DELEGATION UNDER TEST: SPECIFICATION-BASED CONFORMANCE TESTING ACROSS THE LLM–FRAMEWORK BOUNDARY

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models operate within software systems whose frameworks maintain state, execute actions, and sometimes compensate for model errors. System success does not establish which requirements the model fulfills; aggregate scores can also conceal lost behavior after an inference change. We introduce *Delegation Under Test* (DUT), a specification-based method that turns fixed system requirements and responsibility allocations into component obligations and matched tests. Designers choose the contract and case support; executable specifications derive required actions, timing, and startup sets. Stage-specific scoring retains attributable violations and uses aligned evidence for downstream comparisons and regression judgments. We study state exposure and task startup assignment on a structurally selected suite. Passing explicit-state tests can coexist with phase and prerequisite failures when history is present. A correct business-level WAIT can violate model-assigned startup, and correct, aligned startup can precede different downstream timing. A crossed control changes the interpretation of a negative full-interface state-supply contrast; even nonnegative contrasts lose previously passing obligations. Repeated format comparisons exchange passing identities at unchanged totals, and a device change with weights and partitioning fixed also changes conformance. The construction and findings distinguish local correctness, assigned-role fulfillment, and preservation of behavior an application relies on.

1 INTRODUCTION

Language models increasingly serve as software components with behavioral requirements. Workflows with prerequisites and phases require timely actions, phase-appropriate rules, and initiation of assigned tasks. Given sufficient information and an appropriate action interface, fulfillment can be assessed at the component boundary. Surrounding software may supply computations, execute actions, or compensate for errors, so system success does not establish component fulfillment. In a recorded shipping example, both required checks have completed, yet the model waits for an unrelated report; with supplied state, it ships on time (Section 4.2). An ordinary timing test detects the violation. The matched execution also identifies an interface where the same obligation passes. Responsibility allocation specifies the framework's promised support and the model's obligations; an interface realizes that allocation through instructions, observations, and action protocols. Multiple interfaces can realize the same allocation. A fixed system requirement can impose different component obligations across allocations. We ask: *Which specified behavioral requirements are fulfilled under different allocations of responsibility between the language model and its surrounding framework?*

Building on executable contracts and behavioral testing, *Delegation Under Test* links allocation-specific obligations to matched interfaces and executable checks. From a declared system requirement, it constructs tests for those obligations and establishes the evidence each contract judgment requires. The result connects per-allocation fulfillment, differences on comparable identities, and preservation across implementation changes. DUT tests state exposure and task initialization (Figure 1). State exposure supplies ($+R$) or withholds ($-R$) authoritative current state recoverable from the same lossless history; task initialization assigns startup to framework or model.

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.

Applications depend on behavior produced by a deployed implementation, so changing a model or its inference configuration raises a separate question: are previously fulfilled component requirements preserved? DUT retains test identities under each allocation and compares them across interface and implementation changes.

We make three contributions: an executable construction linking fixed system requirements and allocation-specific obligations to matched tests and contract judgments through judgment-specific evidence rules; controlled findings separating local rule conformance from fulfillment of assigned execution obligations; and identity-linked evidence that implementation changes can exchange fulfilled obligations even when aggregate scores or model weights remain unchanged.

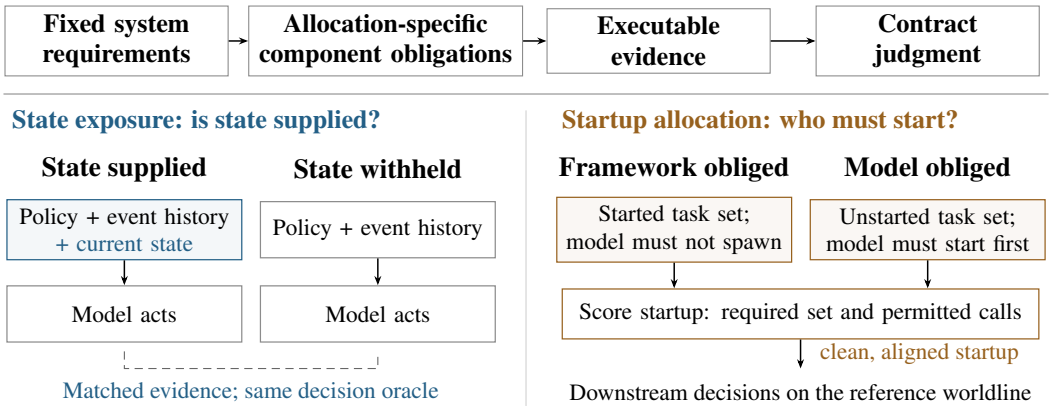


Figure 1: From requirements to judgments. State exposure changes promised support, not the action obligation. Startup allocation assigns the actor; interfaces implement it through initial task status and permitted calls. Startup checks differ; aligned downstream states share a decision oracle.

2 Related Work

Specification-based conformance and executable oracles. Specification- and model-based testing derive cases and expected behavior from explicit models (Stocks & Carrington, 1996; Utting et al., 2012), and adequacy, trace-conformance and oracle research connect them to behavioral obligations, quiescence and executable expectations (Goodenough & Gerhart, 1975; Zhu et al., 1997; Tretmans, 1996; Barr et al., 2015; Bhardwaj, 2026); regression testing retains selected passing behavior across changes (Rothermel & Harrold, 1997; Yoo & Harman, 2012). DUT builds on these primitives.

Sequential decisions and information state. Information-state and sufficient-information work ask what information can change or be omitted while preserving decisions (Åström, 1965; Kaelbling et al., 1998; Puterman, 1994; Li et al., 2006; Mahajan & Tatikonda, 2015; Tavafoghi et al., 2022). DUT instead holds an executable policy fixed and tests whether the model produces its required action from the evidence available under each interface, connecting to long-context and working-memory studies that distinguish information availability from effective use (Liu et al., 2024; Hu et al., 2025). A replay-sufficient history and its explicit current-state summary contain the necessary facts, but impose different demands on their use.

Construct validity and criterion-referenced interpretation. Construct-validity theory connects observations to the interpretations and uses they support (Cronbach & Meehl, 1955; Messick, 1995; Kane, 2013; Borsboom et al., 2004); language-model evaluation research examines corresponding risks in benchmark design and assessment (Bowman & Dahl, 2021; Reuel et al., 2024; Bean et al., 2025; Kostić et al., 2026; Antoine, 2026). Criterion-referenced measurement assesses specified requirements rather than relative standing (Glaser, 1963; Popham & Husek, 1969; Hambleton & Novick, 1973). This distinction explains why an elementary test can remain valuable after it ceases to distinguish models on a ranking (Akhtar et al., 2026; Nadgir et al., 2026): an application may still depend on that behavior. DUT uses absolute conformance and paired differences for different judgments, rather than treating a relative improvement as evidence that the required behavior has been achieved. Appendix H develops these connections.

Behavioral language-model tests, agents, and harnesses. Metamorphic and behavioral language-model tests supply controlled variants and local contrasts (Chen et al., 1998; Ribeiro et al., 2020; Cho et al., 2025); agent task graphs and policy-governed environments place requirements in execution (Shen et al., 2024; Lu et al., 2025; Yao et al., 2025). These lines supply test forms and interaction settings. Harness-Bench compares execution harnesses (Yao et al., 2026), whereas AgentSpec supports controlled scaffold composition through component interfaces (Chen et al., 2026). ContractBench checks temporal and integrity contracts, showing regression-prone compliance distinct from general tool competence (Wang et al., 2026); PM-Bench compares memory scaffolds using per-step evidence (Liu & Gabriel, 2026). Trajectory and tool diagnostics further localize failures beyond final success (Mazaheri & Mazaheri, 2026; Anand & Chattaraj, 2026; Luo et al., 2026). DUT connects these checks to a fixed system requirement and the obligation assigned to each actor. This establishes what a paired pass and failure mean: dependence on promised support, failure to perform an assigned action, or a downstream difference after comparable startup. Detecting an error and establishing one of these allocation judgments are distinct steps. Appendix H traces further testing and agent-system lineages.

3 METHODOLOGY

3.1 CONSTRUCTION AND ALLOCATION

The designer selects the case structures and obligations to cover, and supplies an executable policy, responsibility allocation, interface, case support, observation horizon and actor-specific checks. Replay and oracles determine expected states, required actions and earliest execution times; initially applicable task references determine the startup set. Construction validates proposed states, histories and declared matches. Matching depends on the judgment being sought. Allocation pairs retain the system requirement, inference configuration, action semantics and reference dynamics. State exposure retains the required action while changing what the framework supplies. Startup allocation changes the model's required actions and its tool prefix; comparing downstream timing therefore requires clean startup and reference-aligned states, not identical preceding interactions.

State exposure pairs the same lossless history and action oracle, supplying or withholding its deterministically recoverable authoritative current state. Both conditions contain the facts needed for action: history plus state tests conformance with both present, while history alone tests action from sufficient history. The state-only anchor tests the policy against explicit current facts without history and provides a separate local reference.

Counterfactual construction asks which fact changes should alter the action within each interface. The designer selects a fact and two legal values at two successive policy decision points, yielding four separate decision inputs with replayable history witnesses. The registered example in Figure 2(a) uses preparation and execution. Each history enters its selected phase, starts the fraud review and an auxiliary report, records rejection or approval, then requests a decision while the report remains pending. Replaying these histories under the unchanged policy yields (NEXT, NEXT) in preparation, advancing the phase regardless of review result, and (WAIT, ACT) in execution, shipping only on approval. Varying the fact constructs the unit; supplying or withholding current state pairs its four obligations across interfaces. Appendix A.3 gives the exact policy and prompts.

The four sensitivity transitions organize structural coverage: a fact stays irrelevant (00), becomes relevant (01), becomes irrelevant (10), or stays relevant (11). Different structures can share a signature: entering a phase can make a result relevant, while completing one AND prerequisite can make the other decisive. AND/OR joins distinguish joint from sufficient prerequisites; result guards distinguish completion from an acceptable outcome; phase guards govern applicability; background progress tests invariance. Appendix B.1 specifies the two structures selected per transition and their renderings. Legal states require replay witnesses, and every prefix of a sequential variant must obey the transition rules.

Tool-mediated exposure uses the same policy obligations through an action interface. Waiting advances the reference event sequence; acting submits a business action. Tools, canonical startup and the reference worldline remain fixed. The original interfaces provide the same initial state facts but label them "current authoritative state" and "initial snapshot; subsequent changes through events only." Subsequent tool observations append current state only in the supplied condition. Runtime and reference state must agree at the prefixes used for comparison. Appendix B.2 specifies tool schemas, prompt wording and scheduler behavior.

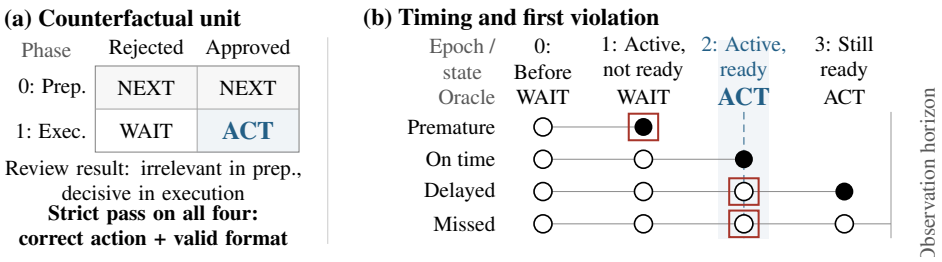


Figure 2: (a) Registered 01 unit: four replayed inputs yield the same four oracle actions under each interface. (b) A separate timing example on one reference trajectory: open circles mark WAIT, filled circles ACT, squares the first violation, and shading the first epoch requiring ACT.

