftarrow A \cup X_t$
  $r_t \leftarrow$ SCORE(c_t, A) ▷ execute $a^\star$ on A; FA/FR/PV and counterexamples
  if r_t has no counterexample **then**
    stop (`sampled_oracle_satisfied`)
  end if
  $c_{t+1} \leftarrow$ REVISE(c_t, r_t) ▷ structured transition-diff prompt
  if $c_{t+1} = c_t$ **then**
    stop (`duplicate_model_output`)
  end if
end for
return last well-formed c_t

4.1 CANDIDATE-PARTITIONED SELECTOR

The selector decides which b unaudited states to execute. Active-CEGIS composes two rules.

Balance (candidate-partitioned sampling). Partition unaudited states by the current candidate: $P^+ = \{s : \mathrm{pre}_{c_t}(s)\}$ and $P^- = \{s : \neg\mathrm{pre}_{c_t}(s)\}$. Roughly half the batch is drawn from each predicted partition to seek false accepts and false rejects that uniform sampling may miss. Domain adapters differ in odd-batch allocation and replenishment (Appendix D). Partitioning uses candidate predictions, not hidden labels: sparse reference acceptance and sparse candidate-error regions are distinct.

Coverage (relational-feature diversity). Within each side, states greedily maximize relational-feature novelty, with seeded random tie-breaking; Appendix C gives the feature vector.

We ablate each rule: **No-Balance** keeps coverage but drops the partition; **No-Coverage** keeps the partition but draws uniformly within each side; **Uniform** drops both. All four selectors are deterministic functions of the seed, the candidate, and the audited set, and are recorded in the artifact as a named policy.

4.2 STRUCTURED TRANSITION-DIFF COUNTEREXAMPLES

The Active loop uses structured transition-diff feedback. Its contribution is evaluated jointly with the loop; feedback and schedule differences are detailed in Appendices C and D.

Each Active audit rescores accumulated states and returns up to three counterexamples, cycling over FA/FR/PV. Each includes the symptom, before/after states (after is `null` on rejection), and changed/unchanged fields. Revision requests supply the previous contract and ask for contradicted clauses to be corrected, without conversation history.

Illustrative revision. In one Gemma-4-31B Deployment run (seed 0), the candidate admitted an authenticated state with an empty allocation and valid quota, region, and tier; execution rejected it because no resources were allocated. The revision preserved all other clauses and added (surrounding JSON omitted) `{"op":"not", "arg":{"op":"is_empty", "arg":{"var":"allocated_resources", "when":"before"}}}`. The sampled audit's false accepts fell from 2/16 states to 0/32 accumulated states; evaluator-only closure scoring changed from 36 to zero over 11,712 states, with no false rejects or postcondition violations. This single run illustrates candidate → execution → revision, not a representative effect estimate.

Table 1: Four core domains and the legacy Basic Shopping check. Accepting: successful states; clauses: reference preconditions. †Six of eight clauses have independent witnesses. Retail uses upstream tool code on an author-built fixture. Details: Appendix A.

Domain	Variant	States	Transitions	Accepting	Clauses	Role
Shopping (price)	full	8,064	95,437	13	8	sweep
Shopping (price)	reduced	462	4,251	13	8	pilot ($n = 5$)
Shopping (price)	**relaxed**	288	1,048	8	8†	difficulty study
Shopping (price)	identifiable	864	5,480	8	8	identifiability check
Shopping (basic)	–	4,752	47,470	22	6	legacy ceiling check
Deployment	–	11,712	129,464	88	6	main
Calendar	–	1,676	12,376	35	10	main comparison
τ-bench retail (external)	`exchange_delivered_order_items`	1,294	8,116	14	8	external-tool fixture

4.3 Guards and repair

Byte-identical revisions and unrepaired vacuous candidates terminate the loop. A parse failure triggers one budgeted repair call with the type error; a run with no well-formed contract is inexact (Appendix C).

5 Experimental setup

Domains. Table 1 lists four core domains and the legacy Basic Shopping check. The external fixture uses the upstream retail tool (Yao et al., 2024). Price-aware Shopping has full, reduced, relaxed, and identifiable variants. Relaxed independently tests six of eight clauses, while singleton payment and address sets weaken two of those tested boundaries to non-null checks (Section 7). Definitions and provenance are in Appendix A.

Methods. Methods share each protocol's domain description, DSL, decoding, and evaluator. Direct and Self-Refine (Madaan et al., 2023) receive no iterative execution feedback. Random is non-interactive except in legacy retail. Fixed-pool matches budget caps but differs in feedback schedules and realized costs (Appendix D). Active-CEGIS re-selects against each candidate; No-Balance, No-Coverage, and Uniform ablate its selector (Appendix B).

Budgets and decoding. Runs use $B = 48$, $Q = 4$, temperature 0.2, and top-p 0.95; stars and diamonds mark the recorded reasoning suites. After observing truncation, we froze separate manifests for 58 main-table and 52 ablation runs before one full-run retry with doubled completion, run-token, and context limits. Each retry replaces its original outcome regardless of success; eight truncate again without further retry. Exact limits, both-attempt costs, serving details, and OFF/ON checks are in Appendices D, C, and P.

Models. Main models are Gemma-4-31B-it, Qwen3.8-27B, and Qwen2.5-32B-Instruct; Qwen2.5-14B is a smaller reference, and Qwen3.6-35B-A3B and Gemma-4-26B-A4B are MoE comparisons. Later suites add Qwen2.5-Coder-32B-Instruct (Qwen2.5-Coder-32B below) and Muse-Glimmer-30B (Appendix Q). Family, training, and decoding differences preclude attributing performance to active parameter count alone.

Protocol and statistics. Main cells are 20 paired seed repetitions on a fixed fixture (five for reduced Shopping), not 20 independent tasks. Exact counts and Wilson intervals accompany same-seed McNemar tests with Holm correction in historical core ($m = 4$), selector ($m = 3$), and post-result factorial ($m = 16$) families; retry-affected contrasts, later model suites, and follow-ups are reported separately (Appendices C, I, and R).

6 Results

Every reference precondition clause has a single-clause falsifying witness in Deployment, Calendar, and retail; relaxed Shopping is the exception (Table 26, Appendix T). These witnesses test individual clause omissions.

Table 2: Mean IoU/CFR (%, $n = 20$) under the 58-run truncation-retry replacement policy. Bold: Active; underline: positive per-metric row maximum, including ties. *: reasoning ON; ◇: native low; selected retries use enlarged limits. Lower block: descriptive. Calendar: three-method subset shared by the original dense suites; later historical controls in Appendices P, Q. Protocols: Appendix I; exact: Table 14.

Model	Deployment										Calendar					
	Direct		Self-Ref.		Random		Fixed-pool		**Active**		Direct		Self-Ref.		**Active**	
	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR
Gemma-4-31B	71.0	71.0	71.0	71.0	48.4	47.9	75.3	75.3	**100.0**	**100.0**	57.5	57.5	50.0	50.0	**100.0**	**100.0**
Qwen3.8-27B	31.9	0.0	63.9	17.7	56.0	10.2	52.6	12.1	**95.0**	**55.0**	15.0	15.0	30.0	30.0	**65.0**	**65.0**
Qwen2.5-32B	22.9	22.9	22.9	22.9	55.8	55.8	35.6	35.6	**57.1**	**53.4**	73.2	73.2	76.4	76.4	**87.4**	**82.4**
Qwen2.5-14B	18.4	11.2	13.8	3.8	36.9	20.0	36.3	30.8	**88.2**	**88.2**	37.2	35.7	36.7	14.8	**40.1**	**40.1**
Gemma-4-26B-A4B*	9.8	0.0	27.5	0.0	18.2	0.0	45.1	10.2	**85.3**	**55.0**	41.3	41.3	41.5	35.2	**68.5**	**68.5**
Qwen3.6-35B-A3B*	37.1	15.0	61.9	28.4	53.8	23.5	49.9	41.7	**100.0**	**100.0**	35.3	20.2	55.9	29.5	**77.9**	**76.0**
Muse-Glimmer-30B◇	66.9	41.3	82.7	64.2	33.3	31.6	77.2	52.3	**100.0**	**100.0**	85.0	70.0	97.5	92.5	**100.0**	**100.0**
Qwen2.5-Coder-32B	3.9	0.0	32.9	7.1	21.5	18.5	48.6	8.9	**80.0**	**15.0**	78.2	43.2	76.7	36.7	**83.7**	**83.7**

Table 3: Retail (1,294 states): mean IoU and CFR (%, $n = 20$) under the truncation-retry replacement policy; notation as in Table 2. Fixed-pool reuses B states. Legacy Sampled and Random are excluded because their policies differ from the other domains' controls. The reference matches upstream behavior throughout the closure.

Model	Direct		Self-Refine		Fixed-pool		**Active**	
	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR
Gemma-4-31B	14.0	14.0	14.0	14.0	53.2	53.2	**98.9**	**98.9**
Qwen3.8-27B	16.1	12.6	16.3	16.3	41.2	34.9	**51.2**	**47.6**
Qwen2.5-32B	12.3	12.3	11.9	7.1	35.2	27.1	**39.6**	**39.6**
Qwen2.5-14B	9.1	9.1	7.0	7.0	19.7	18.4	**27.9**	**27.9**
Gemma-4-26B-A4B*	6.9	5.9	7.0	5.5	25.2	23.8	**37.8**	**35.1**
Qwen3.6-35B-A3B*	9.2	9.2	17.0	13.3	28.6	27.8	**35.3**	**35.3**
Muse-Glimmer-30B◇	18.4	15.9	55.9	43.1	46.0	33.6	**69.3**	**69.3**
Qwen2.5-Coder-32B	39.2	39.2	38.8	25.2	48.2	48.2	**68.5**	**64.3**

6.1 Main comparison: Deployment and Calendar

Table 2 separates applicability from effect consistency: Qwen3.8 Deployment Active reaches 95% IoU but 55% CFR. Random has higher CFR than Active for Qwen2.5-32B (55.8 versus 53.4) and Coder (18.5 versus 15.0), reflecting more partial fulfillment despite fewer exact runs (Table 14). Muse Calendar Self-Refine already reaches 97.5% IoU, 92.5% CFR, and 18/20 exact; Active reaches 100% on both measures and 20/20. Its incremental gain is therefore small in this supplied-vocabulary task.

Exact gains over Direct remain significant for Gemma, Qwen3.8, and 14B on Deployment (all $p_{\text{Holm}} < 0.004$), and for Gemma and Qwen3.8 on Calendar. Qwen2.5-32B's Deployment difference is non-significant (raw $p = 0.51$, $p_{\text{Holm}} = 1.0$). Exact tables and protocol distinctions are in Table 14 and Appendix I.

6.2 τ-bench retail

Gemma-4-31B reaches 19/20 exact versus 0/20 for Direct and Self-Refine. Qwen3.8's CFR rises from 12.6% to 47.6% despite only 1/20 exact. Table 3 uses the corrected fixed-pool controls. Across Gemma's $B \in \{8, 16, 24, 48\}$ sweep, fixed-pool recovery is 0–3/20 versus Active's 4–20/20; residual fixed-pool failures at $B = 48$ mainly over-accept (Appendix J). Strengthened effect semantics retain Gemma's 19/20 versus 3/20. All methods pass that bounded postcondition audit, locating the difference in applicability recovery (Appendix T.2).

Table 4: Relaxed Shopping (288 states): mean IoU and CFR (%, $n = 20$) under the truncation-retry replacement policy; notation as in Table 2. Exact and intervals: Table 14. Table 5 applies the same policy to the selector comparison.

Model	Direct		Self-Refine		Random		Fixed-pool		**Active**	
	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR	IoU	CFR
Gemma-4-31B	36.4	0.0	36.4	0.0	41.7	0.9	77.7	34.8	**100.0**	**100.0**
Qwen3.8-27B	9.5	2.1	10.2	1.8	45.8	0.1	60.4	37.1	**70.0**	**70.0**
Qwen2.5-32B	3.2	0.0	3.2	0.0	0.0	0.0	10.5	3.5	**8.1**	**0.0**
Qwen2.5-14B	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	**0.7**	**0.0**
Gemma-4-26B-A4B*	30.5	10.9	36.4	19.1	11.9	0.3	74.1	58.0	**71.7**	**56.7**
Qwen3.6-35B-A3B*	33.0	16.8	36.4	20.0	22.7	9.7	89.5	68.1	**100.0**	**100.0**
Muse-Glimmer-30B◇	32.7	0.0	34.5	0.0	97.0	2.5	72.3	46.1	**95.0**	**95.0**
Qwen2.5-Coder-32B	0.0	0.0	1.8	0.0	0.0	0.0	4.0	0.0	**0.0**	**0.0**

Order-generalization check. With two exchangeable orders (4,500 states), Active recovers preconditions in 9/20 runs versus 0/20 for Direct. Here W1 is the originally delivered order and W2 is the second, newly deliverable order. The supplied legacy snapshot predicate is W1-specific and rejects valid W2 transitions. Jointly neutralizing the prompt and predicate yields 9/20 versus 0/20 exact ($p_{\text{Holm}} = 0.023$); residual false rejects concern W2 (Appendix A.5).

6.3 Shopping difficulty study: relaxed variant

On deliberately relaxed Shopping (Table 4), Gemma reaches 100% IoU/CFR. On the separately evaluated identifiable variant, where every clause has a witness, it reaches 0/20 exact (Section 7). Under the retry policy, Qwen3.8 Active reaches 70.0% IoU/CFR, versus 60.4% IoU and 37.1% CFR for Fixed-pool. In a separate post-hoc four-comparison core family, their same-seed Exact contrast is 10:2 discordant ($p_{\text{raw}} = p_{\text{Holm}} = 0.0386$). Selector contrasts form their own family (Table 5).

What counterexamples repair. Clause-specific witness tests locate most Gemma repairs in omitted composed clauses; named-set and dedicated-predicate clauses are usually present. On Calendar, Qwen3.8 Direct rejects every single-clause witness yet achieves only 3/20 exact, exposing additional constraints or effect errors (Appendix S).

Call budgets and evidence selection. On relaxed Shopping, Gemma's Self-Refine doubles Direct's calls without improving exact recovery. Fixed-pool and Active share caps and counterexample format: the former rescores 48 states; the latter adds about 16 candidate-selected states per round. Mean calls are similar (3.00 versus 3.15), but exact recovery is 5/20 versus 20/20. This contrast combines selection and feedback timing.

6.4 Selector ablation

Removing balance lowers exact recovery in all nine above-floor unstarred rows of Table 5, significantly in six of eight tested contrasts. In the starred follow-ups it lowers recovery in six of seven above-floor cells; retail with Qwen3.6 ties at 1/20. Under the retry policy, Qwen3.8 Shopping also exceeds Uniform (14/20 versus 6/20; $p_{\text{Holm}} = 0.043$). On Deployment, the descriptive MoE Active/No-Coverage results are 11/9 for Gemma-26B and 20/17 for Qwen3.6. On retail with Gemma-31B, No-Balance stops on sampled satisfaction in all twenty runs at a mean of 1.15 calls; across $B = 8, 16, 24, 48$ it remains at 0/20 (Appendix J). Coverage effects are mixed: No-Coverage exceeds Active in several starred cells, and order-neutral W2 retail favors coverage only numerically (9/20 versus 6/20, $p_{\text{Holm}} = 0.91$), so we attribute the selector's contribution to partitioning. Full-closure scoring also distinguishes sample satisfaction from exact recovery: on Deployment with Qwen2.5-32B, 15/20 runs satisfy their sample and 7/20 are closure-exact.

Later descriptive suites also show lower exact/20 without balance: Muse drops from 19 to 1, 20 to 6, and 20 to 13 on relaxed Shopping, Deployment, and Calendar; Qwen2.5-Coder-32B drops from 14 to 3 on Calendar (Appendix Q). These contrasts include balance's effects on evidence exposure.

Table 5: Selector ablation, exact/20, with the same truncation-retry policy for Active and ablations (58 main-table and 52 ablation retries). Basic Shopping uses $B = 32$; other rows use $B = 48$. Unstarred rows: paired McNemar p (Holm, $m = 3$); changed contrasts are post-hoc policy families. "n.s." = no difference detected. Starred ON rows remain descriptive; historical OFF/ON results: Appendix P. Dense Calendar suites completed only No-Coverage and are omitted here.