Task initialization changes the startup actor, not the required active-task set. The designer-declared parallel-launch contract supplements business policy: the oracle collects all tasks referenced in the initially applicable clause, including both OR branches. This models eager dispatch: both alternatives activate so either can satisfy the prerequisite; business policy separately determines when to act. The framework launches this set before invocation and forbids model startup calls, or assigns the model to launch it before waiting or acting. Both supply authoritative state but differ in initial task status.

3.2 Scoring and Attribution

Evidence binds the evaluated checkpoint, template and backend. Tool-mediated tests check elementary tool calls before interpreting task-level choices; attributable action violations, including omissions during a task, remain behavioral outcomes. Recorded runtime and reference states determine admission: unexplained mismatches receive measurement NA. Textual semantic scoring requires uniquely resolved actions; a correct choice in a non-compliant format still fails the contract. The replay rules in Appendix C.1 use the actual recorded decision at each reached prefix.

First-deviation scoring follows the order of the assigned obligations; initialization requires precisely the required tasks to start (exact-set conformance), without malformed, unknown, duplicate or forbidden calls (clean conformance). Consider a system that must ship once inventory confirmation passes and either shipping approval or compliance review completes: its parallel-launch contract requires all three referenced tasks to start before waiting or acting. The shipping requirement stays fixed: at the initial decision, framework startup leaves all three tasks pending and forbids model restarts; model-assigned startup leaves them unstarted and requires the model to initiate them. Business prerequisites are unmet in both, so WAIT conforms only in the former; in the latter it violates startup. This establishes a whole-contract contrast: the startup violation remains an attributable failure, but its trace cannot support downstream timing comparison because the startup precondition was never met. Evidence admission is thus judgment-specific. Appendix C.3 gives the recorded construction and contrasts with simpler readouts.

At a comparable decision stage, a selected-epoch strict unit requires all four answers to satisfy their action and output obligations; any attributable violation establishes failure without resolving every answer. State-exposure tests judge these actions directly, not just their counterfactual relation: two WAIT answers in Figure 2(a)'s preparation row preserve equality but violate both NEXT obligations. Sequential scoring compares the first accepted non-WAIT action and its epoch with the oracle's earliest required action (Figure 2(b)). An episode includes assigned startup and ends at the first accepted business action or the observation horizon, with post-terminal submissions undergoing only protocol checks; if no action becomes due, success requires the full horizon without a non-WAIT action. Complete admitted traces without violations conform; incomplete or undecidable traces retain their status. Appendix A.5 specifies wrong-action and no-action outcomes; Appendix C.4 records attempted, attributable and admitted counts.

3.3 Contract Judgments and Regression

Absolute conformance asks whether a configuration fulfills its mandatory obligations. A paired condition contrast asks how the same identities behave under changed support or startup allocation, using the judgments admitted at the relevant stage. Its signed total gains meaning from the paired

outcomes: zero can record shared success, shared failure, or offsetting repairs and regressions. The original state-exposure pair changes both initial framing and continuing state supply. For selected units, both conditions judge the same four registered atoms through determined conjunctions; an empty shared support leaves the paired contrast undefined (Appendix C.1).

Regression testing holds the allocation, application controller and obligation checks fixed while changing the inference implementation, including numerical representation, backend or reasoning mode. Retaining test identities reveals which failures are repaired, which prior passes are preserved, and which prior passes are lost. A declared acceptance contract names both mandatory cases and protected baseline passes. It requires the new implementation to fulfill the mandatory obligations and retain protected behavior, so a repair on one identity cannot compensate for a loss on another. This yields an obligation-level adoption judgment even when aggregate scores are equal. Repetitions retain these identities and add new response realizations, rather than new obligations. Acceptance across a declared set of rounds requires the specified conditions in every round. Invocation conditions, protected identities and missing-evidence rules are fixed before those responses, and no score-selected replacement is allowed. Appendices F.1 and E provide the invocation protocol and identity-linked witnesses.

4 EXPERIMENTS

4.1 POLICY CONFORMANCE AND STATE EXPOSURE

The three input conditions separate explicit rule application from action with sufficient history. The exposure contrast pairs the two history-bearing conditions on byte-matched histories, as reported in Table 1. Appendices C.4 and E give admission counts and source identities.

Table 1: Single-round text conformance, including action, timing and output obligations. Anchor is state-only. With state and Without state both contain the same history; only the former supplies authoritative current state. Cells give passing identities over the registered support; Δ is With state minus Without state in percentage points.[2]

	Selected epoch (72 units)				Sequential (81 trajectories)		
System / mode	Anchor	With state	Without state	Δ	With state	Without state	Δ
OpenRouter API							
Claude Sonnet 5 (off)	70/72	71/72	43/72	38.9	81/81	57/81	29.6
DeepSeek V4 Flash 0731 (off)	72/72	70/72	31/72	54.2	55/81	19/81	44.4
DeepSeek V4 Pro 0813 (off)	72/72	67/72	55/72	16.7	64/81	51/81	16.0
Gemini 3.8 Flash (low)	72/72	72/72	72/72	0.0	81/81	81/81	0.0
GLM 5.3 Flash (low)	72/72	72/72	68/72	5.6	75/81	53/81	27.2
GPT-5.6 Luna (off)	72/72	72/72	63/72	12.5	81/81	62/81	23.5
GPT-5.6 Terra (off)	72/72	72/72	63/72	12.5	81/81	79/81	2.5
Kimi K3 (off)	72/72	72/72	66/72	8.3	81/81	81/81	0.0
MiniMax M3 (off)	66/72	58/72	38/72	27.8	73/81	69/81	4.9
Mistral Medium 3.5 (off)	72/72	72/72	61/72	15.3	81/81	79/81	2.5
Seed 2.1 Turbo (off)	72/72	72/72	62/72	13.9	81/81	78/81	3.7
Tencent HY3 Release (off)	72/72	72/72	69/72	4.2	81/81	70/81	13.6
Tencent HY4 Preview (off)	72/72	72/72	64/72	11.1	81/81	75/81	7.4
Local llama.cpp							
Qwen3.5-9B Q8_0 (off)	72/72	52/72	41/72	15.3	76/81	52/81	29.6

Passing an explicit-rule test does not establish fulfillment when the same facts appear amid history. DeepSeek V4 Flash 0731 passes every state-only unit in Figure 3, yet violates a conjunction in both history conditions: action requires checks K and N to finish, but only N has finished. State-only correctly waits; both history responses act, including the one showing K still pending. A separate OR identity passes with history alone but waits despite an already satisfied branch when state is supplied.

[2]Remote models use OpenRouter; off and low denote the registered reasoning settings, with response-token evidence in Appendix D. The local Qwen configuration also disables reasoning and reports R1; Appendix F.7 gives identity retention across its three rounds.

Thus the net support gain coexists with a retained failure and a newly lost pass. Appendix D.2 gives the matched inputs and responses.

Full conformance is attainable: Gemini 3.8 Flash with low reasoning passes all selected and sequential conditions in one serving profile (Appendix D). Output compliance also affects sequential scores: Mistral's two history-only failures return a history-event label outside the allowed actions. Such output-protocol (OP) violations do not establish semantic correctness or a subsequent timing outcome (Appendices D and D.3). Stable totals need not imply reliable fulfillment: in three reasoning-off Qwen3.5 rounds, 52/72 history-plus-state units pass each round but only 51/72 identities pass throughout (Appendices F.6 and F.7).

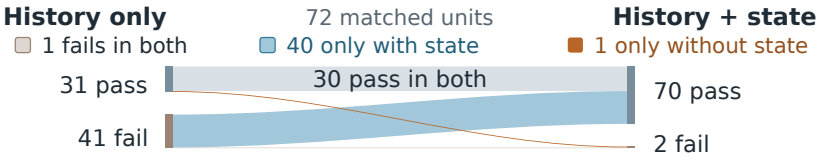


Figure 3: DeepSeek V4 Flash 0731: 72 selected-epoch strict units (eight latent cases, nine renderings each), all passing state-only anchors. Ribbon widths count matched units with the same outcomes across history interfaces.

4.2 Tool-mediated responsibility tests

State exposure, framing and retained behavior. With canonical startup and the same 81 reference trajectories, the supplied-minus-withheld contrast is positive for Qwen3.5 and negative for GLM-4.7-Flash[3] in the historical record and all three fresh-server rounds (Table 2). Matched reference trajectories support identity-level contrasts even when model-induced interaction histories diverge; Appendix C.3 gives their recurrence.

Table 2: State-exposure pass pairs $(+R, -R)$, each out of 81. Left: original interfaces, historical H and fresh rounds R1–R3. Right: separate factorial rounds; Full retains the original interface pair, other rows fix the named heading.

Original-interface recurrence

Model	H	R1	R2	R3
Qwen3.5-9B Q8_0	$(77, 60)$	$(76, 59)$	$(77, 61)$	$(77, 60)$
GLM-4.7-Flash	$(69, 76)$	$(70, 73)$	$(69, 75)$	$(71, 74)$

GLM-4.7-Flash: factorial study

Contrast	R1	R2	R3
Full	$(70, 75)$	$(72, 74)$	$(67, 73)$
Authoritative	$(70, 69)$	$(72, 72)$	$(67, 64)$
Snapshot	$(73, 67)$	$(78, 70)$	$(74, 63)$

In the positive-contrast shipping case, fraud review and inventory confirmation complete by epoch 2. The supplied trace ships then despite a pending auxiliary report; the history-only trace waits for that irrelevant report until epoch 3. In the negative-contrast parsing case, either schema validation or bibliography indexing suffices. Indexing completes at epoch 1: the history-only trace parses then, whereas the supplied-state trace waits for the other branch until epoch 2. Both timing contrasts recur in all three fresh rounds; Appendix C.3 gives the traces and repeated comparisons.

The original pair changes both initial framing and continuing state. A three-round factorial control crosses these factors, retaining both original interfaces and matching initial prompts within each heading to isolate continuing supply. GLM's negative full-interface contrast becomes nonnegative under either fixed heading (Table 2). Every fixed-heading GLM contrast still loses passing identities; even the equal-total authoritative-heading round exchanges four repairs and four regressions. Appendix F.4 gives those identities and the corresponding Qwen controls.

The larger-model controls retain the authoritative heading and disable reasoning for a single-round comparison.[4] Among 81 matched identities, one repair accompanies three regressions for DeepSeek V4 Pro and two for MiniMax M3, and MiniMax's delayed OR actions act on time only in the events-only arms. Appendix F.5 links these single-round repairs and regressions across all three conditions to their matched identities.

[3] Qwen3.5-9B: pre-existing Q8_0; GLM-4.7-Flash: MXFP4_MOE-no-MTP.

[4] DeepSeek V4 Pro 0813: Baidu FP8; MiniMax M3: CoreWeave FP4.

Startup fulfillment across allocations. Correct business-level WAIT need not fulfill assigned startup. On the phase-plus-nested extension case, Qwen3.5 BF16 omits startup in all three rounds under model-assigned startup, while the framework-started counterpart and both retained model-startup controls pass throughout. Under the three-task contract in Section 3.2, a response states: "All relevant tasks are NOT_STARTED. Therefore, rule 2's IF condition is unsatisfied, and the policy returns WAIT," then invokes the waiting tool before starting any task, so that trace establishes an end-to-end failure but cannot admit a downstream timing comparison.