Domain	Model	**Active**	No-Balance	No-Coverage	Uniform
Shopping (relaxed)	Gemma-4-31B	**20**	0 (5.7×10^{-6})	20 (n.s., 0:0)	5 (1.2×10^{-4})
Shopping (relaxed)	Qwen3.8-27B	**14**	2 (1.5×10^{-3})	13 (n.s., 5:4)	6 (4.3×10^{-2})
Shopping (relaxed)	Qwen2.5-32B	**0**	0	0	0
Shopping (basic)	Qwen2.5-32B	**20**	7	20	4
Deployment	Gemma-4-31B	**20**	3 (3.1×10^{-5})	20 (n.s., 0:0)	1 (1.1×10^{-5})
Deployment	Qwen3.8-27B	**11**	1 (3.9×10^{-3})	9 (n.s., 2:0)	0 (2.9×10^{-3})
Deployment	Qwen2.5-32B	**7**	2 (n.s., 7:2)	4 (n.s., 5:2)	1 (n.s., 7:1)
Deployment	Qwen2.5-14B	**16**	1 (5.5×10^{-4})	16 (n.s., 1:1)	0 (9.2×10^{-5})
τ-bench	Gemma-4-31B	**19**	0 (1.1×10^{-5})	19 (n.s., 1:1)	0 (1.1×10^{-5})
τ-bench	Qwen3.8-27B	**1**	0 (n.s., 1:0)	4 (n.s., 1:4)	0 (n.s., 1:0)
Shopping (relaxed)	Gemma-4-26B-A4B*	**8**	6	11	8
Shopping (relaxed)	Qwen3.6-35B-A3B*	**20**	2	20	7
Deployment	Gemma-4-26B-A4B*	**11**	0	9	0
Deployment	Qwen3.6-35B-A3B*	**20**	2	17	2
Calendar	Gemma-4-26B-A4B*	**13**	0	15	1
Calendar	Qwen3.6-35B-A3B*	**14**	0	16	5
τ-bench	Gemma-4-26B-A4B*	**0**	0	0	0
τ-bench	Qwen3.6-35B-A3B*	**1**	1	3	0

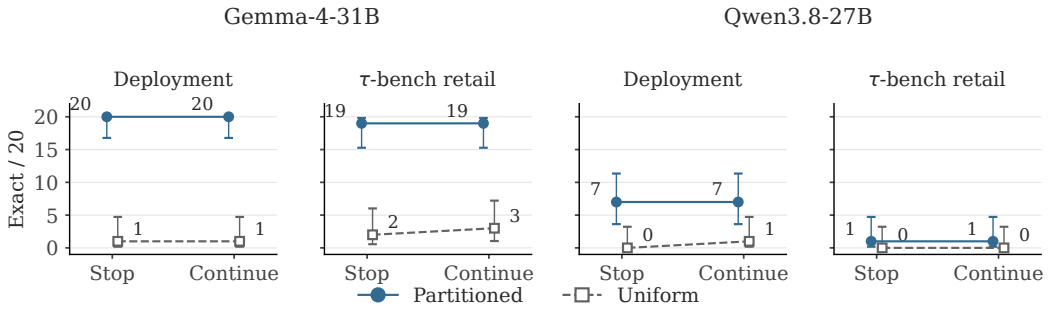

Figure 2: Selection and termination in the common-loop study ($B = 48, Q = 4$, coverage off). Points show exact/20; bars are Wilson 95% intervals across repeated seeds on fixed fixtures. Stop ends on a clean audit; Continue audits further without a model call until a counterexample or budget stop. Lines connect categorical conditions. Paired tests and domain-adapter differences are detailed in Appendix K.

Separating selection from termination. A common-loop 2×2 study (320 runs; shared initial candidates) crosses selection and termination (Figure 2). For Gemma, partitioning outperforms uniform selection in both domains under both rules (all four Holm-adjusted $p < 0.001$ across sixteen comparisons); Qwen3.8 differences are non-significant. Partitioned exact outcomes match by seed across stopping rules; continued auditing improves two uniform cells by one seed each. This supports a within-protocol selection advantage; interaction and equivalence require separate tests (Appendix K). Two additional Deployment configurations show a Gemma advantage (10/10 versus 0/10 each, Holm $p = 0.0078$), with non-significant Qwen3.8 differences (Appendix L).

Basic Shopping ceiling and literature baselines. Basic Shopping provides a ceiling check: Active reaches 20/20 versus at most 6/20 for the reduced literature-inspired controls, whose definitions and limitations are given in Appendix H.

7 WHERE THE METHOD FAILS

Difficulty sweep. We simplified full price-aware Shopping until a model achieved nonzero exact recovery. All four dense models score 0/20 for both Direct and Active on full (8,064 states), and 0/5 in the reduced pilot (462 states). Reduced restores rejected payment/address options, a remote surcharge, a stricter coupon threshold, and binding payment-capacity settings (Appendix A.1). All eight clauses have independent witnesses, versus six in relaxed, whose payment/address boundaries reduce to non-null checks. These variants change several structural features together.

The identifiable variant. Repairing all four relaxed-domain degeneracies yields 864 states versus 288, with every clause identifiable (Appendix A.1). Gemma-4-31B falls from 20/20 on relaxed to 0/20 for both Direct and the loop. Refinement reduces false accepts from 104 to a median of 12 but misses some clauses. Doubling budgets still yields 0/20; Qwen3.8-27B also scores 0/20. Thus relaxed recovery certifies only distinctions supported by that closure.

Model failures on relaxed Shopping. Original reasoning-off Qwen2.5-32B and 14B runs confuse pre/post-order payment fields, sometimes retaining contradicted clauses. Qwen3.6 shows repetition-induced truncation and type errors; changing the repetition penalty reduces truncation but leaves exact recovery rare. Final parse failures affect 72/160 14B and 141/160 Qwen3.6 runs; parsed failures retain incorrect conditions (Appendices N, O). Reasoning and adaptive-budget follow-ups recover some successes (Appendix P), showing these outcomes' dependence on decoding and budgets.

8 LIMITATIONS

External validity. Three core domains and the legacy check are small synthetic environments; relaxed Shopping was constructed to permit some exact recovery. The external τ-bench tool (Yao et al., 2024) runs unmodified on our sliced fixture, whose default closure admits an order-specific exact contract (Appendix A.5). This tests one external protocol, not downstream agent performance. **Contract language.** The DSL has limited arithmetic and domain-specific predicates, without quantifiers or temporal operators. Expressibility and learnability remain unresolved contributors to full Shopping's difficulty. **Closure discrimination.** Exhaustive evaluation removes sampling error within the reachable support, but cannot distinguish rules without witnesses. Relaxed Shopping's missing and weakened boundaries limit what exactness certifies (Section 7); all clauses in other audited domains have independent witnesses (Appendix T). **Serving dependence.** Original Qwen2.5-14B uses a different serving stack; separating its effect requires cross-server replication (Appendix C). **Postcondition strength.** Among 153 exact Active contracts, 118 reject all tested single-field faults and 138 all reference-invalid faults (Appendix T.1). Legacy retail omits three frame conditions, one beyond its DSL. Exposing that field and strengthening effects yields 160/160 postconditions passing the bounded audit (Appendix T.2). This establishes neither effect equivalence nor a postcondition-learning advantage; supplied effect examples and predicates also support recovery (Appendix S).

Reference-contract validity. The reference agrees with the environment on every reachable state and, in retail, with the third-party tool throughout the fixture. Shared errors and validity beyond that support remain unchecked.

9 CONCLUSION

Execution-guided synthesis recovers closure-exact applicability on Deployment, Calendar, and retail, including Muse's 20/20 on Deployment and Calendar with native low reasoning. The selector component that matters is candidate partitioning: removing it lowers exact recovery in every above-floor unstarred ablation row, and the common-loop study confirms the advantage for Gemma under both stopping rules, with the same direction but no significant difference for Qwen3.8. In local and hosted cohorts, the partitioned loop's residual failures are mostly false rejects hidden in the candidate's large predicted-reject region (Appendices J and R). The open problem is undisclosed rules: once each hidden Shopping rule has a witness, neither tested model recovers the contract.

AI USE STATEMENT

Generative AI tools assisted with literature search and synthesis, research ideation, experimental design, and interpretation of results. They also assisted in constructing synthetic environments, implementing and refactoring experiment and analysis code, preparing figures, and drafting and editing parts of this manuscript. AI assistance also covered the mathematical formulation: the formal problem setting, the evaluation metrics, and the short supporting arguments stated with them (Section 3 and Appendix T). The paper states no formal theorems, and the author verified these arguments. The language models evaluated here are themselves the objects of study: synthesized contracts are model outputs. The author reviewed and tested AI-assisted code, recomputed reported results from preserved experimental artifacts using verified analysis scripts, and reviewed and edited AI-assisted prose and figures. The author takes responsibility for the final text, claims, code, and artifacts.

REPRODUCIBILITY STATEMENT

Each reported cell is backed by per-seed JSON artifacts containing run outcomes and available configuration hashes, request and response traces, per-round metrics and counterexamples, and final full-closure evaluations. Recorded prompt-protocol hashes are retained; the fifteen basicShopping cells (300 runs) predate that field, so their stored requests, configurations, and source revision are preserved without reconstructing the missing hash. For each selected retry, aggregation records both the base source and retry artifact, verifies their hashes and common sandbox/evaluation lineage, and retains their distinct decoding and budget hashes rather than requiring one cell-wide setting hash. Read-only scripts check run invariants and reconcile the 136 main-table cells (2,720 runs), their exact counts, Wilson intervals, and IoU/CFR means against stored evaluations; paired exact tests and selected printed p-values are checked separately. The depth-24/28 Deployment closure was re-enumerated; the historical postcondition audit is reconciled against stored source hashes. The domain definitions, DSL, prompts, runners, configurations, analysis scripts, correction gate, and every experimental artifact cited in this paper are available at `https://github.com/Anonymous-692/fidref-submission` (revision `16815af`) and in the anonymized supplementary archive; the restore step checks each result file against a release manifest before the analysis scripts run.

ETHICS STATEMENT

The experiments use synthetic fixtures, including the fixture for external tool code, and involve no human subjects or personal data. Contracts synthesized by this method are not verified beyond the enumerated state space and should not be used as safety guarantees in deployed systems without independent verification.

REFERENCES

Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

Rahul Suresh Babu and Laxmipriya Ganesh Iyer. Contract2Tool: Learning preconditions and effects for reliable tool-augmented LLM agents, 2026. URL https://arxiv.org/abs/2606.07904.

Matthias Cosler, Christopher Hahn, Daniel Mendoza, Frederik Schmitt, and Caroline Trippel. nl2spec: Interactively translating unstructured natural language to temporal logics with large language models. In *Computer Aided Verification*, pp. 383–396, 2023. doi: 10.1007/978-3-031-37703-7_18.

Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, 69(1–3):35–45, 2007.

Yaron Fairstein, Oren Kalinsky, Zohar Karnin, Guy Kushilevitz, Alexander Libov, and Sofia Tolmach. Class balancing for efficient active learning in imbalanced datasets. In *Proceedings of the 18th Linguistic Annotation Workshop*, pp. 77–86, 2024. doi: 10.18653/v1/2024.law-1.8.

Yu He and Weikai Yang. Skillcommit: Evolving agent skills through behaviorally validated scope expansion, 2026. URL https://arxiv.org/abs/2608.15165.

Gabriel Leite, Filipe Arruda, Pedro Antonino, Augusto Sampaio, and A. W. Roscoe. Generating formal smart-contract specifications: Comparing few-shot learning and fine-tuned llms. *Science of Computer Programming*, 252:103464, 2026. doi: 10.1016/j.scico.2026.103464.

Zihan Lin, Zhenyu Chen, Jiawen Wei, Xiaohan Wang, Jie Cao, Jiajun Chai, Wei Lin, Guojun Yin, and Ran He. When not to imitate: Boundary-aware skill memory for reliable tool-use llm agents, 2026. URL https://arxiv.org/abs/2608.22339.

Zijian Lu, Yiping Zuo, Yupeng Nie, Xin He, Weibei Fan, Lianyong Qi, and Shi Jin. Contractskill: Repairable contract-based skills for multimodal web agents, 2026. URL https://arxiv.org/abs/2603.20340v3. v3, April 2026.

Lezhi Ma, Shangqing Liu, Yi Li, Xiaofei Xie, and Lei Bu. Specgen: Automated generation of formal program specifications via large language models. In *Proceedings of the 47th IEEE/ACM International Conference on Software Engineering*, 2025. doi: 10.1109/ICSE55347.2025.00129.

Lezhi Ma, Shangqing Liu, Yi Li, Qiong Wu, Han Wang, and Lei Bu. SpecSyn: LLM-based synthesis and refinement of formal specifications for real-world program verification, 2026a. URL https://arxiv.org/abs/2604.21570.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. Skillgen: Verified inference-time agent skill synthesis, 2026b. URL https://arxiv.org/abs/2605.10999.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Elizabeth Polgreen, Ralph Abboud, and Daniel Kroening. Counterexample guided neural synthesis. *CoRR*, abs/2001.09245, 2020.

Jie-Jing Shao, Haiyan Yin, Yueming Lyu, Xingrui Yu, Lan-Zhe Guo, Ivor W. Tsang, James T. Kwok, and Yu-Feng Li. Lifting traces to logic: Programmatic skill induction with neuro-symbolic learning for long-horizon agentic tasks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Xujie Si, Aaditya Naik, Hanjun Dai, Mayur Naik, and Le Song. Code2inv: A deep learning framework for program verification. In *Computer Aided Verification*, pp. 151–164, 2020.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 404–415, 2006.

Marcell Vazquez-Chanlatte, Karim Elmaaroufi, Stefan Witwicki, Matei Zaharia, and Sanjit A. Seshia. L*lm: Learning automata from demonstrations, examples, and natural language. In *Proceedings of the International Conference on Neuro-symbolic Systems*, volume 288 of *Proceedings of Machine Learning Research*, pp. 543–569, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Conference on Language Modeling*, 2025a. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025b.

Kohei Yamamoto and Marie Katsurai. Verification-guided specification synthesis with large language models for intrusion detection rules, 2026. URL https://arxiv.org/abs/2608.22889.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

A Domain descriptions

The synthetic environments share a common state representation; the externally sourced τ-bench tool (Appendix A.5) is wrapped in the same interface. A state is an immutable record of scalar, enum, and canonical-multiset fields. Actions are grounded (every parameter is enumerated from a public option set), deterministic, and either succeed with a new state or are rejected with the state unchanged. The agent-facing transition code never imports the evaluator-only reference; a regression test checks that the two agree on every reachable state. The reachable set is enumerated by BFS from a fixed initial state with `max_depth` $= 20$ and `max_states` $= 15{,}000$; all reported reachable-state sets are complete under the checks below. Deployment receives an additional larger-depth verification because its deepest level reaches the default cap (Table 1). In every closure except the relaxed Shopping variant, each reference precondition clause has at least one reachable state in which that clause alone changes the decision (Appendix T). Prompt disclosure varies by domain: price-aware Shopping hides accepted subsets and thresholds, whereas other domains name valid sets or expose clause-level predicates (Appendix S).

A.1 Price-aware Shopping (difficulty study)

State (14 fields): `logged_in`; `cart` and `stock` (item multisets); `payment_method`, `shipping_address`, `coupon_code` (optional strings); `wallet_balance`, `card_limit` (integer cents); the derived fields `cart_subtotal` and `checkout_total`; and the order slot `order_status` $\in$ {none, placed, confirmed, cancelled} at the state-type level. Under the price-aware action space used here, only `none` and `placed` are reachable. The remaining order fields, `order_items`, `order_total`, and `order_payment_method`, are populated only by a successful `place_order`.

Actions (12 kinds): `login`, `logout`, `add_to_cart`(i), `remove_from_cart`(i), `clear_cart`, `set_payment`(m), `clear_payment`, `set_address`(a), `clear_address`, `apply_coupon`(c), `clear_coupon`, and `place_order` (target).

Monetary rules (hidden). Subtotal is the price-weighted cart total. A coupon is eligible iff it is an accepted code, the subtotal meets `coupon_min_subtotal`, and the cart contains none of `coupon_excluded_items`; an eligible coupon subtracts `coupon_discount`. Shipping is free at or above `free_shipping_threshold`, otherwise `shipping_fee`, plus `remote_shipping_surcharge` for the `remote` address. The payable total is subtotal $-$ discount $+$ shipping.

Reference precondition for `place_order` (eight clauses): (1) logged in; (2) order slot empty; (3) cart non-empty; (4) payment method in the accepted set; (5) address in the serviceable set; (6) stock covers the cart; (7) no coupon applied, or the applied coupon is eligible; (8) payment capacity: wallet balance $\geq$ payable total if paying by wallet, otherwise payable total $\leq$ card limit.

Reference postcondition: order status becomes `placed`; `order_items` equals the previous cart; cart is emptied; stock is decremented by the cart; `order_total` equals the payable total; `order_payment_method` equals the previous payment method; wallet is debited by the payable total iff paying by wallet; card limit unchanged; derived totals reset to zero; coupon cleared; `logged_in` unchanged; payment method and address cleared or kept per the configuration flag `clear_checkout_selections_after_order`.

Variants. Table 6 gives the four configurations; the configuration files are `shopping_price_default.json` (full), `shopping_price_moderate_v3.json` (reduced), `shopping_price_easy_v4.json` (relaxed), and `shopping_price_easy_v5.json` (identifiable), and result directories carry the file names. The full variant has three payment methods (one rejected), three addresses (one unserviceable), a remote surcharge, a coupon threshold above the cheapest single item, and wallet/card limits below the largest possible total, so all eight clauses are active; it is the domain on which the tested Direct and Active cells scored 0/20. The reduced variant retains the payment and address options but changes the coupon configuration and post-order exploration (Table 6). The relaxed variant additionally collapses the payment and address option sets to singletons, lowers the coupon threshold below any non-empty subtotal, and raises the wallet balance and card limit above any reachable total, so that clause (8) is constant over the reachable set. Clauses (4) and (5) still vary in relaxed, but only as non-null tests, because the single selectable option is accepted; clause (7) takes both truth values but never independently of the other clauses, and clause (8) is constant (Appendix T). The identifiable variant starts from relaxed and changes only five settings: it adds a rejected payment method (`voucher`) and an accepted one (`wallet`), adds an unserviceable address (`remote`), excludes `gift` from the coupon, and lowers the wallet balance to 3,000 so that it binds; its closure has 864 states and every clause has a witness (Appendix T).

Table 6: Price-aware Shopping configurations. Amounts in cents. Shared by all four: items `book` (2400) and `gift` (2500) with opening stock 2 and 1; cart capacity 2; coupon `save500` discounts 500; shipping fee 600 below a free-shipping threshold of 4000. Parenthesised sets are the hidden accepted subsets.

	full	reduced	relaxed	identifiable
Payment methods (accepted)	card, wallet, exp._card (card, wallet)	card, wallet, exp._card (card, wallet)	card (card)	card, wallet, voucher (card, wallet)
Addresses (serviceable)	home, remote, po_box (home, remote)	home, remote, po_box (home, remote)	home (home)	home, remote (home)
Coupon codes (accepted)	save500, invalid (save500)	save500 (save500)	save500 (save500)	save500 (save500)
Coupon min. subtotal / excluded	3000 / gift	3000 / none	500 / none	500 / gift
Remote surcharge	400	400	0	0
Wallet balance / card limit	4300 / 5000	4300 / 5000	10000 / 10000	3000 / 10000
Terminal after order	no	yes	yes	yes
Post-order cart add / clear selections	no / no	yes / no	yes / yes	yes / yes
States / transitions / accepting	8,064 / 95,437 / 13	462 / 4,251 / 13	288 / 1,048 / 8	864 / 5,480 / 8
Independently identifiable clauses	8	8	6	8

Identifiable-variant diagnostics. Gemma-4-31B Direct produces one contract in all twenty seeds, missing the accepted payment set, serviceable address set, coupon exclusion, and empty-order-slot clause (FA $= 104$). The loop recovers the order-slot clause in 16 of 18 parsed runs and the payment set in eight, the address set in two, and the coupon exclusion in none; final false accepts have a median of 12. Exact recovery is 0/20, including after the tested budget doubling.