With task names, startup set and reference actions fixed, the matched controls in Appendix F.3 vary phase context and Boolean form. BF16's business-presentation omission persists without phase context, whereas both flat-conjunction model-startup cells pass: phase composition is unnecessary for this omission. Removing phase context also produces forbidden framework-startup restarts in business and software presentations throughout three rounds (Figure 4). Under Q6_K, the neutral-symbolic combined case omits startup in all three rounds while all three simpler matches pass in two.

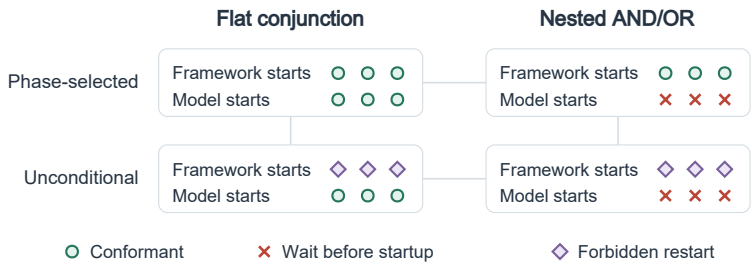


Figure 4: Startup across matched policy forms: Qwen3.5 BF16, business operations, three rounds (one mark each). Startup set and reference actions stay fixed; the vertical change removes phase selection and its inactive clause. Startup passes also conform downstream. Full grid: Figure 5.

On the original suite, BF16 completes model startup whereas Q6_K repeatedly waits before phase/OR startup. In the extension, Q6_K passes every new model-startup case but loses its protected nested-guard control in one round: expanded coverage and preservation of an earlier pass are distinct results. Table 20 gives the allocation-by-format extension grid; Appendices F.1 and D.5 give the repetition plan, protected baselines and historical scores.

The separation also recurs in a larger model with a different rationale: on the neutral-symbolic phase-selected flat conjunction, MiniMax M3 reads the current phase as completed initialization and then waits without starting tasks, passing the other two rounds and framework startup; DeepSeek V4 Pro shows no omission (Appendix F.5).

A separate downstream comparison uses BF16's original conjunctive control: both allocations start correctly and reach the same reference-aligned epoch-0 state, with both gates pending and oracle WAIT. Across all ten original and three extension rounds in Appendix F.2, framework-started episodes execute prematurely at epoch 0, whereas model-started episodes wait correctly until epoch 2.

Responsibility transfer and checker repair. A deterministic fault-injection study executes the unchanged τ-bench return tool on structured receipts and database fixtures (Table 3). The delegated controller deliberately omits its assigned identity and confirmation checks. Yes-only transfers this duty to a wrapper that checks ownership and a positive receipt; intent-bound keeps the same allocation and adds order, item-multiset and refund-destination matching. A four-answer unit reuses the counterfactual constructor: exact confirmation stays irrelevant before delivery but decides WAIT versus RETURN afterward.

Table 3: Constructed retail controls: all three configurations face the same 320 whole-contract input identities, not four-answer units. The tool checks delivery, refund eligibility and item multiplicity.

Configuration	Assigned checker	Implemented check	Passes
Delegated	Controller	None (deliberate omission)	16/320
Yes-only	Wrapper	Ownership + yes	288/320
Intent-bound	Wrapper	Ownership + yes + argument match	320/320

The defect grid also varies those other prerequisites. Moving from yes-only to intent-bound repairs 32 pre-dispatch violations without regressions; in 28, a tool prerequisite already prevents mutation. For example, yes-only forwards a return of item I1 using confirmation for I2, but the tool rejects the undelivered order; intent-bound blocks dispatch. Both leave the database unchanged, yet only the latter fulfills the confirmation duty. Appendix C.3 gives the fixtures and independent upstream and mutation assertions.

4.3 Behavioral regression across inference implementations

The Qwen3.8 comparison crosses Q4_K_M, Q6_K and Q8_0 with two layouts (I/II), with three fresh-process runs per cell. Within each layout, inputs, policy, oracle, invocation settings and scoring stay fixed across formats; all local-anchor, history-plus-state and supplied-state sequential controls conform in every round. Complete hardware and invocation settings are in Appendix G.

Table 4: Identity-linked inference contrasts, counted separately by evidence unit.

Factor	Change	Unit	Pass before → after	Repair	Regress
Format	Qwen3.8: Q6_K → Q8_0	History-only strict unit	$61/72 \rightarrow 61/72$	1	1
Device	Qwen3.8 Q4_K_M: I → T	History-only strict unit	$62/72 \rightarrow 63/72$	1	0
	Qwen3.8 Q4_K_M: I → T	History-only traj.	$74/81 \rightarrow 73/81$	0	1
Format	Qwen3.5: BF16 → Q6_K	Tool $+R$ trajectory	$77/81 \rightarrow 80/81$	3	0
	Qwen3.5: BF16 → Q6_K	Clean model startup	$12/12 \rightarrow 9/12$	0	3
Reasoning	Flash: default → off	Complete response	$10/12 \rightarrow 10/12$	2	2

In both layouts and every round, changing Q6_K to Q8_0 preserves the strict-unit total but swaps passes on two near renderings of one phase guard (Table 4). Both require phase advancement during preparation: unauthorized deployment becomes a correct advance in software infrastructure, while a correct advance becomes unauthorized parsing in data research. Equal totals therefore do not establish interchangeability. A separate business-operations history-only counterfactual pair advances during preparation when a later-phase review is approved but waits when it is rejected, in every format, layout and round. Both inputs require NEXT, and supplied-state controls advance for both; Appendix G.2 gives the paired responses.

On the same Qwen3.8 identities, I-to-II retains format and weights while changing devices and partitioning; the control T in Appendix G.4 replaces only I's tail device, with weights, engine and partitioning fixed. In Q4_K_M's far data-research rendering, approval during preparation elicits unauthorized parsing under I but required phase advancement under II and T; Q6_K and Q8_0 responses remain unchanged across I/II. I-to-T also regresses a conjunctive requirement: shipping requires inventory confirmation and fraud review, but only inventory and an unrelated auxiliary report complete within the observation horizon. I waits; T ships while fraud review remains pending. Since shipping never becomes due, the violation is spurious rather than premature.

The single-realization Qwen3.5 comparison uses BF16 and its derived Q8_0 and Q6_K formats, which retain local-anchor conformance and the history-only selected pass set (Table 17). BF16-to-Q6_K repairs supplied-state tool timing while regressing model-assigned startup (Table 16). The ten BF16-to-Q8_0 history-only tool regressions in Appendix D.6 share one pattern: a completed OR branch authorizes action, yet execution waits for the other.

The single-realization reasoning diagnostic holds DeepSeek V4 Flash's release, provider, prompts, temperature and output cap fixed; default receipts report positive reasoning-token use and disabled receipts zero. On a history-only preparation input, default reasoning advances the phase whereas disabled reasoning selects a business action before it is authorized. Across the diagnostic in Appendix F.8, disabling reasoning replaces literal-name violations with semantic failures while the complete-response total stays unchanged.

5 Discussion

Component capability and conformance. The findings distinguish applying a rule to explicit facts from fulfilling its obligations under an execution interface. State-only passage does not guarantee con-

formance when history is present, even alongside authoritative state. The startup witness establishes a different separation within one execution: the model correctly identifies unmet business prerequisites yet waits before initiating its assigned tasks. DUT makes these distinctions testable by tying each outcome to its obligation and support conditions. Improvement also requires interpretation at that level: a net-positive state-exposure contrast can coexist with losses on previously passing identities. The repeated negative full-interface contrast, which changes both initial wording and continuing state supply, becomes nonnegative under either fixed heading (Table 2); Appendix F.4 details the controls. Framework support therefore cannot be characterized by net benefit alone: absolute conformance identifies which obligations are fulfilled, while identity-level contrasts identify which are gained or lost. Framework support is therefore part of the capability claim. The resulting capability claim is relational: it concerns which obligation the component fulfills under a declared support condition, not a context-free aggregate score. That obligation identity is also the unit carried into regression and acceptance.

Responsibility allocation and framework compensation. Delegation changes the demands on a component, but does not ensure fulfillment of its remaining obligations. Model-assigned startup can be omitted, while framework-assigned startup can be followed by forbidden restarts (Figure 4). Correct startup can also precede different downstream timing despite aligned task states and a common oracle; Appendix F.2 establishes the comparison. Alignment permits a judgment, not a presumption of conformance. Repairing the component under its current allocation and adopting a supported allocation are distinct engineering responses: success under the latter does not establish fulfillment under the former, and remaining obligations still require verification. The retail adaptation likewise leaves pre-dispatch violations visible even when a downstream tool prevents database mutation; Appendix C.3 details the masked repairs. Stage-specific judgments retain that upstream failure while requiring appropriate evidence for each downstream judgment. These results connect function placement to application requirements (Saltzer et al., 1984): surrounding software can secure an outcome without establishing that the component fulfilled its assigned role.

Implementation identity and engineering adoption. The repeated format comparisons make behavioral compatibility more specific than aggregate quality. Equal totals conceal a repaired failure and a new violation of a previously fulfilled requirement (Table 4). An application relying on the latter behavior cannot treat those implementations as interchangeable merely because their scores agree. The timing repairs and startup regressions in a separate format comparison concern different assigned requirements, not interchangeable units of improvement (Table 4). Compatibility therefore depends on which responsibilities the application requires that implementation to fulfill. Retaining allocation-specific test identities carries the original fulfillment judgment into regression testing: the question is whether the candidate preserves the behavioral commitments on which the application depends. This concerns numerical representation (Li et al., 2024), reasoning configuration and execution settings, not just the checkpoint. The reasoning diagnostic exchanges failure identities and types under matched prompts, while the repeated device comparison changes behavior with weights and partitioning fixed; their respective evidence is detailed in Appendices F.8 and G.4. Repetition establishes whether particular repairs and regressions recur. Adoption remains tied to intended use: protected behaviors matter because the application relies on them, while mandatory requirements can include obligations the baseline has never fulfilled. Behavioral preservation thus complements numerical-quality and performance checks by testing whether the changed implementation retains application-required behavior.

Methodological reuse and transfer. Reuse follows obligations and judgment stages, rather than interface changes alone: unchanged action requirements can retain their oracle, while transferred duties need allocation-specific checks. The retail adaptation reuses the condition language and counterfactual constructor while supplying return-specific predicates and pre-dispatch checks; Appendix C.3 distinguishes these shared and adapted elements. Reusing the construction does not settle coverage or preservation. Original controls can continue to pass while new compositions fail; conversely, success on new compositions can coexist with the loss of a retained control, as documented in Appendix F.2. The same checks therefore expose genuinely new failures without making passage on the extension a guarantee of compatibility with retained obligations. Repetition can strengthen evidence about tested obligations without addressing an untested interaction; coverage and recurrence therefore support different parts of an acceptance judgment. What carries across domains is the requirement-to-judgment relationship: domain contracts supply the predicates and duties, while the construction preserves the links needed to interpret fulfillment, dependence and behavioral change.

Reproducibility statement

The Appendix describes the executable grammar, test construction, responsibility allocations, scoring and admission rules, and experimental configurations. The accompanying code provides the experiment infrastructure, including executable oracles, runners, and scoring procedures. Representative inputs and execution witnesses in the Appendix illustrate how observations support the reported judgments.

AI use statement

Generative AI tools assisted with implementation review and debugging, literature search organization and citation-candidate triage, methodological and experimental feedback, interpretation audits, and drafting and editing. AI-generated code, interpretations, and editorial suggestions are treated as proposals for checking against primary sources, executable specifications, and raw records. Gold labels come from executable oracles rather than generative judgments; response readings and inclusion decisions have been separately verified. The authors take responsibility for the final text, claims, code, and artifacts.

References

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel J. Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When AI benchmarks plateau: A systematic study of benchmark saturation, 2026. URL https://arxiv.org/abs/2602.16763.

Atul Anand and Sourav Chattaraj. Diagnosing tool-selection reasoning in LLM agents with canary tools, 2026. URL https://arxiv.org/abs/2608.04719.

Elie Antoine. Evaluation and assessment as complementary frameworks. In *Proceedings of the 1st Symposium on Natural Language Generation Evaluations*, pp. 16–23, 2026. URL https://aclanthology.org/2026.retroeval-main.3/.

Karl Johan Åström. Optimal control of markov processes with incomplete state information I. *Journal of Mathematical Analysis and Applications*, 10(1):174–205, 1965. doi: 10.1016/0022-247X(65)90154-X. URL https://lup.lub.lu.se/record/8867084.

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015. doi: 10.1109/TSE.2014.2372785. URL https://discovery.ucl.ac.uk/id/eprint/1471263/.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, Chris Schmitz, Karolina Korgul, Hunar Batra, Oishi Deb, Emma Beharry, Cornelius Emde, Thomas Foster, Anna Gausen, María Grandury, Simeng Han, Valentin Hofmann, Lujain Ibrahim, Hazel Kim, Hannah Rose Kirk, Fangru Lin, Gabrielle Kaili-May Liu, Lennart Luettgau, Jabez Magomere, Jonathan Rystrøm, Anna Sotnikova, Yushi Yang, Yilun Zhao, Adel Bibi, Antoine Bosselut, Ronald Clark, Arman Cohan, Jakob Foerster, Yarin Gal, Scott A. Hale, Inioluwa Deborah Raji, Christopher Summerfield, Philip H. S. Torr, Cozmin Ududec, Luc Rocher, and Adam Mahdi. Measuring what matters: Construct validity in large language model benchmarks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0590. URL https://papers.neurips.cc/paper_files/paper/2025/hash/1967e0fc3aa6cbbace562f5cb8e3954e-Abstract-Datasets_and_Benchmarks_Track.html.

Varun Pratap Bhardwaj. Agent behavioral contracts: Formal specification and runtime enforcement for reliable autonomous AI agents, 2026. URL https://arxiv.org/abs/2602.22302.

Denny Borsboom, Gideon J. Mellenbergh, and Jaap van Heerden. The concept of validity. *Psychological Review*, 111(4):1061–1071, 2004. doi: 10.1037/0033-295X.111.4.1061. URL https://dare.uva.nl/id/45c23710-9842-4ec9-a7ac-eee24bce2a20.

Samuel R. Bowman and George E. Dahl. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4843–4855, 2021. doi: 10.18653/v1/2021.naacl-main.385. URL https://aclanthology.org/2021.naacl-main.385/.

Jixuan Chen, Jianzhi Shen, Haoqiang Kang, Zhi Hong, Qingyi Jiang, Soham Bose, Yiming Zhang, Leon Leng, Amit Vyas, Lingjun Mao, Siru Ouyang, Kun Zhou, and Lianhui Qin. AgentSpec: Understanding embodied agent scaffolds through controlled composition, 2026. URL https://arxiv.org/abs/2606.14674.

Tsong Yueh Chen, Shing Chi Cheung, and Siu Ming Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Hong Kong University of Science and Technology, 1998. URL https://www.cse.ust.hk/~scc/publ/CS98-01-metamorphictesting.pdf.

Steven Cho, Stefano Ruberto, and Valerio Terragni. LLMORPH: Automated metamorphic testing of large language models. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering, Tool Demonstrations Track*, 2025. URL https://conf.researchr.org/details/ase-2025/ase-2025-tool-demonstration-track/34/LLMorph-Automated-Metamorphic-Testing-of-Large-Language-Models.

Lee J. Cronbach and Paul E. Meehl. Construct validity in psychological tests. *Psychological Bulletin*, 52(4):281–302, 1955. doi: 10.1037/h0040957. URL https://conservancy.umn.edu/server/api/core/bitstreams/1531d762-af5a-4515-95f3-90c01ac994be/content.

Avia Efrat, Or Honovich, and Omer Levy. LMentry: A language model benchmark of elementary language tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10476–10501, 2023. doi: 10.18653/v1/2023.findings-acl.666. URL https://aclanthology.org/2023.findings-acl.666/.

Romain Froger, Pierre Andrews, Matteo Bettini, Amar Budhiraja, Ricardo Silveira Cabral, Virginie Do, Emilien Garreau, Jean-Baptiste Gaya, Hugo Laurençon, Maxime Lecanu, Kunal Malkan, Dheeraj Mekala, Pierre Ménard, Gerard Moreno-Torres Bertran, Ulyana Piterbarg, Mikhail Plekhanov, Mathieu Rita, Andrey Rusakov, Vladislav Vorotilov, Mengjue Wang, Ian Yu, Amine Benhalloum, Grégoire Mialon, and Thomas Scialom. Gaia2: Benchmarking LLM agents on dynamic and asynchronous environments. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/c26a67e0470774df98c12480ec5d2d7b-Abstract-Conference.html.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkula, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, 2020. doi: 10.18653/v1/2020.findings-emnlp.117. URL https://aclanthology.org/2020.findings-emnlp.117/.

Robert Glaser. Instructional technology and the measurement of learning outcomes: Some questions. *American Psychologist*, 18(8):519–521, 1963. doi: 10.1037/h0049294. URL https://doi.org/10.1037/h0049294.

Gonzalo Gonzalez-Pumariega, Leong Su Yean, Neha Sunkara, and Sanjiban Choudhury. Robotouille: An asynchronous planning benchmark for LLM agents. In *International Conference on Learning Representations*, 2025. URL

https://proceedings.iclr.cc/paper_files/paper/2025/hash/d033407a1a4b7a75cd5f9e4575ad9fb5-Abstract-Conference.html.

John B. Goodenough and Susan L. Gerhart. Toward a theory of test data selection. *IEEE Transactions on Software Engineering*, SE-1(2):156–173, 1975. doi: 10.1109/TSE.1975.6312836. URL https://doi.org/10.1109/TSE.1975.6312836.

Ronald K. Hambleton and Melvin R. Novick. Toward an integration of theory and method for criterion-referenced tests. *Journal of Educational Measurement*, 10(3):159–170, 1973. doi: 10.1111/j.1745-3984.1973.tb00793.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1745-3984.1973.tb00793.x.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025. doi: 10.18653/v1/2025.acl-long.1575. URL https://aclanthology.org/2025.acl-long.1575/.

Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011. doi: 10.1109/TSE.2010.62. URL http://crest.cs.ucl.ac.uk/fileadmin/crest/sebasepaper/JiaH10.pdf.

Haitao Jiang, Lin Ge, Hengrui Cai, and Rui Song. PABU: Progress-aware belief update for efficient LLM agents, 2026. URL https://arxiv.org/abs/2602.09138.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL https://www.cassandra.org/arc/papers/aij98.pdf.

Michael T. Kane. Validating the interpretations and uses of test scores. *Journal of Educational Measurement*, 50(1):1–73, 2013. doi: 10.1111/jedm.12000. URL https://www.ets.org/research/policy_research_reports/publications/article/2013/jpdc.html.

Bogdan Kostić, Conor Fallon, Julian Risch, and Alexander Löser. Same meaning, different scores: Lexical and syntactic sensitivity in LLM evaluation. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference*, pp. 4626–4637, 2026. doi: 10.63317/297j5gohq2ip. URL https://aclanthology.org/2026.lrec-1.363/.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, 2006. URL https://thomasjwalsh.net/pub/aima06Towards.pdf.

Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Evaluating quantized large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28480–28524, 2024. URL https://proceedings.mlr.press/v235/li24bb.html.

Genglin Liu and Saadia Gabriel. PM-Bench: Evaluating prospective memory in LLM agents, 2026. URL https://arxiv.org/abs/2607.12385.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act?, 2026. URL https://arxiv.org/abs/2606.28733.

Aditya Mahajan and Sekhar Tatikonda. An algorithmic approach to identify irrelevant information in sequential teams. *Automatica*, 61:178–191, 2015. doi: 10.1016/j.automatica.2015.08.002. URL https://www.ece.mcgill.ca/~amahaj1/projects/graphical-models/preprint/irrelevant.pdf.

Parsa Mazaheri and Kasra Mazaheri. AgentAtlas: Beyond outcome leaderboards for LLM agents, 2026. URL https://arxiv.org/abs/2605.20530.

Samuel Messick. Validity of psychological assessment: Validation of inferences from persons' responses and performances as scientific inquiry into score meaning. *American Psychologist*, 50(9):741–749, 1995. doi: 10.1037/0003-066X.50.9.741. URL https://www.ets.org/research/policy_research_reports/publications/article/1995/buwh.html.

Nitya Nadgir, Sayash Kapoor, Kangheng Liu, Peter Kirgis, Matilda Orona, Stephan Rabanser, Tilman Bayer, Abhishek Shetty, Yue Ling, Derrick Chan-Sew, Rumi Nakagawa, Saiteja Utpala, Zachary S. Siegel, and Arvind Narayanan. Life after benchmark saturation: A case study of CORE-Bench, 2026. URL https://arxiv.org/abs/2606.26158. Accepted at an ICML 2026 workshop.

Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017. doi: 10.1145/3132747.3132785. URL https://www.cs.columbia.edu/~junfeng/papers/deepxplore-sosp17.pdf.

W. James Popham and T. R. Husek. Implications of criterion-referenced measurement. *Journal of Educational Measurement*, 6(1):1–9, 1969. doi: 10.1111/j.1745-3984.1969.tb00654.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1745-3984.1969.tb00654.x.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994. doi: 10.1002/9780470316887. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9780470316887.

Junjia Qi, Zichuan Fu, Jingtong Gao, Wenlin Zhang, Hanyu Yan, Xian Wu, and Xiangyu Zhao. LLM-as-Code: Agentic programming for agent harness, 2026. URL https://arxiv.org/abs/2606.15874. Accepted at the KDD 2026 Workshop on Agentic Software Engineering.

Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. BetterBench: Assessing AI benchmarks, uncovering issues, and establishing best practices. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0685. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/26889e8359e7ef8a7f5d77457364ca55-Abstract-Datasets_and_Benchmarks_Track.html.

Marco Tulio Ribeiro and Scott M. Lundberg. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 3253–3267, 2022. doi: 10.18653/v1/2022.acl-long.230. URL https://aclanthology.org/2022.acl-long.230/.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, 2020. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

François Roewer-Després, Jinyue Feng, Zining Zhu, and Frank Rudzicz. ACCORD: Closing the commonsense measurability gap. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3799–3829, 2025. doi: 10.18653/v1/2025.naacl-long.193. URL https://aclanthology.org/2025.naacl-long.193/.

Gregg Rothermel and Mary Jean Harrold. A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology*, 6(2):173–210, 1997. doi: 10.1145/248233.248262. URL https://doi.org/10.1145/248233.248262.

Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, 1984. doi: 10.1145/357401.357402. URL https://www.cs.cmu.edu/~15712/papers/saltzer84.pdf.