A.2 Basic Shopping (ceiling and literature baselines)

The basic storefront domain has no prices. **State** (7 fields): `logged_in`, `cart`, `stock`, `payment_method`, `shipping_address`, `order_status`, `order_items`. **Actions** (13 kinds): relative to price-aware Shopping, this domain omits `apply_coupon` and `clear_coupon`, and adds `confirm_order`, `cancel_order`, and `clear_order`. Configuration: items `book`, `pen` with stock 2 each; cart capacity 2; selectable payment methods `card`, `voucher`, `expired_card` of which the first two are accepted; selectable addresses `home`, `po_box` of which only `home` is serviceable. **Reference precondition** (six clauses): logged in; order slot empty; cart non-empty; payment method accepted; address serviceable; stock covers cart. Postcondition: order placed with the cart as `order_items`, cart emptied, stock decremented, other fields unchanged. Closure: 4,752 states, 47,470 transitions, 22 accepting states.

The literature-baseline comparison (Table 9 in Appendix H) uses $B = 24$, $Q = 4$, 1,024 completion tokens, and the same decoding for all seven arms; the state-budget sweep and selector ablation on this domain use $B \in \{4, 8, 16, 32, 48, 64, 96\}$ and $B = 32$ respectively. Observed-Error Repair

improves with budget (6/20 at $B = 24$, 10/20 at $B = 48$, 12/20 at $B = 96$) but remains below Active-CEGIS at $B = 8$ (20/20).

A.3 DEPLOYMENT

A cloud-orchestration lifecycle with no vocabulary shared with Shopping. **State** (7 fields): `authenticated`; `allocated_resources` and `available_quota` (multisets over resource types `cpu`, `ram`); `target_region` and `cluster_tier` (optional strings); `deployment_status` $\in$ {idle, deployed, stopped, rolledback}; `active_deployment` (multiset). **Actions** (13 kinds): `authenticate`, `revoke_auth`, `allocate_resource`(r), `release_resource`(r), `clear_resources`, `set_target_region`(g), `clear_target_region`, `set_cluster_tier`(t), `clear_cluster_tier`, `deploy_service` (target), `stop_service`, `rollback_service`, `clear_deployment`. Configuration: quota 2 per resource type; allocation limit 2; selectable regions `us-central`, `europe-west`, `unsupported-edge` of which the first two are valid; selectable tiers `standard`, `premium`, `deprecated-v0` of which the first two are supported. Rollback refunds quota; stop does not.

Reference precondition for `deploy_service` (six clauses): authenticated; status idle; allocation non-empty; region valid; tier supported; available quota covers the allocation. **Postcondition**: status becomes `deployed`; `active_deployment` equals the previous allocation; allocation cleared; quota decremented by the allocation; authentication, region, and tier unchanged. Closure: 11,712 states and 88 accepting states. The deepest discovered level equals the default depth cap of 20, so we re-enumerated at caps of 24 and 28; both reproduce the state set and the complete set of 129,464 state-changing successful transitions. The quota-coverage clause is the one relational constraint; its distinguishing states require a prior deployment to have consumed quota.

A.4 CALENDAR

A workspace-scheduling domain designed around relational constraints. **State** (7 fields): `authenticated`; `draft_attendees` (set over `alice`, `bob`); `draft_slot`, `draft_room`, `draft_type` (optional strings); `external_booking` (an independent booking with its own room, slot, and attendees, or none); `active_meeting` (a snapshot of the scheduled meeting, or none). **Actions** (16 kinds): `login`, `logout`, `add_attendee`(p), `remove_attendee`(p), `clear_attendees`, `set_slot`(s), `clear_slot`, `set_room`(r), `clear_room`, `set_meeting_type`(t), `clear_meeting_type`, `set_external_booking`(b), `clear_external_booking`, `schedule_meeting` (target), `cancel_meeting`, `load_active_meeting`. Configuration: slots `morning`, `afternoon`, `midnight` (last rejected); rooms `room_alpha` (capacity 2, video-capable), `room_beta` (capacity 1, no video), `deprecated_room` (rejected); meeting types `in_person`, `video_conf`, `unsupported_type` (last rejected); two selectable external bookings (`room_alpha`/morning/alice and `room_beta`/afternoon/bob).

Reference precondition for `schedule_meeting` (ten clauses): authenticated; no active meeting; attendees non-empty; slot valid; room valid; type valid; attendee count $\leq$ room capacity; type is not `video_conf` or the room supports video; the external booking does not occupy the same room in the same slot; the external booking does not share an attendee in the same slot. The last four clauses are relational (they compare two or more fields, or a field against a configuration table) and have no counterpart in Shopping or Deployment; each is separately falsifiable by a reachable state at BFS depth ≤ 6. **Postcondition**: `active_meeting` equals the snapshot of the draft; the draft fields are cleared; external booking and authentication unchanged. Closure: 1,676 states, 12,376 transitions, 35 accepting states; all ten clauses vary over the reachable set.

A.5 τ-BENCH RETAIL (EXTERNAL)

This is the only domain whose target transition function comes from external code. The transition function is the upstream tool implementation itself, vendored without modification from `sierra-research/tau-bench` at commit `59a200c6` (MIT); we replaced only the package `__init__` files, with empty ones, so that the tools run without the upstream inference dependencies.

The retail database is not upstream's: it is a small fixture we wrote (Section 8). Checksums of the vendored files are listed with the released artifacts.

Target skill. `exchange_delivered_order_items`: exchange items in a delivered order for other variants of the same products, charging or refunding the price difference to a payment method.

Fixture. A minimal slice of the upstream retail database: two valid orders (`#W1` delivered, `#W2` pending) plus one non-existent order id; two order items and one item that belongs to no order; three replacement variants (two in stock, one not) plus one that is not a variant of anything ordered; and three payment methods—a gift card, a credit card, and one that does not belong to the user. The draft exchanges one item for one replacement.

Auxiliary actions. Three further upstream tools drive the state space: `cancel_pending_order` cancels `#W2` and refunds it to the gift card (raising the balance from \$15 to \$40, which is what makes the higher-priced exchanges reachable), `return_delivered_order_items` moves `#W1` out of `delivered` and so out of eligibility, and `modify_pending_order_payment` changes the payment method on the pending order. Together with the draft-building actions this gives 27 grounded actions and a closure of 1,294 states over 8,116 successful transitions, BFS depth 6, untruncated; 14 states (1.08%) accept the target skill.

Reference precondition (eight clauses, transcribed from the tool source): (1) the draft order id exists; (2) that order is `delivered`; (3) every item to exchange is present in the order; (4) the number of items to exchange equals the number of replacements; (5) each replacement is a valid variant of the corresponding product; (6) each replacement is in stock; (7) the payment method belongs to the user; (8) if the payment method is a gift card, its balance covers the price difference. The postcondition sets the order status to `exchange requested` and records the exchange. Over the full closure the reference contract agrees with the vendored tool's own accept/reject decision on every state (FA = FR = PV = 0).

Clause distinguishability. Every clause has states in which that clause alone flips the decision:

Clause	Witnesses	Density	Min. depth
`order_exists`	56	4.33%	1
`order_delivered`	22	1.70%	2
`items_in_order`	8	0.62%	4
`item_count_matches`	20	1.55%	3
`new_items_valid_variant`	8	0.62%	4
`new_items_available`	4	0.31%	4
`payment_method_exists`	16	1.24%	1
`sufficient_balance_if_gift_card`	2	0.15%	4

Domain vocabulary. Like Calendar, this domain adds clause-level predicates to the shared grammar (`items_in_order`, `valid_variants`, `items_available`, `sufficient_balance`, the postcondition predicate `exchange_snapshot_matches`, and the term `price_difference`; Appendix S). The postcondition predicate is written against order `#W1`, the only order that is delivered in the default closure. Beyond these, the domain required one change to a shared operator: `in_set` now accepts a list-valued term and holds when every element is in the named set, with the empty list vacuously true; option- and text-valued terms behave as before, and numeric or boolean arguments still raise a type error. The reason is that the draft carries *lists* of item ids, and models wrote `in_set` over the whole list rather than element by element; before the extension that natural form was a parse failure. The change accepts a form the models already produce and adds no information about the reference contract, without specifying which conjunction is correct. It is nevertheless a domain-specific language adaptation.

The W2-deliverable variant. We rebuilt the fixture so that a second order also becomes deliverable (4,500 states, 42 accepting, of which 26 name `#W1` and 16 name `#W2`) and reran four methods with Gemma-4-31B. Under the legacy prompt and vocabulary, no run is exact. Preconditions are recovered completely (FA = FR = 0) in 9/20 Active-CEGIS runs, 7/20 without coverage, 5/20 for legacy Sampled, and 0/20 for Direct. Postcondition violations occur in 11–20 runs per method. The supplied `exchange_snapshot_matches` implementation checks W1 effects and preserves W2 status, so it rejects successful W2 exchanges even when their preconditions are correct. This is

a vocabulary-support defect in the expanded fixture, rather than isolated evidence of model effect-reasoning failure. A hand-written closure-exact contract shows that other expressions in the grammar can describe the observable effects.

We then reran the same fixture under an order-neutral protocol, which removes the order-specific prompt examples and literals and generalises the snapshot predicate to the order named by the draft (four methods, twenty seeds, identical budgets, one shared initial request whose hash is identical across all eighty runs). Postcondition violations disappear in all eighty runs, so exact recovery and precondition-completeness coincide: 9/20 for Active-CEGIS, 6/20 without coverage, 3/20 for the fixed pool, and 0/20 for Direct. Across the six pairwise comparisons, only Active-CEGIS against Direct survives correction (9:0 discordant, $p = 0.0039$, $p_{\mathrm{Holm}} = 0.023$); Active-CEGIS against the fixed pool (8:2, $p_{\mathrm{Holm}} = 0.44$) and against No-Coverage (5:2, $p_{\mathrm{Holm}} = 0.91$) do not. The correction family was defined after inspecting the results. The recovered postconditions reflect a joint prompt-and-predicate intervention, with the exchange payload outside the state representation.

The neutral protocol retains two patterns. First, Active-CEGIS recovers preconditions completely in 9/20 runs under both protocols, with three seeds gaining and three losing recovery. Second, every residual false reject is a `#W2` state, including with coverage removed; no `#W1` state is falsely rejected. The legacy results are also predominantly W2-specific, with exceptions. W1-specific clauses in the final contracts directly cause the residual W2 false rejects. The fixture, exposed representation, and one coverage feature are asymmetric across orders, but their individual contributions to generating these clauses are not isolated; the same W1-specific clauses also occur in Fixed-pool and No-Coverage runs. Sample satisfaction also occurs beyond the fixed pool: runs that stop on a counterexample-free batch yet remain inexact number 17 for the fixed pool, 12 without coverage, and 9 for Active-CEGIS.

B Method definitions and information budgets

Within a matched protocol, methods share the domain description, DSL, decoding settings, and final evaluator. They differ in evidence, refinement, and stopping policies; historical exceptions are identified below.

- **Direct**: one proposal call with no execution evidence, plus the shared syntax/type repair call if the proposal fails to parse (which is why $\bar{c}$ exceeds 1.00 for some models).
- **Self-Refine**: Direct followed by one critique-and-revise call with no environment evidence (Madaan et al., 2023).
- **Random**: outside retail, one call whose prompt lists execution outcomes on uniformly random states. It executes 48 uniformly sampled states (the same state-budget cap, not necessarily the same states as Active-CEGIS) but shows the model only the first 15, as state-key/outcome lines rather than transition diffs, and does not iterate. Its visible-information budget is therefore *not* matched to Active-CEGIS. The legacy retail arm instead uses iterative uniform batches of eight; it is not this one-shot probing control.
- **Fixed-pool**: CEGIS over a *fixed* random pool of B states drawn once, rescored each round. Retail uses the separately collected `sampled_cegis_fixed` arm; its legacy Sampled arm instead performs partitioned No-Coverage selection. This is an iterative reference with its own feedback schedule.
- **Active-CEGIS**: Algorithm 1 with the balance+coverage selector.
- **No-Balance**, **No-Coverage**, **Uniform**: selector ablations (Section 4.1).

Figure 3 contrasts where the evidence for a revision comes from under Self-Refine, Fixed-pool, and Active-CEGIS.

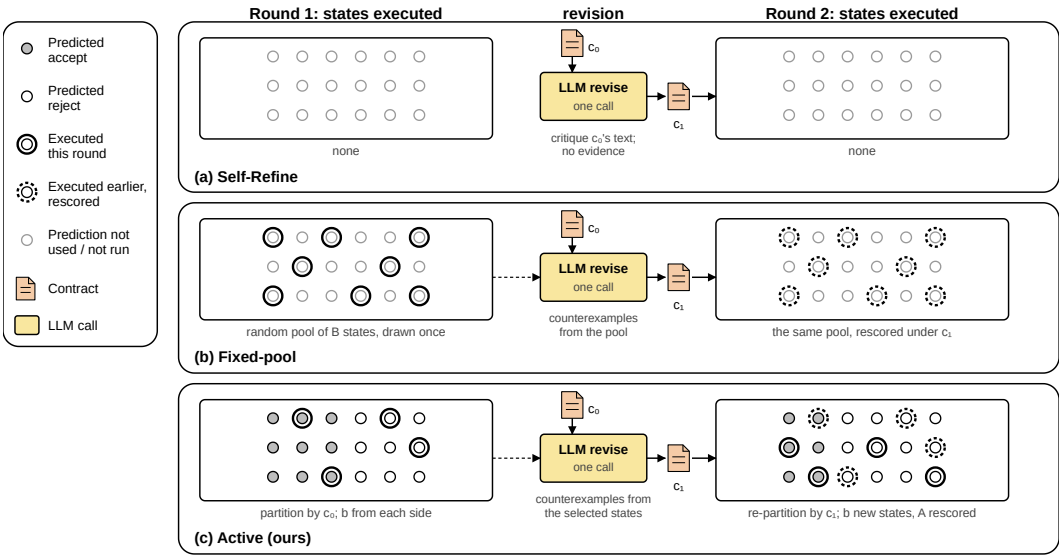

Figure 3: Evidence available to each method over two rounds on the same reachable set (schematic). Self-Refine revises from the contract text alone. Fixed-pool executes B states drawn once, independently of the candidate, and rescores the same pool after revision. Active-CEGIS partitions the unaudited states by the current candidate, selects from both predicted sides, and after revision repartitions with c_1 and adds new states while the earlier audit set is rescored. Ring styles mark execution; fills mark the candidate's prediction, not hidden labels.

C Implementation and evaluation protocol

Coverage features. Within each side of the candidate partition, states are chosen greedily to maximize novelty of a relational feature vector (in Shopping: login flag, cart size and contents, subtotal, checkout total, payment method, address, coupon, wallet balance, sign of the wallet gap, sign of the card-limit gap, order status, and stock shape), breaking ties by a seeded uniform draw. The intent is to spread the batch across the boundary structure rather than duplicate near-identical states.

Guards and repair. If a revision is byte-identical to its predecessor the loop stops (`duplicate_model_output`); if a candidate predicts acceptance on zero audited states it is sent back once for revision (`vacuous_candidate_unrepaired` if still vacuous). If a response fails to parse under the DSL type checker, one additional repair call is issued with the parser error; it is charged to the query budget and recorded as a separate interaction role. A run that ends without any well-formed contract is scored as inexact.

Model and serving coverage. Gemma-4-31B-it, Qwen3.8-27B, and Qwen2.5-32B-Instruct are the main models; Qwen2.5-14B-Instruct is a smaller reference across all domains. Original OFF comparisons also cover Qwen3.6-35B-A3B (3B active parameters) on relaxed Shopping, Deployment, and τ-bench, and Gemma-4-26B-A4B (4B active parameters) on all four domains. Original serving configurations use vLLM 0.19.1 on A100 (bf16) or 0.8.5.post1 on V100 (fp16, tensor-parallel 4); later decoding conditions and additional-model suites are distinguished in Appendices P and Q. Matching recorded prompt or protocol hashes does not establish identical effective decoding across servers. Table 2 identifies retained earlier-protocol Deployment and Calendar baselines; historical artifacts remain separate from final-protocol results.

Artifact gate and statistical families. Main comparison cells use 20 seeds (0–19); the additional Deployment configurations use ten per condition (Appendix L). The seed fixes the sampler; the model is called at temperature 0.2. Artifacts preserve configuration, decoding, and budget hashes plus requests and responses; protocol hashes are absent in the fifteen legacy basic-Shopping cells listed in Appendix I. The read-only gate checks seed completeness, sandbox and evaluation lineage,

per-attempt decoding/budget hashes, transport errors and empty responses, untruncated closure evaluation, oracle isolation, and call/state-budget compliance. A retry-policy cell may legitimately contain base and retry setting hashes; the explicit source–retry map is checked instead of asserting one decoding hash per cell. Historical basic-Shopping oracle-guided controls are described separately in Appendix F. We report exact/20 with Wilson 95% intervals. Historical pairwise comparisons against Active-CEGIS are same-seed McNemar exact binomial tests on discordant pairs, Holm-corrected within two families per model–domain cell: *core* (vs. Direct, Self-Refine, Random, Fixed-pool; $m = 4$) and *selector* (vs. No-Balance, No-Coverage, Uniform; $m = 3$). Retry-affected core and selector contrasts use separate post-hoc policy families; historical results remain in the corresponding appendix tables. The common-loop factorial uses a separate sixteen-comparison Holm family defined during post-result analysis (Appendix K); its contrasts are not members of these earlier families. The versioned retail follow-up and additional Deployment configurations use separate families fixed before their respective runs (Appendices T.2 and L). Some Direct cells repeat identical outputs across seeds, but recorded request identifiers are unique and request seeds match artifact seeds, so output duplication alone does not show the cross-seed dependence that would invalidate same-seed McNemar tests. Non-significance is not evidence of equivalence.