Sergio Segura, Gordon Fraser, Ana B. Sánchez, and Antonio Ruiz-Cortés. A survey on metamorphic testing. *IEEE Transactions on Software Engineering*, 42(9):805–824, 2016. doi: 10.1109/TSE.2016.2532875. URL https://eprints.whiterose.ac.uk/id/eprint/110335/.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0148. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/085185ea97db31ae6dcac7497616fd3e-Abstract-Datasets_and_Benchmarks_Track.html.

Phil Stocks and David Carrington. A framework for specification-based testing. *IEEE Transactions on Software Engineering*, 22(11):777–793, 1996. doi: 10.1109/32.553698. URL https://doi.org/10.1109/32.553698.

Hamidreza Tavafoghi, Yi Ouyang, and Demosthenis Teneketzis. A unified approach to dynamic decision problems with asymmetric information: Nonstrategic agents. *IEEE Transactions on Automatic Control*, 67(3):1105–1119, 2022. doi: 10.1109/TAC.2021.3060835. URL https://doi.org/10.1109/TAC.2021.3060835.

Jan Tretmans. Test generation with inputs, outputs and repetitive quiescence. *Software—Concepts and Tools*, 17(3):103–120, 1996. URL https://research.utwente.nl/en/publications/test-generation-with-inputs-outputs-and-repetitive-quiescence.

Mark Utting, Alexander Pretschner, and Bruno Legeard. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability*, 22(5):297–312, 2012. doi: 10.1002/stvr.456. URL https://eprints.qut.edu.au/57853/.

Jicheng Wang, Yifeng He, Zili Wang, Hanwen Xing, Arkaprava De, and Hao Chen. ContractBench: Can LLM agents preserve observation contracts?, 2026. URL https://arxiv.org/abs/2605.17281.

Wenyi Wu, Sibo Zhu, Kun Zhou, Aayush Salvi, Zixuan Song, and Biwei Huang. StructAgent: Harness long-horizon digital agents with unified causal structure, 2026. URL https://arxiv.org/abs/2607.11388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. HarnessBench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

Shin Yoo and Mark Harman. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012. doi: 10.1002/stvr.430. URL https://coinse.github.io/publications/pdfs/Yoo2010fk.pdf.

Raoyuan Zhao, Abdullatif Köksal, Yihong Liu, Leonie Weissweiler, Anna Korhonen, and Hinrich Schütze. SynthEval: Hybrid behavioral testing of NLP models with synthetic CheckLists. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7017–7034, 2024. doi: 10.18653/v1/2024.findings-emnlp.412. URL https://aclanthology.org/2024.findings-emnlp.412/.

Zenghui Zhou, Man Li, Xiaoke Fang, Xinyi Zhou, Weibin Lin, and Zheng Zheng. LGMT: Logic-grounded metamorphic testing for evaluating the reasoning reliability of LLMs. *Knowledge-Based Systems*, 348:116324, 2026. doi: 10.1016/j.knosys.2026.116324. URL https://arxiv.org/abs/2605.23965.

Hong Zhu, Patrick A. V. Hall, and John H. R. May. Software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4):366–427, 1997. doi: 10.1145/267580.267590. URL https://www.cs.odu.edu/~zeil/cs795SR/Papers/zhu_compSurv_testAdequacy.pdf.

A Executable construction and scoring

A.1 Instrument and matched interventions

The instrument supports two uses of the same executable system specification. Allocation comparisons vary the assigned support or action with inference implementation fixed. Regression comparisons retain the allocation and test identities while changing the inference implementation; their protected baseline and acceptance rules are specified in Appendix F.1. The designer specifies the contract, responsibility allocation, interface rendering, case support, observation horizon, and stage checks before responses. Executable replay and oracles then compute states, required actions and times, and startup sets. The construction below defines the allocation comparison whose tests are retained. Preservation applies to the system specification, not to identical model commands. For allocation α, the model-facing obligation is the projection of that specification onto the actions assigned to the model. Framework initialization and model initialization therefore have different startup checks but the same required active-task set. Downstream comparison requires both runtime states to equal the reference materialization; only then are downstream outcomes comparable under the common decision oracle. The allocation-specific stage checks are fixed before responses.

Let Σ be an executable system specification, E its upstream evidence, ρ a named boundary responsibility, $m_\rho \in \{\text{exposure}, \text{realization}\}$ its intervention mode, $\eta \in \{\text{selected}, \text{sequential}, \text{tool}\}$ its execution regime, and $\mathcal{C}_\rho$ its two registered condition labels. Define

$$T_{\rho,\eta} = (\Sigma, E, m_\rho, \eta, g_\rho, \{\mathcal{I}_\rho^\alpha, \Gamma^\alpha\}_{\alpha\in\mathcal{C}_\rho}, \mathcal{O}_\Sigma, \mathcal{V}^{attr}, \mathcal{V}^{sem}).$$

The ordered construction is specification $\rightarrow$ object or obligation $\rightarrow$ matched interface conditions $\rightarrow$ admission $\rightarrow$ conformance. Here $\mathcal{I}_\rho^\alpha$ denotes the complete registered interface realization, including instructions, observation rendering and action protocol; E denotes underlying task evidence, not a literal prompt string. In exposure mode, $\mathcal{C}_\rho = \{+, -\}$: $g_\rho(E)$ is authoritatively computed in both conditions. At the information-content level, $\mathcal{I}_\rho^+$ supplies $E \oplus g_\rho(E)$, and $\mathcal{I}_\rho^-$ supplies E. The estimand compares these registered interface realizations, with no assertion about internal realization of g_ρ.

Selected-epoch exposure pairs retain byte-matched histories; condition-specific headings describe the available information. Tool-mediated pairs share initial state facts and a reference worldline, while initial headings and subsequent observation rendering differ as registered in Appendix B.2. Their interaction histories may diverge after behavioral differences. Both contrasts therefore compare complete registered interfaces, including framing and state supply.

In realization mode, $\mathcal{C}_\rho = \{F, M\}$ and g_ρ is an observable action obligation: the framework executes it under F, whereas the model must execute it under M, where conformance is scored directly. This mode varies the action's assigned actor; each interface implements that assignment through

instructions, initial status and permitted calls. In both modes, g_ρ and $\mathcal{O}_\Sigma$ are generated before responses are inspected. Allocation pairs fix the inference configuration, system specification, action semantics and reference dynamics. Startup actions and tool prefixes may differ; downstream comparisons require runtime/reference alignment and use the shared decision oracle.

A.2 Specification grammar and oracle

The frozen condition language is the following finite, serializable AST:

$$\begin{aligned} q ::= & \, TRUE \mid EQ(x, v) \mid TASK_COMPLETED(t) \\ & \mid TASK_RESULT_EQ(t, v) \mid NOT(q) \\ & \mid AND(q_1, \ldots, q_n) \mid OR(q_1, \ldots, q_n). \end{aligned}$$

A policy is an ordered collection of clauses

$$c = (when = q_w,\ if = q_i,\ then = a_1,\ else = a_0).$$

For state s, construction requires exactly one applicable clause. The oracle evaluates that clause's `if` AST and returns its `then` or `else` action. The serialized policy retains node types, child order, variables, values, task identifiers, and action space; prompts and replay use the same serialization. Unsupported nodes, duplicate task identifiers, zero or multiple applicable clauses, and references outside a declared catalog fail construction.

The candidate fact e is applied to a copy of the registered state rather than to the policy. A selected-epoch item is admitted only after both intervened states validate, both have replay witnesses when history is present, and the oracle actions reproduce the declared transition cell. Sequential construction additionally replays every epoch and wake edge. This ordering prevents a model response from changing item eligibility or its gold action.

A.3 State exposure and counterfactual units

For the authoritative-state exposure instantiation, an instance contains policy P, state and action spaces $\mathcal{S}, \mathcal{A}$, lossless event history H_t, deterministic replay G, and action oracle a^*. The authoritative state and gold action are $s_t = G(H_t)$ and $a_t^* = a^*(P, s_t)$. The registered interfaces specify how the deterministic rendering $R : \mathcal{S} \to \mathcal{X}$ is supplied or withheld. The model's action is judged against a_t^* in either condition, and the matched contrast concerns these interface realizations, including the tool framing specified above.

Figure 2 illustrates how sensitivity changes with phase while exposure preserves the oracle action. Each counterfactual unit $c = (I_{c,0}, I_{c,1})$ has two valid state interventions at adjacent epochs $u \in \{t, t+1\}$. Let $r_u(c)$ indicate whether $a^*(P, I_{c,0}(s_u)) \neq a^*(P, I_{c,1}(s_u))$. The adjacent signature $(r_t, r_{t+1}) \in \{00, 01, 10, 11\}$ identifies persistent invariance, activation, deactivation, or persistent sensitivity; it is a transition diagnostic, not a global relevance label. For a history-based case, intervention validity additionally requires a replay witness $\exists H_u^{(e)} : G(H_u^{(e)}) = s_u^{(e)}$. Sequential variants require every declared worldline prefix to satisfy the transition semantics.

A fully instantiated selected unit. The business-operations, far-lexical realization of `gating_01` uses this policy: preparation and wrap-up require entering the next phase (`NEXT`); execution requires shipping (`ACT`) if the fraud review is approved, otherwise waiting (`WAIT`). The two selected states have phase preparation at t and execution at $t+1$. In both, the fraud review is COMPLETED and the monthly demand report is PENDING with no result. The candidate fact changes only the review result: e_0 is approved and e_1 rejected. Thus

$$\begin{aligned} \big(a^*(s_t^{(e_0)}), a^*(s_t^{(e_1)})\big) &= (\texttt{NEXT}, \texttt{NEXT}), \\ \big(a^*(s_{t+1}^{(e_0)}), a^*(s_{t+1}^{(e_1)})\big) &= (\texttt{ACT}, \texttt{WAIT}). \end{aligned}$$

The signature is therefore 01. Each recorded history has the following form, with p replaced by the selected phase and r by the candidate result:

t0: enter phase p
t1: start fraud review
t2: start monthly demand report
t3: fraud review completes with result r
t4: request a decision

Replaying this history gives exactly the state above: the report remains pending because no completion event has occurred. The t0–t4 labels index entries within each input; these four selected prompts are separate decisions with separate history witnesses, not successive turns of one execution.

The state-only input supplies the phase and both task states; history-plus-state supplies the history followed by that same state block; history-only supplies the identical history without the block. Their opening sentences identify the available information as current workflow information, history with authoritative current state, or history, respectively. All retain the same policy, three displayed action names, and instruction to return only one listed name. `NEXT`/`ACT`/`WAIT` above are semantic identifiers; the actual answers must use the displayed Chinese names. Each condition passes only if all four answers match their oracle and response obligation. For example, waiting on the rejected review during preparation fails the unit even if the other three answers conform. Appendix G.2 gives the recorded responses, passing controls and four source-row pointers for this unit.

Retail eligibility mapping. These are component requirements within broader agent tasks. In τ-bench's retail policy, with other eligibility conditions fixed, confirmation of the order, items, and refund method authorizes a return only between delivered and return-requested status (Yao et al., 2025). Phase and confirmation predicates preserve this activation/deactivation relation while separating policy use from database lookup and user simulation. This eligibility mapping motivates the construction without adding an empirical domain; the synthetic policy assigns one action per branch, not a unique next conversational move in the original application.