D BUDGET ACCOUNTING AND STATE-ACCESS ASSUMPTIONS

The symbol B caps *distinct states allocated to feedback*. A fixed-pool ledger reserves its sampled states before the first valid candidate, so allocation need not imply an executed audit. The number of states actually shown to the model can be smaller, as in the Random control. It is not a count of action invocations, and it does not cover the cost of building the environment closure. This appendix separates the three quantities that B is easy to misread as covering, and states the access assumption each one carries. Methods use the same reachable-state support within a domain configuration, but neither total execution cost nor closure-construction frequency is equalized by B.

Post-hoc truncation-retry cost. After observing output truncation, we froze a 58-run source–retry manifest before executing the retries. The selected base attempts used 193 model calls and 959,686 reported tokens; their one-time full-run retries used 158 calls and 799,431 tokens. Thus the two attempts together cost 351 calls and 1,759,117 tokens. This is a selected-subset total, not a corpus-wide campaign cost, and both attempts are charged even though only the retry outcome enters the main tables. All 58 retry outcomes are adopted: 16 previously inexact runs become exact, two exact runs regress, and neither failures nor regressions are filtered after execution. Base completion/run/context limits were 2,048/16,000/8,192, doubled to 4,096/32,000/16,384; for reasoning-ON MoE runs they were 3,072/20,096/8,192, doubled to 6,144/40,192/16,384. State and query budgets were unchanged, and six retries that again ended for length received no further retry.

The selector comparison subsequently applies the same policy to a separately frozen list of 52 ablation runs. Their base attempts used 202 calls and 1,146,097 tokens, and retries used 171 calls and 964,920 tokens: 373 calls and 2,111,017 tokens across both attempts. Exact recovery in this selected subset rises from 2 to 13, with 11 gains and no regressions; two retries still truncate and all 52 yield evaluable final contracts. Table 5 combines these outcomes with the main-table Active retry outcomes, retaining 20 seeds per cell. The historical OFF/ON and additional-model appendix tables retain their original policies.

Closure enumeration (cost per construction). A BFS from the fixed initial state constructs the reachable-state support before synthesis. Runner initialization can rebuild it, so construction counts are specific to each execution. One construction invokes the environment far more often than the closure has states, because BFS attempts the whole action space at each state and keeps only the successful transitions. Table 7 reports counts for one construction, not campaign totals. On relaxed Shopping, for instance, 288 reachable states cost 4,032 action invocations, of which 288 are invocations of the target skill. This cost is charged to no method and is not included in B.

In-loop auditing (charged, and what the ledger counts). In the domain runners, the state ledger records distinct allocated states: repeated audits do not add another distinct state, and the artifact's `states_observed` is the size of that set, bounded by B. Fixed-pool runs can reserve all B states yet perform no audit if candidate generation fails. Repeated scoring of the accumu-

Table 7: Measured cost of building and scoring each closure. *Actions* counts every environment invocation performed by the enumerating BFS, successful or rejected; *target* counts those that invoke the skill under contract. *Closure pass* is the cost of scoring one candidate contract over the whole closure, one target invocation per reachable state.

Domain	States	Transitions	Actions	Target	Closure pass
Shopping (price), full	8,064	95,437	153,216	8,064	8,064
Shopping (price), reduced	462	4,251	8,316	462	462
Shopping (price), relaxed	288	1,048	4,032	288	288
Shopping (price), identifiable	864	5,480	14,688	864	864
Shopping (basic)	4,752	47,470	85,536	4,752	4,752
Deployment	11,712	129,464	222,528	11,712	11,712
Calendar	1,676	12,376	41,900	1,676	1,676
τ-bench retail	1,294	8,116	34,938	1,294	1,294
τ-bench retail (W2-deliverable)	4,500	40,102	126,000	4,500	4,500

lated evidence adds target executions, not distinct states. Summing recorded audit sizes reproduces `sampled_states_checked` in all 100 runs of the five domain–method cells summarized next. For Deployment/Gemma, mean distinct states/audit executions are 48.0/96.0 for Active and 48.0/55.2 for fixed-pool; for Deployment/Qwen3.8 Active, 41.6/76.8. For retail/Gemma, they are 23.2/46.4 for Active and 48.0/110.4 for the corrected fixed pool. These are recorded audit executions, not total runtime or all validation calls. The cost ordering reverses across these domain–method pairs, so matched caps do not establish equal execution cost. The common-loop factorial uses its own loop-state counters and round-level reports (Appendix K), rather than assuming one accounting field applies to every runner.

Batch schedules and adapter boundaries. Legacy retail Active requests eight fresh states per audit and stops on a clean audit. Deployment, Calendar, and price-aware Shopping request $\lceil B/(Q-1)\rceil$ for $Q > 1$; Deployment continues checking a clean candidate while budget remains. Fixed-pool controls instead reveal B states once and rescore that same pool after revisions. Deployment's fixed-pool feedback can include five counterexamples per request, versus three for Active; it is not an identical-message control. For balanced selection, Deployment and basic Shopping allocate $\lceil b/2\rceil$ to predicted accepts and replenish shortfalls from the other side. Calendar, price-aware Shopping, and retail allocate $\lfloor b/2\rfloor$ to accepts and do not replenish a nonempty short side; an empty side transfers the budget, and at most b remaining states are all returned. Previously audited states are excluded from fresh batches. The common-loop study standardizes the loop and requested schedule, not these domain-specific adapters. Thus its within-domain effects concern the implemented selection policy, not a cross-domain invariant partition operator. Computing partitions also scans candidate predictions over reachable states; this CPU work is not charged to B.

Offline evaluation (charged to no method). Every reported metric is computed after the run by scoring the final contract over the untruncated closure, one target invocation per reachable state (the last column of Table 7). This pass is the evaluator's, is identical for all arms, and is never visible to the model. The same holds for the reference contract itself, which is evaluator-only.

State access. Selecting an arbitrary reachable state assumes the environment can be put into that state on demand. We satisfy this by construction: states are immutable records produced by the enumeration, and the target skill is applied to a chosen state directly, so no replay of an action prefix is needed. A deployed system without resettable state would pay an additional cost we do not model, and the shortest action prefix is bounded by the closure depth in Appendix A. This assumption is shared by every method here, including the non-interactive baselines, which execute states drawn the same way.

Oracle isolation is not free exploration. The oracle-isolated runs record `oracle_feedback_queries=` 0: no full-closure candidate evaluation is supplied during synthesis. Auditing selected states still evaluates candidate disagreements. Historical oracle-guided controls are separate (Appendix F). It is not a claim that state access, enumeration, or re-scoring is free. Those costs are the ones tabulated above.

E PROMPTS

Common contract and lifecycle. Each experiment requests one JSON object with keys `skill`, `precondition`, and `postcondition`, with no surrounding prose. The shared grammar uses `var` and `const` terms and formula operators `const`, `and`, `or`, `not`, `is_true`, `is_null`, `is_empty`, `eq`, `ne`, `lt`, `le`, `gt`, `ge`, `in_set`, and `unchanged`. Domain operators are listed per domain below; comparisons join like-typed terms. `before` is legal in the precondition, while `after` and `unchanged` are postcondition-only. The neutral JSON envelope is a schema demonstration, not a candidate. The initial prompt supplies state/configuration; the edit prompt receives `<previous contract JSON>` and preserves valid clauses; recovery receives `<previous response>` plus `<parser error>` and repairs only JSON/DSL errors. Optional stages revise vacuous preconditions, append random `Ok`/`Error` observations, or append deterministic changed/unchanged transition fields. No hidden oracle result enters synthesis.

E.1 PRICE-AWARE SHOPPING (RELAXED VARIANT)

Source	`experiment/shopping_price/prompts.py`
Configuration	`experiment/configs/shopping_price_easy_v4.json`
`prompt_sha256`	preserved in the full supplement

Public information: prices {`'book':2400,'gift':2500`}, cart capacity 2, choices `['card']`, `['home']`, `['save500']`, initial wallet/card limits 10000, and computed subtotal/checkout total. Selectability does not prove acceptance; policy boundaries are learned from observations. State covers cart/stock, order/payment fields, login, coupon, address, and balances. Domain terms are multiset `count`, `total`, `difference`, and `covers`; numeric subtraction is `subtract`. The prompt supplies generic operator syntax and a neutral JSON envelope with constant-true pre/postconditions, rather than domain-specific effect clauses or dedicated policy predicates (Appendix S).

Stages and reproduction. `system_prompt()` defines `place_order`; direct synthesis supplies the public configuration; self-refine critiques without new observations; parse repair names the error and prior response; vacuity revision preserves effects; random probing appends up to 15 sandbox states. Reproduce at `experiment/shopping_price/runner.py:534--538`; the verified artifact path and full hash are preserved in the supplement.

E.2 DEPLOYMENT

Source	`experiment/deployment/prompts.py`
Configuration	`DEFAULT_DEPLOYMENT_CONFIG`
`prompt_sha256`	preserved in the full supplement

Public information: resources `['cpu','ram']`; valid/rejected regions `['europe-west','us-central']`/`['unsupported-edge']`; valid/rejected tiers `['premium','standard']`/`['deprecated-v0']`; generated maxima quota/allocation 2. These maxima must not become contract thresholds without observations. State covers authentication, region/tier, active deployment, allocated/available resources, sizes, and status. Domain terms are `difference`, `union`, `get`, and `total`; sizes are already numbers, so no arithmetic or get/total applies to them. Critique checks authentication, quota, region/tier, effects, and frames. Reproduce from `experiment/deployment/prompts.py`; the verified artifact path and full hash are preserved in the supplement.

E.3 CALENDAR

Source	`experiment/calendar/prompts.py`
Configuration	`DEFAULT_CALENDAR_CONFIG`
`prompt_sha256`	preserved in the full supplement

Public information: slots `['morning','afternoon']`/`['midnight']`, rooms `['room_alpha','room_beta']`/`['deprecated_room']`, types `['in_person','video_conf']`/`['unsupported_type']`, attendees

['alice','bob'], max attendees 2. State covers auth, draft/active snapshots, external booking, and conflict flags. Domain terms are count, room_capacity, and room_supports_video; formulas additionally express overlap/conflict, capacity, support, and snapshot matching. Reproduce from experiment/calendar/prompts.py; the verified artifact path and full hash are preserved in the supplement.

Structured transition-diff revision. experiment/counterexamples.py:12--29 sorts state keys and emits changed before/after values or an unchanged list. The actual transmitted revision retained the complete previous contract in the preservation file. Its readable revision template was:

```
Previous contract: <complete previous contract JSON>
Observed transition diff: <changed fields and unchanged fields>
Output only the complete revised JSON contract.
```

Short counterexample: a logged-in state with cart {"gift":1}, order_status="placed", and checkout_total=3100 was a false accept; the sandbox returned "episode ended after order placement" and after-state null. The revision reported 8 false accepts, 0 false rejects, 0 postcondition violations, and 16 states checked.

F LITERATURE-INSPIRED MECHANISM CONTROLS

These controls compare selected mechanisms adapted to our DSL. Single-Demo omits ASI's source-task execution verification (Wang et al., 2025a); Positive-Replay omits SkillCommit's contrastive induction, compatibility grouping, and source-preserving consolidation (He & Yang, 2026); Observed-Error Repair omits ContractSkill's constrained patch operators and improvement-based acceptance (Lu et al., 2026). Scores therefore concern these adaptations; original-system superiority would require faithful reproductions. A separate Deployment study restores restricted patches and improvement-based acceptance under different evidence policies (Appendix G). It examines that adaptation's behavior while leaving the fidelity limitations of these controls intact.

The three literature-inspired controls are implemented in experiment/modeling/runner.py with prompts in experiment/modeling/prompts.py. Their key distinction is how the observation corpus is built and whether rejected as well as accepted states are observed.

Corpus and comparison conditions. probe_states(seed,count) samples reachable states and may contain both outcomes; it is the corpus used by contractskill_repair and, at the same seed and budget, by random_probe and sampled_cegis ("runner.py:380–388,626–628,1019–1026"). successful_states(seed,count) first filters for successful place_order executions ("runner.py:454–466"), so its observations are positive-only; the filtering scan is an offline corpus-construction simplification disclosed by the implementation.

	asi_replay	skillcommit_replay	contractskill_repair
Corpus sampler	successful_states	successful_states	probe_states
Observed instances	1 success	$\leq$ state budget	$\leq$ state budget
Accept/reject split	no	no	yes
Model calls	1	proposal + at most 2 repairs	$\leq$ query budget
Scored in loop against	—	replay set	replay set
oracle_feedback_queries	0	0	0

asi_replay makes exactly one call from one successful state and has no repair round on parse failure ("runner.py:821–849"); it does not query the full-closure oracle. skillcommit_replay proposes from the first positive state and cross-validates on the remaining positive replay states, with at most one semantic revision (and one parse-recovery round), without replaying that revision ("runner.py:851–981"). contractskill_repair is the only control that sees the accept/reject split. It repeatedly repairs against its bounded probe set while errors and query budget remain, but never queries full-closure equivalence inside the loop ("runner.py:983–1056"); final exact/inexact scoring for every arm still uses the full reachable closure.

These are controlled adaptations, not reimplementations of the source systems: the ASI-style arm omits Python skill execution and source-task validation; Positive-Replay does not re-replay its final revision; and the ContractSkill-style arm omits restricted patch operators, edit-budget/ improvement-based acceptance, and multi-step recovery. The historical basic-Shopping oracle-guided arm is distinct from these sampled controls and does not isolate a selector alone.

G RESTRICTED REPAIR AND ACCEPTANCE-POLICY DIAGNOSTIC

On Deployment (11,712 states), we implement two mechanisms omitted by Observed-Error Repair: a restricted patch interface and improvement-based acceptance. Patches support conjunct insertion, deletion, replacement, operand retargeting, operator replacement, and negation. Each patch permits at most three edits and each inserted or replaced formula at most seven nodes; these limits were chosen from the reference conjunct sizes. We validate the resulting DSL before adoption and charge rejected proposals to the budget. This is an adaptation: ContractSkill's task-success, cost, and edit objective, multi-step execution recovery, and original selector patches are not reproduced. Whole-contract and patch response formats are part of the treatment.

Within each model, all conditions share the initial contract for each of twenty seeds. A rewrites the whole contract on a uniform fixed pool of 48 states; B uses the same pool but accepts a patch only if $\Delta D_{\mathrm{obs}} < 0$, where $D_{\mathrm{obs}} = \mathrm{FA} + \mathrm{FR} + \mathrm{PV}$ on the observed evidence. C uses whole-contract rewrites with candidate-partitioned fresh evidence. Whole-contract conditions retain valid revisions without an improvement gate. Each condition uses $B = 48$, $Q = 4$, and a nominal 16,000-token budget including initial generation. The post-hoc A′ and B′ conditions instead search up to twelve uniform pools for one exposing an initial error, then hold that pool fixed. Their additional search cost is recorded separately: 1,632 state executions for 14B and 4,896 for Gemma per condition. These evidence-conditioned pools are not unconditional fixed-pool controls. B″ shares B′'s initial contracts, evidence, and patch interface but allows $\Delta D_{\mathrm{obs}} \leq 0$.

Table 8: Deployment repair diagnostic. Exact/20 and median full-closure $D = \mathrm{FA} + \mathrm{FR} + \mathrm{PV}$; A′, B′, and B″ are post-hoc conditions. Qwen2.5-14B uses V100/fp16; Gemma-4-31B uses A100/bf16. Comparisons are within model.

		Qwen2.5-14B		Gemma-4-31B	
Condition	Evidence / revision	Exact	Median D	Exact	Median D
A	Fixed pool / rewrite	0/20	118	3/20	36
B	Fixed pool / strict patch	0/20	160	3/20	36
C	Partitioned / rewrite	16/20	0	20/20	0
A′	Error-exposing pool / rewrite	2/20	52	17/20	0
B′	Error-exposing pool / strict patch	0/20	160	17/20	0
B″	Error-exposing pool / tie-accepting patch	0/20	116	17/20	0

A and B have identical exact outcomes within each model ($p = 1$); this is not an equivalence result. Only 11/20 14B seeds and 3/20 Gemma seeds enter repair on the unconditional pool, limiting what that contrast reveals. In the evidence-conditioned Gemma runs, both revision formats recover 17/17 seeds where an error is found; the three search failures remain in the denominator. Low initial diversity (8 contracts for 14B, 2 for Gemma) limits transfer to other initial errors.

Observed ties can hide improvements and regressions. Offline replay finds that nine of twelve distinct rejected 14B B′ proposals improve full-closure D despite an observed tie. Allowing ties reduces median D from 160 to 116, with nine seeds improved, none worsened, and eleven tied in final D (exploratory two-sided sign test, raw $p = 0.0039$); exact recovery remains 0/20. First-round requests and patches match on all twenty paired seeds. Later trajectories diverge after adoption. Figure 4 contrasts final outcomes with a within-run regression hidden by an observed tie. For Gemma, B′ and B″ each accept all 17 proposals, with no observed ties or rejections; the policies make identical decisions on the proposals encountered. The three missing repair rounds are search failures, not mismatched outputs.

The predeclared A–B, A–C, and B–C exact contrasts form a three-comparison Holm family per model; the later evidence and tie analyses are exploratory and reuse pilot seeds. Offline closure

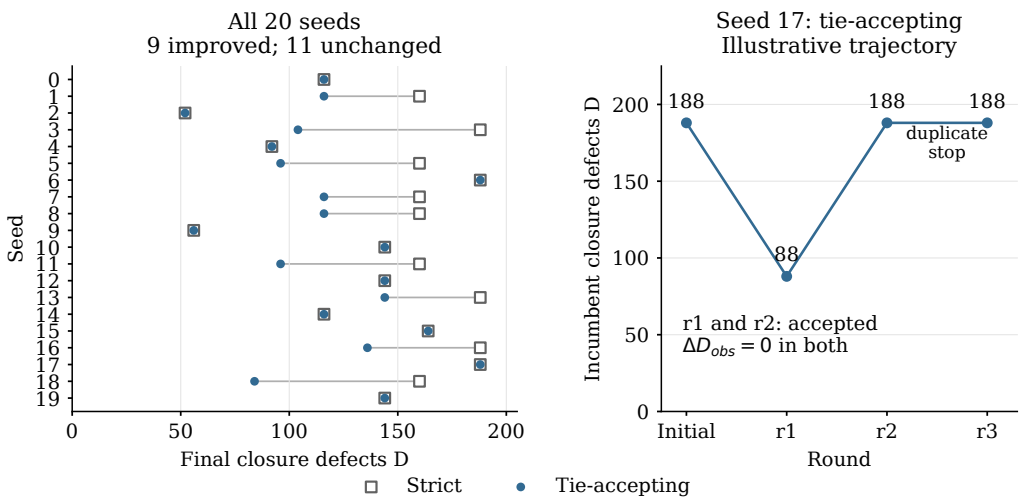

Figure 4: Post-hoc acceptance diagnostic, Qwen2.5-14B on Deployment. Left: all twenty paired seeds, strict B′ versus tie-accepting B″; overlapping markers denote identical final D. Both conditions remain at 0/20 exact. Right: seed 17's incumbent after each round. Two accepted patches each have $\Delta D_{\text{obs}} = 0$, yet full-closure D first falls from 188 to 88 and then returns to 188; r3 stops without adoption. This illustrates hidden improvement and regression, not a general advantage of accepting ties. Closure scores were computed offline, not supplied to synthesis.

diagnostics were not returned to synthesis. A historical top-level acceptance label for B′ was incorrect; round-level decisions and the preserved manifest document the actual strict rule. The corrected-code Gemma B′ rerun matches the earlier final contracts on all twenty seeds. These results describe the adapted mechanisms under limited evidence; they neither reproduce the original system nor validate the basic-Shopping controls. Full accounting and audits are retained in `analysis/contract_repair_core_v1/`.

H Basic Shopping ceiling and mechanism controls

Table 9: Basic Shopping (4,752 states), Qwen2.5-32B unless stated. Exact/20 and mean calls. The three reduced mechanism controls are described in Appendix F.

Method	Exact/20	$\bar{c}$
Direct	0	1.00
Self-Refine	0	2.00
Random	0	1.00
Observed-Error Repair (Lu et al., 2026)	6	1.30
Single-Demo (Wang et al., 2025a)	1	1.00
Positive-Replay (He & Yang, 2026)	1	1.65
Active-CEGIS	**20**	2.00
Active-CEGIS, Qwen2.5-14B	13	3.35
Direct / Self-Refine / Random, Qwen2.5-14B	0 / 0 / 0	1.00 / 2.00 / 1.00

The basic Shopping domain predates the price-aware variants and is at ceiling for Active-CEGIS with Qwen2.5-32B (20/20 in two calls on average, unchanged from $B = 8$ to $B = 96$; 14/20 at $B = 4$). The strongest reduced mechanism control reaches 6/20. We report this domain as legacy supporting evidence, not a comparison against the original skill systems or an isolated selector test; ceiling performance of Active-CEGIS also limits its diagnostic value.

Table 10: Relaxed-Shopping paired exact-binomial McNemar comparisons against Active CEGIS. Core and selector families contain four and three comparisons within each model, including zero-discordance rows ($p = 1$ for correction). Dashes indicate no discordant pairs, not equivalence.

Model	Family	Comparison	n	Active:Other	p_{raw}; p_{Holm}
Gemma-4-31B	core	Direct, Self-Refine, Random	20	20:0 (each)	$1.91e{-6}$; $7.63e{-6}$ (each)
Gemma-4-31B	core	Fixed-pool	20	15:0	$6.10e{-5}$; $6.10e{-5}$
Gemma-4-31B	selector	No-Balance	20	20:0	$1.91e{-6}$; $5.72e{-6}$
Gemma-4-31B	selector	No-Coverage	20	0:0	–; –
Gemma-4-31B	selector	Uniform	20	15:0	$6.10e{-5}$; $1.22e{-4}$
Qwen3.8-27B	core	Direct, Self-Refine, Random	20	11:0 (each)	$9.77e{-4}$; $3.91e{-3}$ (each)
Qwen3.8-27B	core	Fixed-pool	20	8:3	$2.27e{-1}$; $2.27e{-1}$
Qwen3.8-27B	selector	No-Balance	20	9:0	$3.91e{-3}$; $1.17e{-2}$
Qwen3.8-27B	selector	No-Coverage	20	5:7	$7.74e{-1}$; $7.74e{-1}$
Qwen3.8-27B	selector	Uniform	20	7:2	$1.80e{-1}$; $3.59e{-1}$
Qwen3.6-35B-A3B	core+selector	all seven	20	0:0 (each)	–; –
Qwen2.5-32B	core+selector	all seven	20	0:0 (each)	–; –
Qwen2.5-14B	core+selector	all seven	20	0:0 (each)	–; –

I ADDITIONAL STATISTICS

Cells without a recorded protocol hash. The fifteen basic-Shopping cells (300 runs) behind Table 9 in Appendix H and the basic-Shopping row of Table 5 were collected before the runner recorded a prompt-protocol hash. Each of those runs still carries its sandbox, decoding, and budget hashes, and every request and response verbatim, so the prompt text and decoding conditions are recoverable from the artifacts; the original binding hash remains unrecorded. We therefore report this domain as legacy supporting evidence: it is the ceiling and literature-baseline comparison, and no headline claim rests on it. The artifact manifest lists these cells with their recorded hashes and request counts, so a reader can see exactly which field is missing and which are present.

Protocol provenance of the baseline columns. The Deployment Direct and Self-Refine cells and the Calendar Direct cells of Gemma-4-31B, Qwen3.8-27B, and Qwen2.5-32B were collected before the final protocol revision and each retain one recorded `protocol_sha256` value, with six distinct values across these nine cells. We checked the stored requests: in all of them the system and user messages are byte-identical to those of the final-protocol runs (SHA-256 of each message matches), while differing metadata include the refinement protocol, counterexample format, and guided-decoding transport flag. Prompt equality does not establish equivalence of the effective decoding constraints. We retain these cells with an explicit provenance caveat and list their hashes with the artifacts.

Qwen2.5-32B Deployment Active reaches 14/20 under the earlier protocol and 7/20 under the final protocol used in Table 2; these are separate suites.

Rows with 0:0 discordant pairs contain no observed within-pair outcome differences. We retain $p = 1$ in the exact-test multiplicity calculation and display these rows as "no discordant pairs", not as evidence of equivalence. This includes the all-zero original OFF cells on relaxed Shopping and Gemma-4-26B-A4B on Deployment.

The main comparison cells summarized here use seeds 0–19. A run counts as exact when the synthesised contract has $\text{FA} = \text{FR} = \text{PV} = 0$ on the evaluated closure. This criterion tests precondition agreement and observed postcondition soundness, not logical equivalence of postconditions.

Table 14 reports exact counts and Wilson intervals after the main-table truncation-retry replacement policy. The paired tests printed in this appendix retain their historical base-policy outcomes; they are not recomputed by mixing retried Active cells with unretried selector arms. Retry-affected core contrasts use separate post-hoc policy families. The original reasoning-OFF exact results appear in Table 16 (Appendix P). Full intervals for the historical OFF and legacy cells and unabridged legacy statistics are preserved in `supplement/appendix_stats_tables_full.tex`.

The additional legacy OFF comparisons for Gemma-4-26B-A4B and retail are retained in the source supplement, including all non-significant and zero-discordance rows. They use separate core ($m = 4$) and selector ($m = 3$) families within each model–domain cell. The retail column historically named Sampled is partitioned no-coverage, not the corrected fixed pool; its tests must not be read as

fixed-pool evidence. The later starred reasoning rows (Appendix P) are separate conditions and are not covered by these legacy tests.

Later cell completion. The September 12 additions, 14B Calendar Self-Refine and Muse retail fixed-pool, each score 0/20 under their corresponding suite settings. They are descriptive additions outside existing test families; hashes and checks are in `analysis/table_gaps_20260912/verification.json`.

I.1 Mean model calls on relaxed Shopping

These are the original OFF-condition calls, excluding the later starred and adaptive-retry conditions in Appendix P.

Model	Direct	Self-Refine	Random	Fixed-pool	Active
Gemma-4-31B	1.00	2.00	1.00	3.00	3.15
Qwen3.8-27B	1.00	2.05	1.00	3.30	3.30
Qwen2.5-32B	1.75	2.40	1.45	3.35	3.35
Qwen2.5-14B	1.55	2.15	1.25	3.20	3.20
Qwen3.6-35B-A3B	2.00	2.20	1.80	2.10	2.00
Gemma-4-26B-A4B	2.00	2.50	1.70	2.70	3.10

State-budget sweep on basic Shopping. Active-CEGIS with Qwen2.5-32B over the full closure, 20 seeds per budget, all other settings fixed. Eight observed states already saturate the domain; the only budget that loses seeds is $B = 4$. Exact recovery is 14/20 at $B = 4$ and 20/20 at each of $B \in \{8, 16, 32, 48, 64, 96\}$.

J State-budget sweeps and the fixed-pool control

Two questions motivated these runs. First, does the advantage of candidate-partitioned selection survive when the state budget is small, or does it only appear once a domain is near its ceiling? Second, what does the fixed-pool control actually achieve on τ-bench, where the arm previously reported under that name turned out to run the candidate-partitioned loop (Section 6.2)?

All cells below use Gemma-4-31B, 20 seeds, and $Q = 4$. We vary the state budget while retaining each method's selection and termination rules. Deployment's requested batch size consequently changes with B; retail Active retains eight-state requests (Appendix D). The $B = 48$ retail points in Figure 5 reuse the existing cells, whose protocol hashes match the sweep runs. The fixed-pool arm is a separate method name with its own protocol hash; the earlier artifacts are unchanged.

The complete four-method counts, including No-Coverage and No-Balance, are preserved in `analysis/sweep_figure_20260910/all_cells.csv` with the original tables.

For Active versus fixed pool, two-sided exact McNemar tests are Holm-corrected over the four budgets separately within each domain, as an exploratory family. In increasing budget order, adjusted p-values are $3.05 \times 10^{-5}, 1.53 \times 10^{-5}, 2.29 \times 10^{-5}, 3.05 \times 10^{-5}$ on Deployment and $0.125, 2.29 \times 10^{-5}, 1.53 \times 10^{-5}, 2.90 \times 10^{-4}$ on retail. Thus retail at $B = 8$ has a numerical gap, not a significant contrast. The raw tests and discordant pairs are reproducible with `analysis/p0_sweep_audit.py`. These comparisons retain the original runners' schedule differences; the common-loop experiment is separate (Appendix K).

Three descriptive patterns follow. The numerical gap between candidate-partitioned selection and a fixed pool is present at every budget in both domains, and on τ-bench it is widest at $B = 24$ (20 against 1), not at the largest budget. Removing balance leaves the loop at or near zero at every budget. On retail, all 80 No-Balance runs stop on sample satisfaction, with mean calls 1.15 at each budget; raising the cap does not force further auditing under this stopping rule. None of the Active–No-Coverage contrasts is significant (all adjusted $p = 1$, separate four-budget family per domain); this is not evidence of equivalence.

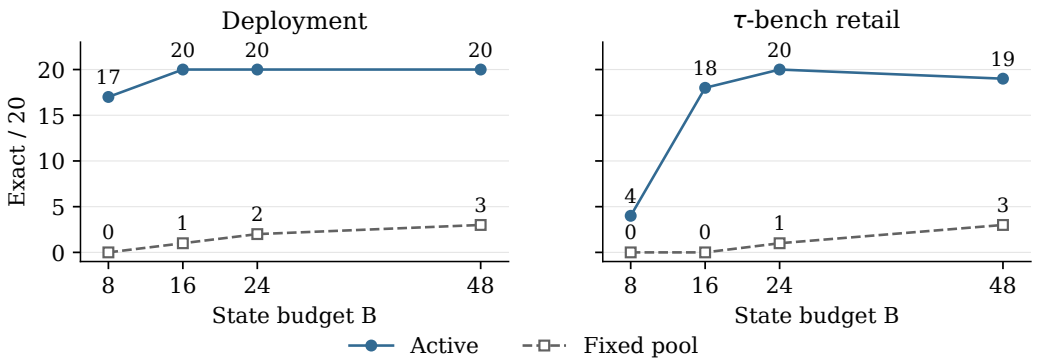

Figure 5: State-budget sweep with Gemma-4-31B, $Q = 4$, and twenty seeds per cell. Fixed pool draws B states once and re-scores them each round. Points mark tested budgets at their actual spacing; connecting lines do not predict untested budgets. Retail at $B = 8$ is not significant after correction ($p = 0.125$); its largest numerical gap occurs at $B = 24$ (20 versus 1). Budget caps are not measured execution cost, and the methods retain different audit schedules.

The fixed-pool failures are mostly not vacuous contracts. At $B = 48$ on τ-bench the fixed-pool arm has a median of 15 false accepts and a median of no false rejects; two of its twenty runs do have false rejects and one of those rejects every accepting state, and three runs are exact. The same shape holds on Deployment, where the fixed pool leaves a median of 36 false accepts and no run rejects everything. These residual errors are predominantly over-acceptance, not evidence that the cause is failure to observe accepting states. The sampling mechanism responsible is not isolated here: each of the seventeen non-exact fixed-pool runs in each domain at $B = 48$ ends by satisfying its own pool while the closure still contains counterexamples.

The same asymmetry appears across the local models and domains. Counting valid inexact runs in which false accepts outnumber false rejects, the main-table cells give 270/391 for Fixed-pool and 187/278 for Random, against 89/280 for Active-CEGIS. In the Table 5 cells, the two selectors without the partition give 265/321 (No-Balance) and 259/307 (Uniform), whereas the two partitioned selectors give 52/127 (Active-CEGIS) and 57/128 (No-Coverage). Evidence chosen without the candidate fails mainly by over-acceptance, while the partitioned loop's residual failures lean toward false rejects: 161 of the 280 main-table Active-CEGIS runs have more false rejects than false accepts. These are post-hoc descriptive counts without paired tests, computed from the retry-policy aggregates and stored FA/FR values; Calendar Random and Fixed-pool cells are outside the main IoU/CFR aggregate and are not included.

The hosted gap-fill cohort of Appendix R does not share the first half of this pattern. Among its valid inexact runs, false rejects outnumber false accepts in 50 of 65 Fixed-pool runs and 51 of 78 No-Balance runs. Terra's retail cell illustrates the case: every method ends with the same contract, which admits no failing state but rejects 4 of the 14 accepting states. The second half does hold: 137 of 200 inexact Active-CEGIS runs have more false rejects than false accepts. Such states lie in the candidate's large predicted-reject region, where a half-batch rarely reaches them. These hosted counts are post-hoc and descriptive; `analysis/openai_api_20260926/appendix_gapfill.py` regenerates them.

K Selection and termination: a common-loop factorial study

Design and scope. The original Deployment and retail runners differ in their no-counterexample handling: Deployment continues auditing the current candidate while state budget remains, whereas retail stops on the first counterexample-free audit. We therefore ran a separate 2×2 experiment using one common loop and domain adapters, not a relabeling of the original arms. Selection is candidate-partitioned uniform (P) or global uniform (U); termination is first-satisfaction stop (S) or continued auditing (E). Coverage is off in all four conditions. For each of two models, two domains, and twenty

seeds, the four conditions replay the same initial candidate, yielding 80 initial generations and 320 condition-runs. Each condition charges one initial model call; the original common-loop runner stores initial token usage separately rather than deducting it from the subsequent token ledger. There were not 320 independent initial generations. Three unparseable Qwen3.8 retail initial candidates are retained in all four conditions, giving twelve non-exact outcomes rather than excluded seeds. These are repeated synthesis trials on fixed fixtures, not twenty distinct tasks per cell.

Budget and audit semantics. All conditions use $B = 48$, $Q = 4$, a 16,000-token budget for subsequent requests, the same guards and revision format, and a requested batch size of 16. Continued auditing makes no model call while the current candidate remains consistent with the audited set; it seeks a counterexample or exhausts the remaining state budget. Parse, duplicate, vacuity, and model-budget stops can terminate a run earlier, so E observes at most 48 states. Both conditions retain previously audited states and exclude them from fresh selection. The retained Deployment adapter replenishes an undersized partition; the retail adapter does not when both sides are nonempty (Appendix D). Actual batch sizes and feedback timing can therefore differ as consequences of selection. This experiment controls the implemented termination rule and requested batch schedule; realized batch sizes and feedback timing remain mediators of its selection effect.

Statistical analysis. We compare P with U separately at S and E, and E with S separately at P and U, within each domain–model cell: sixteen same-seed comparisons in total. Exact recovery uses all twenty seeds, including parse failures. Two-sided exact McNemar tests use only discordant pairs, with Holm correction over *all sixteen* comparisons, including rows with no discordances (assigned $p = 1$ for correction but displayed as dashes). This multiplicity family was chosen during analysis after the results were available, not preregistered. Inference is confined to within-cell comparisons. Shared seeds support pairing across conditions on each fixed fixture, with four model–domain settings in total.

Table 11: Common-loop factorial: exact/20, including initial parse failures. P: candidate-partitioned; U: global uniform; S: stop; E: continue auditing.

Domain	Model	P–S	P–E	U–S	U–E
Deployment	Gemma-31B	20	20	1	1
Deployment	Qwen3.8	7	7	0	1
τ-bench	Gemma-31B	19	19	2	3
τ-bench	Qwen3.8	1	1	0	0

Table 12: Common-loop paired simple effects. Selection: P minus U; termination: E minus S. Holm correction covers all 16 comparisons, including zero-discordance rows. Dashes indicate no discordant pairs (bookkeeping $p = 1$), not equivalence.