A.4 Construction procedure

The designer selects latent cases against structural coverage targets: the exposure distinctions are specified in Appendix B.1, and the startup fixtures and their extensions in Appendices B.3 and F.2. For each selected case, semantic realization, and condition, construction derives and validates the matched tests before responses are scored:

1. validate the specification and the base state;
2. materialize each registered state; exposure units include both counterfactual values;
3. compute oracle actions and timing; exposure units record the $00/01/10/11$ cell, whereas initialization derives the startup set for its allocation-specific checks;
4. generate and replay any history or complete trajectory witness;
5. render condition-specific observations from the admitted latent object;
6. store canonical specification, state, prompt, oracle, and trajectory hashes;
7. after a response is available, classify the first attributable deviation;
8. aggregate only with the denominator appropriate to that validity layer.

Specification, oracle, and reference-worldline agreement are construction-time checks; runtime alignment is checked on the resulting execution. The worked startup pair in Appendix C.3 shows how these two layers support different contract judgments.

A.5 Decision and timing scores

The admitted response and strict-unit sets are defined in Appendix C.1.

Point conformance requires the oracle-matching action and the declared response format; semantic choice correctness is retained as an auxiliary diagnostic. The primary selected-epoch outcome requires all-four strict-unit conformance:

$$Conf^{\alpha}_{unit} = \frac{1}{|\mathcal{U}^{\alpha}|} \sum_{j \in \mathcal{U}^{\alpha}} Y^{\alpha}_{j,U},$$

where $\mathcal{U}^{\alpha}$ contains units with an established conjunctive verdict: all four conformant responses establish passage, whereas any attributable action or output-protocol violation establishes failure.

An unresolved atom is not assigned an action or error. One correct atom cannot mask its matched counterpart. Relational conformance asks whether the model preserves equality versus inequality between the two counterfactual actions. It requires the corresponding resolved actions and is diagnostic only: a model can preserve a relation while returning the wrong action labels.

For sequential traces, τ^* is the first epoch with a non-WAIT oracle action and $\hat{\tau}$ the first accepted non-WAIT model action. We distinguish premature, on-time, on-time-wrong-action, delayed, missed, correct-no-action, and spurious execute. A no-action success requires completing the registered observation horizon without a non-WAIT obligation or action; the tool controller additionally records the terminal no-event observation. Textual decisions express WAIT through its registered response label. In the tool interface, waiting is realized through the wait tool, not by submitting WAIT as a business action; that rejected submission is a protocol deviation.

For a sequential trace, let

$$d_i^\alpha = \min\{k : \pi_{i,\le k}^\alpha \text{ first violates an admitted obligation}\}.$$

Set $d_i^\alpha = +\infty$ for a complete admitted trace with no violation; an incomplete or undecidable trace retains its admission status. A normally completed response that omits the tool action required at the current epoch establishes an observed execution omission, unlike an interrupted invocation with no attributable response. Such an omission does not require inventing a later business-action obligation: the unfulfilled action can itself be WAIT. The first deviation identifies the earliest violated obligation, not the model's internal error mechanism. Stage admission determines whether subsequent decisions can be scored. In realization mode, the assigned action obligation is scored first, followed by conditional downstream analysis. In exposure mode, the scored obligation is the downstream decision.

B Frozen manifests and matched allocations

B.1 Selected-epoch and sequential support

The selected manifest has the following finite support.

Specification	Registered cases	Transition cells
`canonical_gating_01`	`gating_00`, `gating_01`, `gating_10`	00, 01, 10
`canonical_dual_gate_02`	`persistent_11`	11
`canonical_authz_03`	`authz_11`	11
`canonical_join_and`	`join_and_01`	01
`canonical_join_or`	`join_or_00`, `join_or_10`	00, 10

Each transition contains two cases that separate observable failures on different contract structures. For 00, `gating_00` tests irrelevant background progress across a phase change, whereas `join_or_00` tests it across a relevant task completion. For 01, `gating_01` activates a result predicate on entering a phase; `join_and_01` activates a prerequisite only after its partner completes. For 10, `gating_10` deactivates a predicate on leaving a phase; `join_or_10` makes a pending alternative unnecessary once its partner completes. For 11, `persistent_11` retains a result-dependent ACT/WAIT choice across background progress; `authz_11` retains GRANT/DENY selection across an irrelevant stage change. The latter checks action-label selection beyond acting versus waiting. The selection distinguishes phase gating, connective gating, irrelevance, and persistent label sensitivity. Together they exercise the construction's invariant and action-changing counterfactuals, phase applicability, connective conditions, and action-label correctness. Four domains in near/far lexical forms plus a neutral-symbolic control yield nine renderings per case: 288 responses and 72 strict units per condition.

The sequential manifest selects five cases from four of these specifications: `gating_01`, `gating_10`, `persistent_11`, `join_and_01`, and `join_or_10`. After removing four variants whose complete inputs and worldlines duplicate their base variants, nine trajectory identities

remain. Crossing those identities with the same nine semantic realizations produces 81 rows in each sequential condition. The tool-mediated exposure test uses this same 81-identity support in each allocation.

This selection follows the timing distinctions implemented in `infra_code/experiments/run_responsibility.py`. The base trajectories of `gating_00` and `join_or_00` duplicate those of `gating_01` and `join_or_10`; their selected-epoch counterfactuals remain useful, but do not require another base execution. Authorization has no WAIT-to-action boundary and remains a selected decision test. The retained phase, persistent-result, AND and OR trajectories test immediate obligations, waiting for prerequisites, acting after sufficient completion, and no-action horizons under their registered variants. Arbitrary-depth nesting, recovery policies, and repeated business-action sequences are different structural extensions, not additional coverage supplied by these nine identities.

Regime	Framework-exposed or framework-realized condition	History-only or model-realized condition	Matched unit
Selected epoch	history plus $R(s_u)$	same history, no exposed $R(s_u)$	four-atom unit
Sequential	ongoing $R(s_u)$	history/event observations only	trajectory identity
Tool mediated	ongoing $R(s_u)$	local event results only	admitted tool worldline
Initialization	framework realizes Q^*	language model realizes Q^*	structure-domain identity

The selected and sequential supports are separately matched. No itemwise map is invented between 72 selected units and 81 sequential identities.

B.2 Tool-mediated scheduler and worldline invariants

The tool-mediated state-exposure test begins after deterministic environment setup in both conditions. Every task in the reference startup set has a setup witness and is `PENDING` at scored epoch 0; the language model is forbidden from calling `spawn_task` in the scored phase. Both conditions receive identical epoch-0 state facts, but not byte-identical prompts. The supplied heading is "current authoritative state"; the withheld heading is "initial state (opening snapshot; subsequent changes can only be learned through events)." Subsequent observations append current state only in the supplied condition. Thus initial framing is part of the registered interface contrast; an epoch-0 divergence does not demonstrate an effect of subsequent updates.

The shared tool interface exposes `spawn_task(task_id)`, `wait_for_event()` and `execute_action(action_id)`. A first valid spawn activates a catalog task and returns its status without advancing the decision epoch. Waiting consumes the next registered wake, applies its world events and returns their local result; an event-free wake still advances the epoch. Exhausting the wake sequence returns `no_event` and ends the episode. An accepted non-WAIT action also ends the episode at the current epoch, even if its label or timing is wrong: acceptance by the tool is not oracle conformance. WAIT and unknown action identifiers are rejected as business submissions. The prompts identify the legal task and action IDs, require waiting through the wait tool and acting through the execution tool, and state that ordinary text is not an environment action. Initialization uses these same tools with the allocation-specific startup instructions in Appendix B.3; both allocations receive current state. The shared business-action convention and the preceding startup requirement are quoted together in Appendix F.2.

The scheduler enforces

$$NOT_STARTED \xrightarrow{spawn} PENDING \xrightarrow{event} COMPLETED,$$

and therefore

$$COMPLETED(t) \Longrightarrow \exists u < t : \ spawn(t, u).$$

Each wake records its source and destination epoch even when no state-changing event occurs. At every reached matched prefix, canonical runtime and reference state serializations must hash identically. Tool results are independently replayed from the recorded initial state and calls. Any setup, state-hash, or replay mismatch blocks semantic scoring instead of becoming a timing error.

B.3 TASK-INITIALIZATION ORACLE

Task initialization is a separate responsibility axis. For the unique initially applicable clause c^*, its executable oracle is

$$Q^*_{start} = Refs_{task}(c^*.if) \cap C.$$

`Refs` traverses every syntactic child of `AND`, `OR`, and `NOT`, including task predicates and task-state `EQ` nodes. This all-references rule is part of the initialization contract supplied in both prompts; Boolean action semantics alone do not determine it. The workload models eager parallel dispatch. Launching both OR branches allows either branch to satisfy the prerequisite as its tasks finish, while the business policy releases the action as soon as the full clause permits it. Fixing this activation set makes the assigned startup directly observable and preserves which task events the downstream reference schedule can supply across allocations. Task-set optimization would change that scheduling contract. Clause selection uses initial non-task variables. References outside catalog C fail construction rather than being silently removed by intersection. The frozen manifest contains four registered structures:

Structure	Initially applicable obligation	Distractor excluded by oracle
phase selection	`review_gate`	inactive-clause gates and `aux`
conjunctive join	`gate_a`, `gate_b`	`aux`
disjunctive join	`gate_a`, `gate_b`	`aux`
nested result guard	`risk_check`, `legal_review`, `manager_override`	`aux`

Crossing the four registered structures with three semantic domains produces 12 matched identities per allocation, forming the second responsibility-axis instantiation. In the model-realized allocation, the startup boundary is the first non-spawn command. Its primary score is exact set match at that boundary; the stricter clean score also excludes malformed, unknown, duplicate, or forbidden calls. Under-initialization, over-initialization, wrong-task and duplicate initialization are first-deviation classes. They remain end-to-end failures, while downstream decision conformance becomes NA.

Framework exact-set success is established by deterministic setup; clean conformance additionally tests whether the model respects that allocation, including the prohibition on spawning. Downstream fractions use the successful subset of each allocation. Because initialization determines entry to those subsets, their difference is not an unconditional allocation effect. Paired semantic contrasts use only their intersection; direct-stage scores retain attributable startup failures.

C VALIDITY GATES AND INSTRUMENT CHECKS

C.1 FORMAL ADMISSION AND PAIRED SUPPORTS

Admission has two levels. First, a checkpoint–template–backend realization must pass a configuration gate Γ^α binding its frozen identity, response-evidence contract, and any branch-complete protocol anchors. If $\Gamma^\alpha = 0$, the realization remains in the exclusion ledger and supplies no behavioral denominator. Conditional on $\Gamma^\alpha = 1$, let $\mathcal{V}_i^{attr,\alpha}$ indicate sufficient evidence to attribute an attempted trace, D_i^α passage of any directly scored responsibility stage, C_i^α protocol conformance, and W_i^α reference-worldline agreement. Exposure tests set $D = 1$ because they contain no observed intermediate realization stage; the task-initialization test sets $D = 1$ only after clean initialization. Downstream semantic admission is

$$\mathcal{V}_i^{sem,\alpha} = \mathcal{V}_i^{attr,\alpha} D_i^\alpha C_i^\alpha W_i^\alpha.$$

Response-level policy choice and output-protocol conformance are distinct. The text prompts explicitly require only one listed action name. Conformance therefore requires that exact name, ignoring surrounding transport whitespace. A readable correct choice with additional text or a changed name still violates this obligation ($C = 0$). The original textual executor may use its resolved choice; retaining that choice for semantic replay does not waive the output violation. Genuinely unresolved choices pause semantic replay for a separate reading. Such adjudication requires a registered procedure blind to the oracle, followed by comparison with the oracle. Pending adjudication is neither semantic error nor established undecidability. The downstream criterion above still requires a valid executed trace: a current-epoch reading cannot supply decisions after a stopped run. For tool calls, the declared argument and action protocol governs execution; the text-only response rule is not imposed on tool-interface prose. A verified complete response with no selected action violates the response obligation ($C = 0$); this does not supply a policy choice or timing subtype. Completion and attribution must be established separately from the reading.