Domain	Model	Effect	Fixed	a:b	Raw p	Holm p
Deploy.	Gemma-31B	Selection	S	19:0	3.81e-06	6.1e-05
Deploy.	Gemma-31B	Selection	E	19:0	3.81e-06	6.1e-05
Deploy.	Gemma-31B	Termination	P	0:0	–	–
Deploy.	Gemma-31B	Termination	U	0:0	–	–
Deploy.	Qwen3.8	Selection	S	7:0	0.0156	0.188
Deploy.	Qwen3.8	Selection	E	6:0	0.0312	0.344
Deploy.	Qwen3.8	Termination	P	0:0	–	–
Deploy.	Qwen3.8	Termination	U	1:0	1	1
τ-bench	Gemma-31B	Selection	S	17:0	1.53e-05	0.000214
τ-bench	Gemma-31B	Selection	E	16:0	3.05e-05	0.000397
τ-bench	Gemma-31B	Termination	P	0:0	–	–
τ-bench	Gemma-31B	Termination	U	1:0	1	1
τ-bench	Qwen3.8	Selection	S	1:0	1	1
τ-bench	Qwen3.8	Selection	E	1:0	1	1
τ-bench	Qwen3.8	Termination	P	0:0	–	–
τ-bench	Qwen3.8	Termination	U	0:0	–	–

Findings and limits. Selection contrasts favor P in all eight comparisons. After correction, the four Gemma contrasts remain significant ($p_{\text{Holm}} = 6.10 \times 10^{-5}$ on Deployment and 2.14×10^{-4}/3.97×10^{-4} on retail). Qwen3.8 Deployment contrasts have raw $p = 0.015625$/0.03125 but adjusted $p = 0.1875$/0.34375; retail has only one discordant pair under either termination rule. We therefore report directional, not statistically established, cross-model replication. P has identical seed-level exact outcomes under S and E in all four cells. E gains one exact outcome over S in each of two U cells and changes none in the other two. These limited observed changes do not establish equivalence, absence of a termination effect, or absence of interaction. The descriptive differences-in-differences $(P_E - P_S) - (U_E - U_S)$ are 0, -5, -5, and 0 percentage points in table order; these are descriptive interaction contrasts. Resource use and non-exact defects can still change even when exact recovery is unchanged. For example, on Deployment with Gemma, U–E observes 48 states on average and reaches 1/20, while P–S observes 32 and reaches 20/20. This is inconsistent with an explanation based solely on a larger observed-state count, but does not remove every schedule or feedback-content difference. The result supports candidate-partitioned selection within this common-loop experiment, not equivalence to every historical protocol.

Reproduction. Run `python analysis/b16/paired_tests.py` from the project environment. It first rebuilds `rows.json` from four explicit result directories, checks all twenty seeds and common initial hashes, and produces `statistics.json` and Tables 11–12. The JSON records raw and adjusted tests and SHA-256 hashes of all 320 source artifacts. Aggregation uses stored artifacts only. The separate full-closure replay audit re-parsed and re-scored all 308 valid final contracts without disagreement; the twelve parse failures have no contract to re-score.

L Selection on two additional Deployment configurations

We test two author-designed configurations using the existing Deployment rules. We call the quota-stress configuration D2: it limits each resource quota to one while allowing allocations of up to three units. We call the option-filtering configuration D3: it offers one accepted and two rejected options for both region and tier. Their closures contain 5,440 and 11,712 states, with 20 and 22 accepting states, respectively. At depth 24 and a 15,000-state cap, neither is truncated; depth 28 reproduces both the state and transition sets. Every reference clause has an independent witness, and the reference DSL agrees with the environment on normal transitions. Two other proposed configurations exceeded the state cap and were not run with models.

The common loop compares candidate-partitioned selection with fresh uniform selection, with coverage disabled and stopping on a satisfied batch in both conditions. Uniform is not a fixed pool. Each model–configuration pair uses ten seeds (0–9), with one initial contract shared by the two conditions for each seed: 40 initial generations and 80 condition runs. Both conditions use $B = 48$, $Q = 4$, a 16,000-token run cap, and a 2,048-token completion cap. Initial generation is charged to each condition; subsequent inputs are reserved before requesting completion tokens. Requested batches contain 16 states; the Deployment adapter replenishes a depleted partition from the other side. This corrects the earlier no-replenishment description, not the executed protocol; both conditions retain their original results. The configuration checks, protocol, and four-comparison family were fixed before these model runs, after inspecting the earlier experiments.

Table 13: Additional Deployment configurations, exact/10. Partitioned versus fresh uniform selection under the same stopping rule. Holm correction covers all four rows.

Model	Configuration	Partitioned	Uniform	Raw p	Holm p
Gemma-4-31B	D2	10/10	0/10	0.001953	0.007813
	D3	10/10	0/10	0.001953	0.007813
Qwen3.8-27B	D2	2/10	0/10	0.500000	0.500000
	D3	3/10	0/10	0.250000	0.500000

All initial and final contracts parse; none of the 40 initial contracts is exact. Gemma's partitioned condition recovers all ten seeds in each configuration; its uniform condition stops after one batch, leaving median false accepts of 16 in D2 and 9 in D3. Across both models, 37/40 uniform runs stop on sampled satisfaction despite remaining closure defects. Qwen's partitioned conditions recover

2/10 and 3/10; these differences are not significant after correction. Two Qwen runs stop before sending a further request because input reservation leaves no completion budget. Both remain in the denominator; their recorded incomplete-usage flags are distinguished from transport failure. Stored full-closure scores were independently reproduced for all 80 runs.

Equal caps do not imply equal realized costs. Gemma averages 2 calls and 32 audited states under partitioning, versus 1 call and 16 states under uniform selection. The contrast measures the selection policy and its subsequent trajectory, not a pure information-quality effect at fixed realized cost. These are two additional configurations of one domain, not eighty independent tasks. Gemma has only 2 and 1 distinct initial contracts in D2/D3 (Qwen: 4 and 5); the seed repetitions are not pooled across configurations. The executed version, source snapshots, state/token ledgers, and paired tests are recorded in `analysis/g2_results_v2/`.

M Secondary graded metrics and round-by-round convergence

M.1 Metrics and aggregation

Exact recovery remains the primary endpoint. Equation 1 defines applicability IoU and CFR; both are computed per run and averaged over all 20 attempts, assigning zero when no evaluable final contract is produced. Valid yield and full per-cell results are retained in `supplement/metrics_detail.pdf`. PVR is averaged only over valid contracts with $N_+ - \mathrm{FR} > 0$, with the eligible count reported; otherwise it is N/A. A parsed contract with missing evaluation would leave its cell unresolved.

FR and PV count disjoint executable states, so $0 \le \mathrm{FR} + \mathrm{PV} \le N_+$ and $0 \le \mathrm{CFR} \le \mathrm{AppIoU} \le 100$. Per run,

$$\mathrm{AppIoU} - \mathrm{CFR} = 100\frac{\mathrm{PV}}{N_+ + \mathrm{FA}}, \qquad \mathrm{CFR} = \mathrm{AppIoU}(1 - \mathrm{PVR}_{\%}/100)$$

when PVR is defined. This product identity generally does not hold for cell means. Comparisons share the final evaluation support within each domain/protocol; domains are not pooled. IoU/PVR were added after the first CFR reanalysis; these descriptive summaries add no tests or changes to the original multiplicity families. CFR measures consistency with observed effects, not postcondition strength (Appendix T).

Reject-all baseline. A contract rejecting every state has $\mathrm{FA} = \mathrm{PV} = 0$, $\mathrm{FR} = D = N_+$, and $\mathrm{IoU} = \mathrm{CFR} = 0$. Its FAR is 0, FRR is 1, and PVR is N/A because no postcondition is checked. Domain-specific N_+ values are already listed in Table 1.

M.2 Exact recovery and uncertainty

Table 14 retains all 136 main-table results after the 58-run truncation-retry replacement policy and adds eight completed Calendar controls from the later suites (Appendices P, Q); these additions are descriptive and retain their historical outcomes. Each cell has 20 attempts, so its Wilson interval depends only on the exact count k. Full diagnostics for the 136 main cells are in `supplement/metrics_detail.pdf`.

M.3 Residual-failure distributions

Figure 6 diagnoses Active's valid inexact contracts under the main-table truncation-retry policy; full Direct/Active values are retained in `supplement/metrics_detail.pdf`. These conditional subsets differ across methods and do not replace the all-attempt comparisons.

M.4 Historical OFF diagnostics

The original OFF suites are distinct from the main-table ON conditions. Table 15 illustrates why exact counts, parsing yield, and residual defects must be read together. The full historical table is in `supplement/metrics_detail.pdf`; the earlier extended tables and exploratory paired defect tests remain in `supplement/appendix_graded_full.tex`. Their FAR/FRR/PVR

Table 14: Exact successes k out of 20 under the main-table truncation-retry policy; Active is bold. Stars and diamonds identify base decoding suites, with selected retries using enlarged limits. Calendar dashes: no matched runs available. Retail Random n/a: the legacy iterative policy is excluded, not a zero result. The lower panel maps k to its Wilson 95% interval (%).

Model	Direct	Self-Refine	Random	Fixed-pool	Active
Deployment					
Gemma-4-31B	0	0	2	3	**20**
Qwen3.8-27B	0	0	0	1	**11**
Qwen2.5-32B	4	4	3	2	**7**
Qwen2.5-14B	0	0	1	1	**16**
Gemma-4-26B-A4B*	0	0	0	0	**11**
Qwen3.6-35B-A3B*	0	0	3	1	**20**
Muse-Glimmer-30B◇	4	10	3	5	**20**
Qwen2.5-Coder-32B	0	0	1	0	**3**
Calendar					
Gemma-4-31B	3	0	–	–	**20**
Qwen3.8-27B	3	6	–	–	**13**
Qwen2.5-32B	11	12	–	–	**13**
Qwen2.5-14B	0	0	–	–	**0**
Gemma-4-26B-A4B*	0	0	1	5	**13**
Qwen3.6-35B-A3B*	1	3	1	4	**14**
Muse-Glimmer-30B◇	13	18	6	18	**20**
Qwen2.5-Coder-32B	4	3	5	10	**14**
Retail					
Gemma-4-31B	0	0	n/a	3	**19**
Qwen3.8-27B	0	0	n/a	1	**1**
Qwen2.5-32B	0	0	n/a	0	**0**
Qwen2.5-14B	0	0	n/a	0	**1**
Gemma-4-26B-A4B*	0	0	n/a	0	**0**
Qwen3.6-35B-A3B*	0	0	n/a	0	**1**
Muse-Glimmer-30B◇	0	0	n/a	0	**1**
Qwen2.5-Coder-32B	0	0	n/a	0	**0**
Shopping (relaxed)					
Gemma-4-31B	0	0	0	5	**20**
Qwen3.8-27B	0	0	0	6	**14**
Qwen2.5-32B	0	0	0	0	**0**
Qwen2.5-14B	0	0	0	0	**0**
Gemma-4-26B-A4B*	0	0	0	10	**8**
Qwen3.6-35B-A3B*	0	0	0	12	**20**
Muse-Glimmer-30B◇	0	0	0	8	**19**
Qwen2.5-Coder-32B	0	0	0	0	**0**

k	Wilson 95% CI	k	Wilson 95% CI	k	Wilson 95% CI
0	[0.0–16.1]	7	[18.1–56.7]	14	[48.1–85.5]
1	[0.9–23.6]	8	[21.9–61.3]	15	[53.1–88.8]
2	[2.8–30.1]	9	[25.8–65.8]	16	[58.4–91.9]
3	[5.2–36.0]	10	[29.9–70.1]	17	[64.0–94.8]
4	[8.1–41.6]	11	[34.2–74.2]	18	[69.9–97.2]
5	[11.2–46.9]	12	[38.7–78.1]	19	[76.4–99.1]
6	[14.5–51.9]	13	[43.3–81.9]	20	[83.9–100.0]

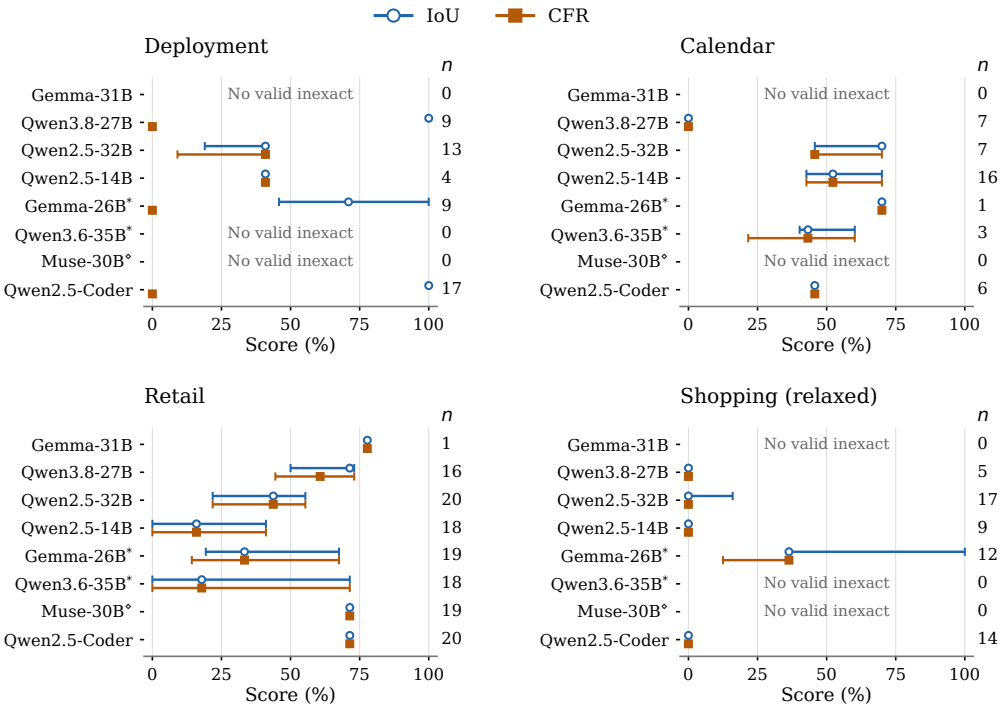

Figure 6: Active's residual failures in all 32 main-table model–domain conditions after the 58-run truncation-retry replacement policy. Points are medians; bars span the 25th–75th percentiles (linear interpolation), not confidence intervals. n counts valid inexact contracts; no mark is drawn when $n = 0$. Parse failures and exact contracts are excluded. Stars and diamonds identify the base decoding suites; selected retries use enlarged limits.

values are medians over eligible valid contracts, not the means reported for the main-table protocols. Parse failures remain in the exact denominator. Residual-defect tests are exploratory and do not establish equivalence.

Table 15: Illustrative diagnostics from the original OFF suites, not the starred ON conditions. Active unless stated; k is exact out of 20, pf counts parse failures, and D medians use valid contracts. Cases illustrate distinct interpretation issues, not a ranking; all 44 historical rows are retained in the detailed supplement.

Domain / model	k	pf	Diagnostic
Calendar / Gemma-26B	1	19	Only one parsed contract; its zero defects do not describe the other 19 attempts.
Deployment / Qwen3.8-27B	11	0	Median $D = 0$ coexists with nine inexact runs.
Shopping / Qwen2.5-32B (Random)	0	0	Reject-all behavior: FAR $= 0$, FRR $= 1$, PVR is N/A.
Retail / Qwen3.8-27B	1	3	Median $D = 4$ still leaves most attempts inexact.

M.5 Round-by-round convergence over the full closure

Figure 7 retains the original OFF trajectories. Early-terminated runs carry forward their final evaluation; initial parse failures are excluded. A shared prompt does not ensure identical initial candidates: four of twenty Deployment/Qwen3.8 seeds differed across arms before refinement, for an unresolved runtime reason. The extended supplement retains all trajectories and paired tests.

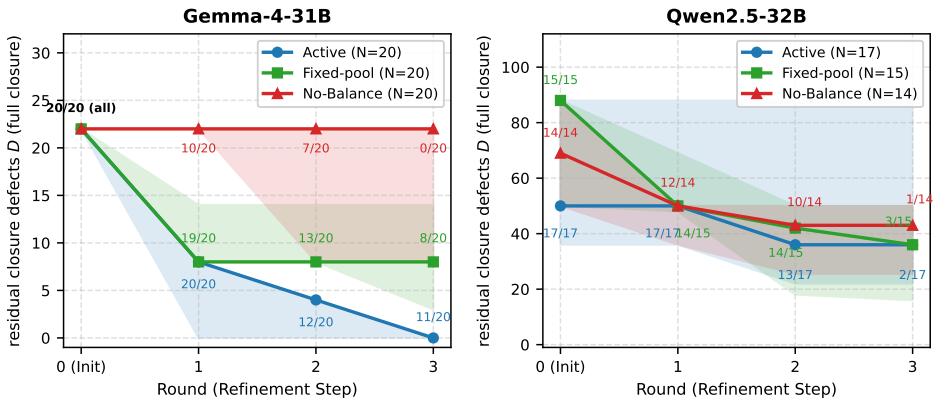

Figure 7: Full-closure residual defects $D = \mathrm{FA} + \mathrm{FR} + \mathrm{PV}$ in original OFF runs. Lines: medians; bands: interquartile ranges; fractions: runs still active at each round. Stopped runs retain their final evaluation. Gemma has $N = 20$ per arm; Qwen2.5-32B has $N = 17$ Active, 15 Fixed-pool, and 14 No-Balance after excluding initial parse failures.

N Qwen2.5-32B failure analysis

The twenty Active-CEGIS runs are the original Qwen2.5-32B easy-v4 cell; full artifact paths and recorded hashes are retained in the source supplement. Seventeen runs end `inexact` and three end in parse failure (empty `rounds`); none ends with the sampled batch satisfied.

`stopped_because`	Runs (of 20)
`duplicate_model_output`	9
`query_budget_exhausted`	8
`vacuous_candidate_unrepaired`	3

The principal semantic failure is a wrong-variable repair. In seed 0, a false accept occurs after `order_status` is already `placed`, but the proposed revision tests `order_payment_method`, a field populated only by successful `place_order`. That clause therefore rejects every accepting state, while the counterexample's missing order-status condition is not added. The next revision is byte-identical, so the run stops on `duplicate_model_output`. This is an illustrative failure trace, not evidence that the changing sampled batch improves or worsens revision quality.

The associated trajectory summary is intentionally limited to the stopping and semantic observations above: changing `states_checked` changes the evidence set, so round-to-round FA/FR/PV totals and rates are not a same-state measure of revision quality. The limitation is consequently that the experiment diagnoses non-repair and termination behavior, but does not establish a causal effect from those batch-dependent trajectories.

O Additional model failure analyses

Failures differ by domain and decoding condition. In the original reasoning-off relaxed-Shopping corpus, Qwen2.5-14B has 72/160 final parse failures; 68 of its 88 parsed preconditions test the post-order payment field and reject all eight accepting states. Under the same reasoning-off condition, Qwen3.6 has 141/160 parse failures, including repetition-induced truncation and numeric arguments to `is_true`. Its separate repetition-penalty experiment reduces truncation but reaches only 1/160 exact at penalty 1.1. These are diagnostic conditions, not evidence that larger budgets or reasoning can never help.

On Deployment, Qwen3.6 instead has 22/160 parse failures and 13 exact outcomes across methods. On retail it often rejects all accepting states (19/19 parsed Direct runs, but only 8/12 parsed Active runs). Thus neither a single proposal-failure explanation nor active parameter count alone accounts for its behavior. Full legacy trajectories and decoding diagnostics are preserved in the source supplement. The follow-up reasoning and adaptive-budget conditions are summarized with explicit markers below.

P Additional decoding conditions

The historical tables in this appendix retain their original outcomes and base decoding limits; they are not retroactively updated by the main-table truncation-retry policy. The base main suites use the later reasoning-enabled MoE conditions consistently across domains. The original OFF rows are retained below as decoding controls; their existing statistical families remain attached to OFF, while ON is reported descriptively.

Table 16: Original reasoning-OFF MoE controls (without adaptive retries), exact/20; these are distinct from the starred reasoning-ON main-table cells. Completion/run caps are 2,048/16,000. D: Direct; SR: Self-Refine; A: Active. These are the original runs, before adaptive retries.

Model (OFF)	Domain	D	SR	A
Gemma-26B	Shopping	0	0	0
	Deployment	0	0	0
	Calendar	0	0	1
	Retail	0	0	0
Qwen3.6	Shopping	0	0	0

The relocated Gemma-26B OFF selector rows have (Active, No-Balance, No-Coverage, Uniform) counts of (0,0,0,0) on both relaxed Shopping and Deployment, and (1,0,8,2) on Calendar. Calendar's original paired tests against Active are No-Balance: n.s., 1:0 discordant; No-Coverage: $p_{\mathrm{Holm}} = 0.047$; Uniform: n.s., 1:2 discordant ($m = 3$). The full original OFF statistics remain in the source supplement (Appendix I); they are not tests of the ON rows below.

Gemma-26B denotes Gemma-4-26B-A4B and Qwen3.6 denotes Qwen3.6-35B-A3B. Starred rows use vLLM 0.28.0, temperature 0.2, top-p 0.95, and $B = 48, Q = 4$. The API field `reasoning_effort=low` is recorded; the numerical control is the requested 1,024-token rea-

Table 17: Historical MoE decoding checks, exact/20. *: reasoning enabled at the base requested reasoning/completion/run caps 1,024/3,072/20,096; these printed values precede the separate main-table retry replacement. †: reasoning off, with the distinct adaptive retry policy below. D: Direct; SR: Self-Refine; R: Random; F: fixed-pool; A: Active; NB/NC: No-Balance/No-Coverage; U: Uniform. Retail R retains its historical iterative-uniform implementation.

Model	Domain	D	SR	R	F	A	NB	NC	U
Gemma-26B*	Shopping	0	0	0	10	8	6	11	7
	Deployment	0	0	0	0	6	0	1	0
	Calendar	0	0	1	5	13	0	15	1
	Retail	0	0	0	0	0	0	0	0
Qwen3.6*	Shopping	0	0	0	13	20	2	20	7
	Deployment	0	0	3	1	14	2	15	2
	Calendar	1	3	1	4	14	0	16	5
	Retail	0	0	0	0	1	1	3	0
Gemma-26B†	Shopping	0	0	0	0	0	0	0	0
Qwen3.6†	Shopping	0	1	0	5	5	2	8	3

soning cap, not a model-independent interpretation of "low". All twenty seeds remain in each denominator, including parsing and runner errors.

The daggered policy was added after inspecting output truncation. An original OFF run uses a 2,048-token completion cap and 16,000-token run budget. If any response ends with `finish_reason=length`, the entire run is restarted once with caps 3,072/20,096 and reasoning disabled. All methods use the same trigger; the new result replaces the old one regardless of success. The 260 retries and 60 untouched runs form 320 method–seed outcomes. Cost accounting includes both attempts; $B = 48, Q = 4$ applies per attempt rather than to their combined cost. This is not a uniform 3,072-token experiment, and unused tokens are not charged as usage.

Qwen's adaptive OFF policy reaches 5/20 on Shopping with Active, versus 20/20 in the starred condition; Gemma reaches 0/20 versus 8/20. The comparison addresses truncation as an alternative explanation and measures a combined change in reasoning, budgets, and execution conditions. These are descriptive follow-up results, not additional members of the earlier statistical families. No-Coverage also numerically exceeds Active in several starred cells (Gemma Shopping 11 versus 8 and Calendar 15 versus 13; Qwen Deployment 15 versus 14), showing mixed effects of coverage while holding candidate partitioning present in both arms. The two reasoning manifests and the adaptive per-seed ledger are retained under `analysis/{gemma26,qwen36}_reasoning_20260909/` and `analysis/shopping_off_budget_20260909/`, respectively.

Q Additional model evaluations

These later suites contain 660 Coder and 660 Muse records over seeds 0–19, including the twenty Muse fixed-pool runs added on September 12. Retail's legacy `sampled_cegis` is partitioned uniform, not fixed-pool; its twenty records per model are excluded from this table (both score 0/20). Both models have separate fixed-pool arms. Retail R is iterative uniform. The table thus summarizes 1,280 outcomes, keeping the legacy partitioned arms separate.

These historical values retain the base-suite artifacts. Both suites record vLLM 0.28.0, temperature 0.2, top-p 0.95, $B = 48, Q = 4$, a 2,048-token completion cap and a 16,000-token run budget, with context-dependent clamping and JSON-schema requests. Coder uses the OFF configuration. Muse's launch configuration sets `reasoning_strength=low`; requests inherit that server default. Its responses contain reasoning text even though the reported reasoning-token counter is zero; separate reasoning cost is therefore unmeasured by that counter. These results are descriptive follow-ups, not additions to the earlier Holm families or a matched model-only comparison with the main tables.

Muse reaches 19–20/20 with Active on the three synthetic domains but only 1/20 on retail. Coder improves numerically over Direct on Calendar (14/20 versus 4/20) yet remains at zero throughout Shopping and retail. These suites extend successful recovery beyond Gemma and show domain-

Table 18: Historical base-suite additional-model results, exact/20; these values precede the separate main-table truncation-retry replacement, and all failures remain in the denominator. D: Direct; SR: Self-Refine; R: Random; F: fixed-pool; A: Active; NB/NC: No-Balance/No-Coverage; U: Uniform. Shopping denotes the relaxed variant. Muse uses its server-default `low` reasoning setting, not the OFF condition or the separately capped MoE condition.

Model	Domain	D	SR	R	F	A	NB	NC	U
Qwen2.5-Coder-32B	Shopping	0	0	0	0	0	0	0	0
	Deployment	0	0	1	0	3	0	3	2
	Calendar	4	3	5	10	14	3	13	4
	Retail	0	0	0	0	0	0	0	0
Muse-Glimmer-30B	Shopping	0	0	1	7	19	1	18	6
	Deployment	4	10	2	5	20	6	19	8
	Calendar	13	18	6	18	20	13	20	15
	Retail	0	0	0	0	1	0	1	0

specific outcomes for each model. Final unparsed counts for Coder are 53/160, 15/160, 3/160, and 0/180 across Shopping, Deployment, Calendar, and retail; Muse's counts are 11/160, 4/160, 0/160, and 3/180. In particular, Coder's retail floor persists without final parse failures, locating this failure beyond final-output parsing.

The input-file hashes, cell-level protocol checks, and reconciliation of stored exact outcomes with FA/FR/PV are retained in `analysis/additional_models_20260910/verification.json`, with the September 12 fixed-pool addition in `analysis/table_gaps_20260912/verification.json`. This reconciliation checks stored evaluations and protocol metadata.

R Completed candidate-pool and hosted-model follow-ups

This appendix reports completed follow-ups that are separate from the main comparison families. All candidate-pool runs use 20 paired seeds, a 48-state budget, four logical calls including the initial proposal, a 16,000-token budget, temperature 0.2, and top-p 0.95. The candidate pools are nested seeded prefixes of pre-enumerated closures; consequently, these experiments measure selection within an available pool rather than the cost of acquiring states online.

R.1 Sensitivity to the candidate-pool size

Table 19 shows that partitioned selection retains a large advantage on Deployment even with only 200 of 11,712 closure states. For retail, Gemma's advantage becomes pronounced at 500 states, whereas Qwen3.8 does not show a statistically resolved advantage at any tested pool size. The full-pool columns are simultaneous controls collected under the same new protocol, rather than copies of the earlier common-loop factorial study (Appendix K).

The new Deployment suite and the common-loop factorial study (Appendix K) share the nominal budgets and selection/stop conditions, but use separately generated initial proposals and distinct protocol hashes, accounting, context reservation, and serving records. We therefore interpret only within-suite paired contrasts. During the read-only audit, seven retail artifacts were found to serialize a truncated last response although evaluation used the preceding valid incumbent; re-evaluating each recorded incumbent reproduced its stored FA, FR, PV, and exact outcome. The paths and reconstruction are recorded in `analysis/completed_evidence_20260921/retail_incumbent_trace.json`.

R.2 Hosted JSON-mode suites

Hosted runs request JSON objects and use the same finite evaluators, but API sampling seeds are not sent: trial identifiers seed local state selection and denote repeated runs, not different tasks. Table 20 keeps the original 20-run cohort, the new 40-run Deployment cohort, and the 20-run Flex suite visibly separate. The strongest positive observation is nano Active at 31/60 across the two

Table 19: Candidate-pool sensitivity, exact recovery out of 20 paired seeds. U is uniform selection and P is candidate-partitioned selection. Adjusted p-values are from paired exact McNemar tests, with separate Holm families for Deployment ($m = 8$) and retail ($m = 6$).

		Small		Medium		Large		Full	
Domain	Model	U	P	U	P	U	P	U	P
Deployment	Gemma-4-31B	3	12	1	19	3	20	3	20
	Qwen3.8-27B	1	8	1	19	1	20	2	20
Retail	Gemma-4-31B	2	9	1	18	–		1	19
	Qwen3.8-27B	1	1	1	5	–		0	3

Deployment pool sizes are 200, 1,000, 5,000, and 11,712; retail uses 200, 500, and 1,294. Deployment Holm-adjusted p-values, in increasing pool order, are Gemma .0078, 5.34×10^{-5}, 6.10×10^{-5}, and 6.10×10^{-5}; Qwen .0156, 5.34×10^{-5}, 3.05×10^{-5}, and 5.34×10^{-5}. Retail values are Gemma .156, 7.63×10^{-5}, and 4.58×10^{-5}; Qwen 1.0, .656, and .656. All final evaluations use the full closure and are withheld from synthesis feedback.

Table 20: Hosted Deployment exact-recovery counts out of the cohort size shown in each suite row. Parentheses give the descriptive original-plus-extension count out of 60. Flex is not pooled with the default-service cohorts.

Suite	Model	Direct	Fixed-pool	Active	No-Balance
Original ($n = 20$)	GPT-5.4 mini	0	0	0	1
	GPT-5.6 luna	1	3	2	3
	GPT-5.4 nano	0	2	12	0
Extension ($n = 40$)	GPT-5.4 mini	0 (0/60)	1 (1/60)	0 (0/60)	0 (1/60)
	GPT-5.6 luna	0 (1/60)	4 (7/60)	5 (7/60)	0 (3/60)
	GPT-5.4 nano	0 (0/60)	2 (4/60)	19 (31/60)	2 (2/60)
Flex ($n = 20$)	GPT-5.6 terra	0	1	20	3

default-service cohorts and Terra Active at 20/20 within its Flex suite. Luna's pooled Active and Fixed-pool counts are both 7/60, so its result does not support a selector advantage.

The original hosted suite also completed all four methods on retail (1,294 states; early stop on sample satisfaction), rather than only on Deployment. Direct/Fixed-pool/Active/No-Balance exact counts are respectively $0/0/1/0$ for GPT-5.4 mini and $0/0/0/0$ for GPT-5.6 luna; the separate nano suite records $0/0/0/0$ for GPT-5.4 nano. Thus the strong hosted Active result is domain-dependent: in these earlier suites it appears on Deployment for nano and terra, while the original retail cohorts remain at or near floor; the gap-fill cohort below extends it to relaxed Shopping and to nano Calendar (10/20 versus 0/20 for Self-Refine, $p_{\mathrm{Holm}} = 0.051$). These counts are direct aggregations of the 240 retail artifacts in the original and nano JSON-mode suites; the fixed-pool artifact method is `sampled_cegis_fixed`.

The same hosted protocol was extended to Calendar and relaxed Shopping. Table 21 retains IoU and contract fulfillment rate (CFR) even when exact recovery is at floor. Mini Active reaches 3/20 exact on Calendar; nano has no exact recovery there, but its No-Balance arm reaches the largest observed IoU/CFR in that model–domain block. On relaxed Shopping, nano Active has the largest IoU and CFR among its four arms, while all exact counts remain zero.

In a separate mini Shopping ablation, preserving the default sampler while enabling low reasoning yields 2/20 exact, 26.2% IoU, and 20.0% CFR, compared with 0/20, 4.2%, and 3.3% under default-sampling reasoning-off. The earlier temperature-0.2/top-p-0.95 reasoning-off arm is 0/20, 0.8% IoU, and 0.0% CFR. This earlier reasoning-off arm is a separate 20-run baseline collected for the ablation; it is not the 20-run Active cell with 3.6% IoU and 0.8% CFR in Table 21. Low reasoning also recovers the missing `order_status==none` precondition in 7/20 runs versus 0/20 in both reasoning-off arms. This three-arm, 20-run-per-arm experiment is descriptive and is not merged with the four-method suites.

Gap-fill cohort. A later cohort adds Self-Refine and a shared-initial Active run to every hosted model–domain cell and completes all five methods for Terra's retail, Calendar, and Shopping cells and for Luna's Calendar and Shopping cells: 47 cells of 20 trials, 940 runs in total. Within each model, domain, and trial, one initial response is generated and shared by every method, so its initial

Table 21: Hosted Calendar and relaxed-Shopping results ($n = 20$ per cell). Each entry is an exact-recovery count out of 20, followed by IoU and CFR in percent.

Model	Domain	Metric	Direct	Fixed-pool	Active	No-Balance
GPT-5.4 mini	Calendar	Exact; IoU; CFR	0; 10.8; 6.5	1; 29.5; 9.0	3; 28.4; 19.3	0; 18.1; 9.0
	Shopping	Exact; IoU; CFR	0; 0.0; 0.0	0; 7.8; 7.0	0; 3.6; 0.8	0; 2.5; 0.0
GPT-5.4 nano	Calendar	Exact; IoU; CFR	0; 27.6; 18.3	0; 29.6; 20.2	0; 30.1; 20.6	0; 34.2; 24.8
	Shopping	Exact; IoU; CFR	0; 8.0; 0.0	0; 22.1; 10.8	0; 25.2; 11.5	0; 23.1; 3.2

Table 22: Hosted gap-fill cohort: exact recovery out of 20 with mean CFR (%) in parentheses. All methods within a model, domain, and trial share one initial response. Dashes mark methods not rerun in this cohort; earlier suites report those cells.

Model	Domain	Direct	Self-Refine	Fixed-pool	**Active**	No-Balance
GPT-5.6 terra	Deployment	–	0 (0.0)	–	**20 (100.0)**	–
	Retail	0 (71.4)	0 (28.6)	0 (71.4)	**0 (71.4)**	0 (71.4)
	Calendar	19 (95.0)	20 (100.0)	19 (95.0)	**20 (100.0)**	19 (95.0)
	Shopping	0 (0.0)	0 (0.0)	9 (53.6)	**19 (95.6)**	2 (11.8)
GPT-5.6 luna	Deployment	–	1 (5.0)	–	**1 (5.0)**	–
	Retail	–	0 (0.0)	–	**0 (3.6)**	–
	Calendar	0 (0.0)	0 (21.0)	1 (9.6)	**0 (21.0)**	0 (4.6)
	Shopping	0 (0.0)	0 (0.0)	4 (29.0)	**10 (61.9)**	1 (6.8)
GPT-5.4 nano	Deployment	–	0 (40.1)	–	**14 (83.2)**	–
	Retail	–	0 (15.8)	–	**0 (35.4)**	–
	Calendar	–	0 (31.9)	–	**10 (54.6)**	–
	Shopping	–	0 (0.0)	–	**0 (12.9)**	–
GPT-5.4 mini	Deployment	–	0 (0.0)	–	**0 (0.0)**	–
	Retail	–	0 (1.4)	–	**0 (18.8)**	–
	Calendar	–	0 (8.3)	–	**2 (17.8)**	–
	Shopping	–	0 (0.0)	–	**0 (4.0)**	–

proposals differ from the earlier suites and the cohort is reported separately (Table 22). Settings otherwise follow the earlier hosted protocol: $B = 48$, $Q = 4$, reasoning off, temperature 0.2, JSON object mode, and no API sampling seed; Terra uses Flex service and the other models default service. Paired exact tests against Active form one Holm family of 31 comparisons defined after the runs, so they are exploratory.