The gate order fixes the outcome taxonomy. Infrastructure failure, missing evidence, an undecidable trace, or a framework/reference mismatch whose cause cannot be assigned to the evaluated configuration has $\mathcal{V}^{attr,\alpha} = 0$ and is measurement NA. Given attributable evidence, failure of D, C, or W gives $Y^{\alpha}_{i,E} = 0$ and downstream semantic NA; this includes an explicitly model-caused post-boundary worldline violation, but not an unexplained scheduler/reference disagreement. Otherwise $Y^{\alpha}_{i,S} = \mathcal{O}_{\Sigma}(\pi^{\alpha}_i)$ and $Y^{\alpha}_{i,E} = Y^{\alpha}_{i,S}$. Direct-stage outcomes such as exact-set and clean-initialization conformance are reported on attributable attempts before the downstream gate. Define

$$\mathcal{A}^{\alpha} = \{i : i \text{ was attempted}\}, \qquad \mathcal{E}^{\alpha} = \{i \in \mathcal{A}^{\alpha} : \mathcal{V}^{attr,\alpha}_i = 1\},$$
$$\mathcal{S}^{\alpha} = \{i \in \mathcal{E}^{\alpha} : D^{\alpha}_i C^{\alpha}_i W^{\alpha}_i = 1\}.$$

with paired supports $\mathcal{M}_E = \bigcap_{\alpha \in \mathcal{C}_\rho} \mathcal{E}^{\alpha}$ and $\mathcal{M}_S = \bigcap_{\alpha \in \mathcal{C}_\rho} \mathcal{S}^{\alpha}$. If α_1 and α_0 are the registered treatment and reference labels, the paired layer-L contrast is

$$\Delta^{(L)}_{\rho} = \frac{1}{|\mathcal{M}_L|} \sum_{i \in \mathcal{M}_L} (Y^{\alpha_1}_{i,L} - Y^{\alpha_0}_{i,L}), \qquad L \in \{E, S\}, \quad |\mathcal{M}_L| > 0.$$

For selected-epoch scoring, let $\mathcal{J}$ be the registered strict units and $A(j)$ their four atoms. Passage requires $A(j) \subseteq \mathcal{S}^{\alpha}$ and $Y^{\alpha}_{i,S} = 1$ for every atom. Failure requires at least one $i \in A(j) \cap \mathcal{E}^{\alpha}$ with $Y^{\alpha}_{i,E} = 0$, including an attributable output violation. Let $\mathcal{U}^{\alpha}$ contain units satisfying either criterion, with $Y^{\alpha}_{j,U} = 1$ for passage and 0 for failure, and $\mathcal{M}_U = \bigcap_{\alpha \in \mathcal{C}_\rho} \mathcal{U}^{\alpha}$. Units satisfying neither criterion remain unresolved. Thus unit-level judgment and complete atom-level interpretation have different evidence requirements; an unresolved response is never assigned a guessed action to complete a denominator.

An attributable wrong action establishes strict-unit failure even when another answer in the unit has no uniquely interpretable choice. An explicit output violation also establishes failure without supplying an invented action or timing subtype. `paper/analysis/audit_output_obligations.py` applies this rule uniformly, retaining semantic readings and raw responses separately.

The index i includes a manifest identity and registered response slot. The primary archive instantiates one slot per evaluated-configuration–condition–row, defining the response set for each reported contrast. Every support in the admission chain is reported. Repeated-run acceptance can use the same construction with independently generated slots and a predeclared invocation and decision rule.

C.2 Outcome taxonomy

Observation	End-to-end status	Semantic timing status
correct admitted action on a valid worldline	pass	scored
wrong, premature, delayed, missed, or spurious action on a valid trace	fail	scored

Observation	End-to-end status	Semantic timing status
attributable protocol deviation, initialization-gate failure, or model-caused post-boundary worldline violation	fail	NA
framework/reference mismatch or unattested worldline not caused by the model obligation	NA	NA
infrastructure/API failure or missing response evidence, without an established earlier violation	NA	NA
trace cannot be uniquely paired or replayed	NA	NA

This taxonomy separates construct failure from measurement failure. In particular, an unspawned task cannot complete, an invalid tool-mediated worldline cannot be relabelled as `missed`, and a task-initialization failure cannot be counted again as a downstream timing failure. A later acquisition failure does not erase an already evidenced violation. Such a trace retains its attributable failure, while any timing subtype requiring the unobserved continuation remains unassigned.

C.3 Anchors, mutants, and regression suite

Mutants follow fault-seeding as an adequacy precedent (Jia & Harman, 2011). The checks below distinguish instrument implementation, model protocol compatibility, and invocation recurrence; renderings separately locate presentation sensitivity.

Ten direct policy anchors cover both branches of each of the five canonical specifications. The tool-mediated tests additionally use three minimal anchors to check that the checkpoint-template-backend realization can issue the registered spawn, wait, and execute calls. These elementary checks are separate from conformance in policy-governed episodes and from invocation recurrence. Five executable mutants have registered detection signatures: `AlwaysActive` on 0-cells, `NeverActive` on 1-cells, `PrematureActive` on 01, `StickyActive` on 10, and `DropOnTransition` on persistent 11 cases.

The frozen prospective snapshot passed 227 automated tests. These include oracle uniqueness, counterfactual validity, prompt matching, no-op removal, duplicate-state/duplicate-epoch handling, lifecycle causality, startup-set derivation, state-hash admission, replay, failure-taxonomy, and mutant-signature checks. Tests validate the instrument; their fixtures do not enter response denominators.

The mutant checks operate on action tuples: they transform oracle actions into specified erroneous response patterns and check their strict-pair failure signatures. Replay and worldline checks validate case and execution consistency, while request and response audits connect the invoked interface to the scored record. Repeated model invocations under a declared acceptance rule separately supply evidence for operational qualification.

Executed controller faults. Complementing action-tuple mutants, deterministic controller substitutes issue actual calls through the shared scheduler with the reference contracts and scorer unchanged. Eight correct controls cover four initialization structures under both allocations. Twelve faulty executions substitute AND with OR or OR with AND, select the wrong phase, ignore a result guard, omit required startup, or restart framework-initialized tasks. All eight controls conform; all twelve faults are detected at their expected stages. Corrupting a recorded state witness instead produces measurement failure, separately from controller nonconformance. The executions and checks are implemented in `infra_code/experiments/controller_validation.py` and `infra_code/tests/test_contract_validation.py`. They test executed behavior against fixed expectations rather than deriving faulty answers from gold action tuples, and do not enter model-response denominators.

Independent oracle expectations. Replay checks internal consistency of the declared specification and recorded execution. Complementing replay, two finite-truth tests in `infra_code/tests/test_formal_core.py` specify expected actions directly in ordinary conditional logic, without traversing the production condition AST, calling its clause selector, or reading generated gold to obtain expected values. They check 324 canonical-policy states and 1,408 initialization-policy states, plus five explicit startup-set expectations, including both phase-selection branches. The finite domains cross task statuses `NOT_STARTED`, `PENDING`, `COMPLETED` and `FAILED`; result guards additionally cross absent, accepted and rejected results. Background-task changes, results stored before completion, AND/OR distinctions, phase selection and the nested result guard all enter these checks. The production oracle supplies only the actual value under test. The same file checks generated selected instances and trajectory actions against explicit registered-policy rules, and derives the earliest required action epoch from those independent expected actions. A source-record check against these rules agrees with all 12,096 selected gold labels and 5,503 retained sequential gold labels in the fourteen historical artifacts enumerated by `paper/analysis/text_action_evidence_screen_20260905.json`. The truth check validates the current oracle action. Separate prefix replay verifies that the prompt follows the actual earlier decision, and response interpretation establishes the action expressed in the retained answer. These checks establish agreement on the stated finite truth domains; prompt realization and request provenance are separate audit objects. The initialization fixture audit also has a separately implemented task-reference walker, although that walker's clause selection is shared with the generator. It complements, rather than substitutes for, the explicit finite-truth expectations.

Prompt semantics and scope. An independent text interpreter in `paper/analysis/audit_foundational_requests.py` additionally reads the actual Chinese policy, event history, supplied state and available action names. It derives the required rendered action without calling the production clause evaluator or renderer; when both history and state are supplied, it compares their independently reconstructed facts. All 12,096 selected and 5,503 retained sequential prompts in the fourteen-artifact corpus agree with their recorded action oracles. The 45 selected-policy realizations were also inspected for phase, connective, result and action-name preservation. The interpreter covers this frozen text grammar; tests directly change AND to OR, accepted to rejected results, and supplied phase facts to check its sensitivity. This checks current policy/fact semantics, while the prior-choice prefix audit determines whether each sequential request followed the model's actual earlier choice.

For initialization, the same auditor's `--independent-startup` mode reads the actual WHEN clause, initial phase, task identifiers, explicit all-references startup rule and allocation text. It checks the resulting set against separately listed expectations and verifies that supplied task statuses implement the stated allocation. The check passes for 744 prompts in 31 artifacts: the five original local initialization configurations and the artifacts linked by the two registered repetition ledgers. Their twelve policy presentations (four structures in three renderings) are unchanged across these records and were inspected for clause selection, conjunction/disjunction and nested-result meaning. The text audit rejects removal of the explicit all-OR-branches requirement. These checks test the declared startup obligation, independently of the production task-reference walker.

Method and implementation reuse across responsibilities. Both modes instantiate the same sequence of roles: executable requirement, allocation intervention, admissible evidence, oracle comparison, and retained identity. Exposure takes the policy action as the requirement, assigns provision of the derived observation, and matches decision prefixes before scoring action and time. Initialization takes the declared startup set as its direct requirement, assigns its execution, and scores exact/clean startup before matching downstream states. The first violated obligation and its admission layer determine the reported outcome in both modes. The startup oracle and stage boundary therefore change, while the construction and interpretation roles remain reusable. The following implementation map supplies engineering evidence for that methodological mapping.

The initialization runner, `infra_code/experiments/run_d4_pilot.py`, imports `DeterministicScheduler`, tool names and serialization from the existing tool-mediated substrate, obtains the same tool schemas, and reuses `_run_anchors` and the transport implementations. Its `_make_executor` wraps scheduler calls with the initialization boundary. The classifier in `infra_code/src/async_controller_bench/d4/runtime.py` invokes the shared scheduler's `replay`; initialization exact/clean checks, first-stage attribution, downstream admission

and timing classification are implemented in that classifier rather than delegated to the exposure classifier. The startup oracle and reference-trajectory adapter are in `d4/spec.py`; allocation-specific rendering and matched-fixture checks are in `d4/render.py` and `d4/audit.py`. Reporting uses the common `paper/analysis/build_evidence.py` entry with separate `audit_d3` and `audit_d4` paths. Thus shared data and execution mechanisms are direct code reuse, while scoring and identity validation adapt the construction to the assigned obligation. The selected four-answer counterfactual generator is not reused as an initialization scoring unit.