The clearest results replicate the local pattern in a different model family. Terra Active reaches 20/20 on Deployment against 0/20 for Self-Refine ($p_{\mathrm{Holm}} = 5.9 \times 10^{-5}$), repeating its earlier Flex result. On relaxed Shopping it reaches 19/20 against 0/20 for Direct and Self-Refine ($p_{\mathrm{Holm}} = 1.1 \times 10^{-4}$ each) and 2/20 for No-Balance ($p_{\mathrm{Holm}} = 4.3 \times 10^{-4}$); its Fixed-pool contrast (9/20) is 10:0 discordant but not significant after correction ($p_{\mathrm{Holm}} = 0.051$). Nano Active reaches 14/20 on Deployment against 0/20 for Self-Refine ($p_{\mathrm{Holm}} = 0.0033$), in line with its 31/60 in Table 20. Three cells limit the interpretation. On Calendar, Terra Direct already reaches 19/20, leaving little room for feedback to show an effect. On retail, every hosted model stays at 0/20 under every method, and the residual errors are false rejects (Appendix J). Luna's Deployment Active attains 85.0% IoU but 5.0% CFR, with postcondition violations in 16 of 20 runs: it recovers applicability but not the effects. Aggregation, hashes of all 940 artifacts, and the paired tests are reproduced by `analysis/openai_api_20260926/compare.py`.

Shared-initial diagnostic. The post hoc diagnostic crosses the model producing the archived initial proposal (columns) with the model performing Active revision (rows), with 20 trials per cell:

Revision model	Mini initial	Luna initial	Nano initial
GPT-5.4 mini	0	3	13
GPT-5.6 luna	1	3	12
GPT-5.4 nano	5	11	11

Nano-origin proposals support substantially more successful revision by all three models, while nano also repairs some mini- and luna-origin proposals. This diagnostic isolates a common starting point only; it is not pooled with the main comparison and does not identify a training or scale effect.

Later Calendar controls. The separately completed Qwen2.5-14B Calendar controls yield Random 0/20 and Fixed-pool 0/20 exact. Across the 40 runs, 34 final contracts parse and six are parse failures. These two descriptive cells complete previously absent controls and do not enter an existing test family.

Across all hosted suites, full-closure evaluation is withheld from synthesis feedback. The default-service, Flex, reasoning, shared-initial, and gap-fill suites are kept separate because model, service tier, decoding, sample size, or diagnostic status changes across them.

S DOMAIN VOCABULARY AND WHAT THE PROMPT DISCLOSES

Representative prompt templates are reproduced in Appendix E; this appendix summarises two facts about them that bear on how the results should be read: which operators the contract DSL offers in each domain, and which reference clauses the prompt makes easy by naming a set or supplying a predicate. Within a matched protocol, methods share the initial domain description and grammar, while feedback prompts differ. The supplied vocabulary limits what can be claimed about discovery of previously unspecified rules.

Operators by domain. The shared grammar is `const`, `and`, `or`, `not`, `is_true`, `is_null`, `is_empty`, `eq`, `ne`, `lt`, `le`, `gt`, `ge`, `in_set`, and `unchanged`, with the terms `var` and `const`. Table 23 lists the additions. The operators in the last two rows each encode one reference clause (or, for the two `_snapshot_matches` predicates, a domain-specific transition check).

Table 23: Domain-specific additions to the shared grammar. Operators marked * correspond one-to-one to a reference clause.

Domain	Formula operators	Term operators
Shopping (price)	`covers`	`count`, `total`, `difference`, `subtract`
Shopping (basic)	`covers`	`count`, `total`, `difference`
Deployment	`covers`	`difference`, `union`, `get`, `total`
Calendar	`overlaps`, `intersects`, `contains`, `capacity_sufficient`*, `room_supports_type`*, `no_room_conflict`*, `no_attendee_conflict`*, `scheduled_snapshot_matches`*	`count`, `room_capacity`, `room_supports_video`
τ-bench retail	`contains`, `items_in_order`*, `valid_variants`*, `items_available`*, `sufficient_balance`*, `exchange_snapshot_matches`*	`count`, `price_difference`

What the prompt discloses per clause. Table 24 classifies every reference precondition clause by how the prompt supports it. *Set named*: the prompt lists the valid and rejected options by value and the grammar exposes them as named sets, so the clause is one `in_set` over a named set. *Predicate*: the grammar has a dedicated operator for the clause. *Composed*: the clause must be written from generic operators over state variables. *Hidden*: the prompt lists only the selectable options or nothing about the rule, so the accepted subset or threshold must be inferred from executions. Only price-aware Shopping has hidden clauses; its prompt supplies a neutral JSON envelope rather than domain-specific postcondition clause examples. In the other four listed domain settings, the original prompts show postcondition clauses of the reference's form as syntax examples (Appendix E), so substantial effect structure is supplied. The order-neutral τ-bench protocol is an exception: it removes the postcondition example block.

For price-aware Shopping the hidden clauses are (4) accepted payment methods, (5) serviceable addresses, (7) coupon eligibility (accepted codes, minimum subtotal, excluded items), and (8) payment capacity (wallet balance and card limit); the composed clauses are logged-in, empty order

Table 24: Reference precondition clauses by prompt support.

Domain	Clauses	Set named	Predicate	Composed	Hidden
Shopping (price), all variants	8	0	0	4	4
Shopping (basic)	6	2	0	4	0
Deployment	6	2	0	4	0
Calendar	10	3	4	3	0
τ-bench retail	8	2	4	2	0

slot, non-empty cart, and stock coverage. Basic Shopping names the accepted payment methods and serviceable addresses in the prompt, and its illustrative example contract already contains three correct precondition clauses (logged in, non-empty cart, payment in the accepted set), which makes the ceiling result difficult to interpret as discovery of hidden rules. In Deployment the named sets are the valid regions and cluster tiers; in Calendar the valid slots, rooms, and meeting types; in τ-bench the valid orders and payment methods. The Calendar predicates cover capacity, video support, room conflict, and attendee conflict; the τ-bench predicates cover item membership, variant validity, availability, and gift-card balance.

These tables distinguish assembly with supplied vocabulary from recovery of undisclosed rules. In basic Shopping, Deployment, Calendar, and τ-bench, prompts supply substantial clause structure, although direct generation can still fail. Price-aware Shopping additionally requires inferring hidden rules. In the tested configurations, exact recovery is observed only on the relaxed variant, whose closure does not distinguish the reference from a weaker contract (Appendix T). This association does not isolate prompt disclosure as the cause of the performance difference. On the identifiable variant (Section 7) the loop recovers the composed clauses and, in 8 of 18 runs, the hidden payment set, but the hidden address set in 2 of 18 and the coupon exclusion in none.

Table 25: Extensional clause recovery by prompt-support group. Each cell is Direct $\rightarrow$ Active-CEGIS recovered clause–run pairs; denominators are parsed runs times independently testable clauses in the group. –: no clause in the group.

Domain	Model	Composed	Set named	Predicate	Hidden
Shopping (relaxed)	Gemma-4-31B	60/80 $\rightarrow$ 80/80	–	–	40/40 $\rightarrow$ 40/40
Shopping (relaxed)	Qwen3.8-27B	12/80 $\rightarrow$ 68/68	–	–	10/40 $\rightarrow$ 34/34
Deployment	Gemma-4-31B	60/80 $\rightarrow$ 80/80	40/40 $\rightarrow$ 40/40	–	–
Deployment	Qwen3.8-27B	71/80 $\rightarrow$ 79/80	40/40 $\rightarrow$ 40/40	–	–
Calendar	Gemma-4-31B	43/60 $\rightarrow$ 60/60	60/60 $\rightarrow$ 60/60	80/80 $\rightarrow$ 80/80	–
Calendar	Qwen3.8-27B	60/60 $\rightarrow$ 60/60	60/60 $\rightarrow$ 60/60	80/80 $\rightarrow$ 80/80	–
τ-bench	Gemma-4-31B	0/40 $\rightarrow$ 39/40	40/40 $\rightarrow$ 40/40	80/80 $\rightarrow$ 80/80	–
τ-bench	Qwen3.8-27B	6/34 $\rightarrow$ 22/34	33/34 $\rightarrow$ 33/34	68/68 $\rightarrow$ 68/68	–

The clearest gains are on composed clauses omitted by Direct: the empty order or deployment slot, Calendar's no-active-meeting condition, and τ-bench's delivered-order and equal-item-count checks. Named-set and dedicated-predicate clauses are generally already recovered before feedback. The relaxed Shopping hidden column covers only payment and address witnesses; coupon eligibility and payment capacity have empty witness sets there, so its apparent recovery does not establish discovery of the full hidden rules.

T Closure discrimination and postcondition audit

Throughout this appendix, a reference clause c has a *witness* in a closure when some reachable state satisfies every other clause and fails c. A clause with no witness is never the sole reason a state is rejected, so removing only that clause from the reference precondition leaves its accepted set unchanged on the closure. This is precondition indistinguishability, not a guarantee for an otherwise arbitrary candidate contract. All numbers below are recomputed from the environment code by `scripts/closure_discrimination_audit.py`.

The relaxed variant. Per-clause witness counts on the relaxed variant (288 states):

Table 26: Clause identifiability over each closure. "Untested" counts reference clauses with no witness.

Domain	States	Accepting	Clauses	Untested
Shopping (price), full	8,064	13	8	0
Shopping (price), reduced	462	13	8	0
Shopping (price), relaxed	288	8	8	2
Shopping (price), identifiable	864	8	8	0
Shopping (basic)	4,752	22	6	0
Calendar	1,676	35	10	0
Deployment	11,712	88	6	0
τ-bench retail	1,294	14	8	0
τ-bench retail (W2-deliverable)	4,500	42	8	0

Clause	Witnesses	Clause	Witnesses
`logged_in`	8	`serviceable_address`	8
`empty_order_slot`	14	`sufficient_stock`	2
`nonempty_cart`	1	`coupon_eligible`	**0**
`accepted_payment`	8	`payment_capacity`	**0**

`coupon_eligible` takes both truth values over the closure but never independently of the others, and `payment_capacity` is constant (the wallet balance and card limit are 10,000 and never bind), so both have zero witnesses. Two further clauses collapse for a different reason: the configuration offers a single payment method, which is accepted, and a single address, which is serviceable, so `payment_method` $\in$ `accepted` and `address` $\in$ `serviceable` are equivalent to non-null tests. Evaluated over all 288 states, the eight-clause reference and the six-clause weakening that drops the two untested clauses and replaces the two set memberships by non-null tests accept exactly the same states. The contracts recovered on this variant do use the weaker forms. The identifiable variant (864 states) removes all four degeneracies; its witness counts are 8, 21, 2, 12, 16, 1, 2, and 4 in reference-clause order (1–8), so no clause is untested.

Postcondition strength. Section 3 defines PV against the after-state the environment actually produces, so a postcondition of $\top$ scores $\mathrm{PV} = 0$; we verified this by replacing a recovered postcondition with $\top$ and re-scoring, which leaves the cell exact. Among the 243 closure-exact artifacts in the G1 cross-method manifest (94 model–domain–method cells, 20 seeds each; 1,880 runs), none has a literal constant-true postcondition. Each inclusion is reconciled against the stored outcome and FA/FR/PV metrics, and the source-file hashes and paths are listed in `analysis/integrated_correction_20260917/step2/exact_postcondition_manifest.json`. This rules out that literal form in this defined cohort, not other semantically weak postconditions. We next audit output faults in a narrower Active cohort.

T.1 Output-fault rejection audit

This output-fault audit covers the 153 closure-exact contracts among 420 Active runs in the recorded main-domain cell manifest (21 domain–model cells, 20 seeds each). The remaining 189 inexact and 78 unparseable runs are outside this conditional audit; this is not a comparison of methods or a success rate over all runs. These 153 records contain 88 distinct normalised JSON contracts, not necessarily 88 semantically distinct formulas. All retain $\mathrm{FA} = \mathrm{FR} = \mathrm{PV} = 0$ when replayed.

Before scoring candidate postconditions, we fixed a deterministic fault generator. For every successful target-action input, it replaces one top-level successor field with another value observed for that field in the closure, adding boolean flips and nonnegative numeric neighbours (± 1). No-ops and duplicates are removed. A fault is a successor different from the actual deterministic output, independently of whether a candidate or reference rejects it. Composite fields are replaced as whole observed objects. The resulting set contains 145 normal transitions and 4,431 faulty transitions across the four domains. Each applicable contract receives the same domain-specific set; Table 27 counts contracts rejecting *all* tested faults, not the fraction of individual faults rejected.

Table 27: Output-fault rejection, conditional on existing exact Active runs. "All faults" includes output changes outside the reference's specified effects; the last column restricts to faults that the reference rejects.

Domain	Contracts	Faults/contract	All faults	Reference faults
Shopping (relaxed)	31	400	30/31	30/31
Deployment	54	2,464	41/54	41/54
Calendar	47	1,085	47/47	47/47
τ-bench retail	21	482	0/21	20/21

Every candidate accepts every normal transition. Constant-true postconditions accept all faults, as expected. The Python references reject all faults except on retail, where 42/482 alter return items, return payment method, or authentication—fields preserved by the actual action but not constrained by the reference. The return-payment field is also absent from the legacy DSL's variable readers, so its rejection cannot be required as evidence of learning under that vocabulary. Gemma's 19 exact retail contracts all reject the 440 reference-invalid faults; three additionally preserve authentication. The reference and learned contracts are unchanged by this audit.

There are also omissions within the specified effects. All seven exact Qwen2.5-32B Deployment contracts permit failure to clear the resource allocation, missing 440/2,464 faults each; these wrong successors are themselves reachable states. Six of sixteen exact 14B Deployment contracts miss allocation clearing or region preservation. One Gemma Shopping contract omits wallet/card-limit preservation, and the sole exact 14B retail contract misses exchange-output constraints. In contrast, Gemma's exact Deployment and Calendar contracts reject every tested fault.

This is synthetic fault injection, not a distribution of real execution errors. Only 24, 1,696, 516, and 12 faulty successors are reachable in Shopping, Deployment, Calendar, and retail, respectively; other mutations need not satisfy cross-field invariants. Every top-level field has a mutation, but joint changes, unobserved nested values, and arbitrary values remain untested. High rejection therefore establishes neither logical completeness nor postcondition equivalence. Zero eligible inputs would be reported as N/A rather than success, and evaluation exceptions would not count as fault rejection; neither occurs in this exact cohort. Reproduction uses `analysis/g1_postcondition_strength/audit.py` and its independent aggregation checker `verify.py`, with source, corpus, and input hashes.

Audit across methods. The same single-field generator was also applied to the explicit manifest of 94 model–domain–method cells (1,880 runs), including inexact contracts: 1,622 parse and 258 retain their original parse-failure status. Postcondition-only evaluation tests all successful target inputs; conditional evaluation tests only those accepted by the candidate precondition. The latter has 308 valid contracts with no eligible inputs, reported as N/A. Respectively 428 and 389 of all 1,880 runs accept every eligible normal output and reject every tested fault; failures and N/A cases do not count as passes. These heterogeneous totals are diagnostic counts, not method rankings. The original 153 exact Active scores reproduce exactly. This manifest's retail Sampled is the legacy partitioned arm; later true fixed-pool runs are absent. Per-cell counts and input hashes are preserved in `analysis/g1_common_audit_v2/`.

T.2 Versioned retail effects and bounded multi-field audit

We rerun the W1 fixture under `w1_semantic_fields_v1`, exposing the return-payment field in observations and the DSL and adding the three missing frame conditions to the reference and its effect predicate. All four methods receive the same vocabulary and prompt policy. The 1,294-state closure and fourteen successful target transitions remain the evaluation support; W2 is outside this follow-up. Earlier results and their semantics are preserved separately.

Before model scoring, we generate 890 single-field and 25,616 two-field faulty successors from the normal outputs. The finite value set includes observed field values, boolean flips, numeric neighbors, payment identifiers, and bounded item-tuple insertion, deletion, duplication, or replacement. All fifteen fields are covered; 22,320 faults edit tuple contents, a subset of the 26,506 total. The strengthened reference and reference DSL accept all fourteen normal outputs and reject all faults,

whereas the legacy reference DSL permits 518. This checks the supplied semantics, not LLM improvement. Three-field changes, arbitrary values, and unbounded nesting remain outside the audit. The corpus and reference DSL are not supplied to the model.

Table 28: Versioned W1 follow-up, exact/20. Every exact contract also accepts all fourteen normal outputs and rejects all 26,506 fixed faults. These are new runs, not replacements for Table 3.

Model	Direct	Fixed-pool	Active	No-Coverage
Gemma-4-31B	0/20	3/20	19/20	17/20
Qwen3.8-27B	0/20	0/20	4/20	4/20

All 160 runs parse and complete without transport or runner failures. They use $B = 48$, $Q = 4$, completion/run token caps 2,048/16,000, and reasoning off; initial messages match across methods within model, but initial outputs are not forcibly shared. Final closure scores were independently reproduced for all runs. The primary endpoint requires exact recovery plus acceptance of all normal outputs and rejection of all fixed faults. All 47 exact contracts pass it. In the two-model Active–Fixed-pool family fixed before these runs, discordant pairs are 16:0 for Gemma (raw $p = 0.00003052$, Holm $p = 0.00006104$) and 4:0 for Qwen ($p = 0.125$). The secondary exact-only family gives identical results on the same runs, not an independent replication.

Postconditions alone pass the audit in all 160 runs, including inexact contracts. The method difference therefore concerns applicability recovery within this supplied effect language. All 80 Gemma contracts and 72/80 Qwen contracts use the dedicated snapshot predicate; the other eight Qwen contracts also pass. Conditional audit inputs are empty in two Qwen runs and remain N/A. These findings do not establish independent effect discovery, arbitrary-output equivalence, or a causal improvement over the old protocol. Reproduction uses `analysis/g1_followup_20260909/model_audit_v2.json` and its independent `verification_finished.json`; the first derived audit is invalid because its null-spec cache merged distinct contracts, and is retained only as a superseded record.