Independent-source contract adaptation. A bounded adapter executes the original τ-bench retail return tool (Yao et al., 2025), with its policy and implementation pinned at commit `59a200c6d575d595120f1cb70fea53cef0632f6b`. `infra_code/vendor/tau_bench/PROVENANCE.json` binds the source files and license by digest. The adapter supplies structured authentication and confirmation receipts and invokes the unmodified tool against local databases. Its evaluation object is this contract boundary, separate from conversational extraction or a user-simulator task.

The fixed system contract requires authenticated ownership, confirmation of the exact order, item multiset and refund destination, delivered status, a permitted refund method, and valid item multiplicity. Identity and confirmation must be checked before dispatch; the tool checks database prerequisites. Three configurations use the same eager controller substitute: delegated checking leaves the upstream check to that controller, which deliberately omits it; a yes-only wrapper checks ownership and a positive receipt; an intent-bound wrapper also checks that the receipt covers the proposed arguments. Delegated-to-wrapper comparison transfers the pre-dispatch duty; yes-only-to-intent-bound comparison repairs the wrapper's checker with that duty unchanged. Final database behavior and the assigned pre-dispatch obligation are scored separately, retaining an upstream violation even when the tool prevents an unauthorized update.

The finite grid crosses two order statuses, two authentication states, two ownership settings, five confirmation settings, four refund-method settings and two item-multiplicity settings, yielding 320 input identities and 960 executions. The confirmation settings include absent, negative, matching positive, stale-item and stale-refund receipts. Some factor identities induce the same facts when authentication is absent; the count denotes inputs, not independent workflows. Whole-contract passes are 16, 288 and 320 under delegated, yes-only and intent-bound checking, respectively; correct final mutation outcomes number 282, 316 and 320. An independently implemented ordinary assertion baseline checks the same upstream duties, database constraints, exact updates and progress on valid requests, agreeing with DUT at both stages on every execution.

Moving from the yes-only to the intent-bound wrapper repairs 32 pre-dispatch violations, preserves all 288 prior whole-contract passes, and yields no regressions. Four repairs prevent unauthorized database updates; in the other 28 inputs another tool prerequisite had already prevented the update. All 316 previously correct mutation outcomes are retained. Thus mutation-only acceptance would conceal most of the repaired component obligations. The finite acceptance decision rejects the yes-only wrapper and retains the intent-bound wrapper for this structured-input contract.

The main-text masked-repair witness has input identity `pending/True/True/stale_items/original/valid` in `retail_return_traces.jsonl`: the receipt confirms I2, but the proposed return requests I1. Yes-only dispatches and receives "Error: non-delivered order cannot be returned"; intent-bound rejects the confirmation mismatch before dispatch. Both preserve the database.

The adapter reuses the executable condition language and four-answer counterfactual constructor, while supplying return-specific facts, tool execution and pre-dispatch/post-mutation checks in `infra_code/experiments/retail_return.py`. `infra_code/experiments/run_contract_validation.py` rebuilds the controller and retail evidence with an explicit output directory. The current receipt is in `paper/analysis/local_completion_20260925/contracts/`: its summary binds the shared source version, all 960 traces and controller executions. Reproduction does not overwrite the original study records.

Paired allocation witness. In `infra_code/experiments/d3_full_Qwen3.5-9B-Q8_0_llamacpp_20260901_002855.json`, `/results/4` and `/results/13` share identity `gating_01/base/data_research/far`, with supplied and withheld state, respectively. The

reference first-action epoch is 0 in both. The supplied trace contains `execute_action(NEXT)` at epoch 0; the withheld trace contains `wait_for_event` followed by `execute_action(ACT)` at epoch 1. Initial prompts, observations and tool traces are retained in each record. The state facts match, but the initial headings differ as specified in Appendix B.2. This pair establishes an initial-obligation violation, not an effect of later state updates; its later action does not erase that violation. Replaying these records checks that attribution. New invocations test recurrence of the paired behavior, a distinct question from identifying the violation in the recorded realization.

The main-text example instead uses `/results/90` and `/results/99` of the same artifact, identity `join_and_01/base/business_ops/far`. Its policy is "If fraud review and inventory confirmation are complete, ship; otherwise wait." Both initial prompts list `gate_a`, `gate_b`, and `aux` as pending, with the distinct headings above. Both traces first call `wait_for_event` twice, receiving inventory completion and then fraud-review completion. The supplied observation additionally lists both gates completed and the auxiliary report pending. At epoch 2 its response is `execute_action(ACT)`; the withheld response is another wait, followed by `execute_action(ACT)` at epoch 3 after the auxiliary report completes. The reference earliest action epoch is 2, with runtime/reference equality through the shared prefix. This is a record-level contrast after state exposure has begun; initial framing remains part of the treatment. In the three registered Qwen3.5-9B follow-up artifacts listed below, this same base identity again ships at epoch 2 with state and at epoch 3 without it in every round, with the same two-wait versus three-wait tool sequences. These fresh records, not replay of the original pair, establish its observed recurrence.

The opposite-sign GLM-4.7-Flash witness is `join_or_10/base/data_research/far`, at `/results/130` (supplied state) and `/results/139` (history only) in each of the three fresh GLM artifacts indexed by the exposure-run ledger below. The policy permits parsing after either schema validation or bibliography indexing completes. Indexing completes at epoch 1, the required action epoch. The history-only trace executes then; the supplied-state trace waits for schema validation and executes at epoch 2. Both traces are decidable and worldline-valid in all three rounds, with the same paired timing outcomes. As in the Qwen witness, this contrast concerns the registered interfaces, including their initial framing.

Same-record comparison with itemwise checks. Both shipping traces above eventually execute the required action, so final-action correctness alone misses the delay. A timing-aware itemwise test detects it. Matching the records additionally establishes a passing supplied-state control for the same obligation under the registered interfaces.

The BF16 `phase_nested_release` pair in Appendix F.2 makes the construction explicit. The designer supplies a fixed system contract: launch every task referenced in the initially applicable clause before waiting or acting, and ship at the first epoch when inventory confirmation has passed and either shipping approval or compliance review has completed. The designer also selects the two allocations, task catalog, interface instructions, reference schedule, and stage checks. From the active RELEASE clause, the startup oracle derives {`release_gate`, `approval_gate`, `legal_review`}, excluding inactive-phase and background tasks. The action oracle returns WAIT both before startup and with these tasks pending; the reference schedule first authorizes shipping at epoch 3. Projecting the same contract onto each allocation produces the checks in Table 9.

A business-label check alone would credit the correct WAIT decision without checking startup. A startup-set check detects the omission. The dependency between these checks determines what follows: the attributable omission remains an end-to-end violation even though it prevents reference alignment; rejecting the entire trace as measurement NA would erase that violation. Conversely, scoring absent shipping against the canonical schedule would add a timing failure without an admitted downstream starting state. The pair supports a whole-contract contrast between allocations, but not a comparison of their downstream timing. Thus admission is specific to the judgment, rather than a single valid-run filter.

Identity retention supplies a separate link for implementation changes. In the Qwen3.8-27B Q6_K-to-Q8_0 comparison, both configurations pass 61/72 history-only selected units, but one identity is repaired and another regresses in each layout and repetition. Ordinary identity-level regression testing also detects this exchange. Protecting the lost pass rejects preservation despite the unchanged total; this retrospective illustration does not retroactively register an adoption rule. `paper/analysis/route_c_same_evidence.py` reconstructs the exposure contrasts on both complete 81-pair

Table 9: One system contract, allocation-specific obligations, and the judgments licensed by the recorded BF16 startup pair. Key instruction and response excerpts, with source pointers to the full records, are in Appendix F.2.

	Framework initializes	Model initializes
Model obligation	Do not restart the initialized tasks; follow the business policy.	Start exactly the derived set before waiting or acting.
Observed first call	`wait_for_event`, after framework startup.	`wait_for_event`, with no task started.
Startup judgment	Conformant; required set active and no forbidden call.	Violated; no required task started at the first wait.
Whole-contract judgment	Conformant; startup aligns with the reference and shipping occurs at epoch 3.	Violated at startup; downstream timing is NA.

exposure supports and all six corresponding 72-unit format comparisons, retaining the witness pointers and source hashes in `infra_code/experiments/route_c_local_20260919/same_evidence.json`.

Audit execution and artifact entry points. From the repository root, run `infra_code/.venv/bin/python infra_code/tests/test_formal_core.py` for finite-truth tests. Run `paper/analysis/audit_foundational_requests.py` with `--independent-prompts` and an artifact path for the prompt audit, using the Python path setup below; `--independent-startup` selects the startup-contract audit. For identity-level analysis of audited exposure artifacts, run `infra_code/.venv/bin/python infra_code/experiments/compare_d3_runs.py` with `--paired-rounds` followed by the artifact paths for one configuration. The output includes each realization's four paired cells, shared-support denominator, unresolved identities and each condition's identity-level violation counts. Source records and invocation dates remain attached to the observations. On the original Qwen3.5-9B Q8_0 and GLM-4.7-Flash records, the paired cells are $(60, 17, 0, 4)$ and $(69, 0, 7, 5)$, respectively, in the order both pass, supplied-only pass, withheld-only pass and both fail. Supplying an artifact twice is rejected. The run registration, rather than the number of input files, establishes which observations are fresh invocations.

Fresh exposure records and request audit. The registered follow-up retains the two original configurations in Table 2, including the pre-existing Qwen3.5-9B Q8_0 rather than its same-source derived format. Three fresh-server rounds per configuration retain all 81 identities in both allocations, giving 972 episodes. Configuration order alternates within each round; serving parameters and initial prompt bytes match their respective historical baselines. Historical responses remain separate from these new invocations.

`infra_code/experiments/d3_revision_20260913_runs.jsonl` locates all six artifacts, their preflight records, request journals and execution timestamps. Each entry's `artifact` and `request_journal` fields supply the two arguments to `paper/analysis/audit_tool_request_links.py`:

```
infra_code/.venv/bin/python \
  paper/analysis/audit_tool_request_links.py ARTIFACT JOURNAL
```

For each `preflight` artifact, the corresponding journal replaces `_preflight.json` with `_preflight_requests.jsonl`. All twelve artifact/journal checks account for 2,628 logical requests, including protocol anchors before and after each grid. This checks wire messages, tools, invocation settings and response linkage; semantic admission additionally requires the trajectory and observation-horizon rules above.

In the table, H denotes the historical record; 1–3 denote fresh-server rounds.

Configuration / round	Both pass	Supplied only	Withheld only	Both fail	Pairs
Qwen3.5-9B Q8_0 / H	60	17	0	4	81
Qwen3.5-9B Q8_0 / 1	58	18	1	4	81
Qwen3.5-9B Q8_0 / 2	61	16	0	4	81
Qwen3.5-9B Q8_0 / 3	59	18	1	3	81
GLM-4.7-Flash MXFP4_MOE-no-MTP / H	69	0	7	5	81
GLM-4.7-Flash MXFP4_MOE-no-MTP / 1	66	4	7	4	81
GLM-4.7-Flash MXFP4_MOE-no-MTP / 2	66	3	9	3	81
GLM-4.7-Flash MXFP4_MOE-no-MTP / 3	68	3	6	4	81

The signed supplied-minus-withheld contrasts remain positive for Qwen3.5-9B Q8_0 (17/81, 16/81, 17/81) and negative for GLM-4.7-Flash MXFP4_MOE-no-MTP ($-3/81$, $-6/81$, $-3/81$). At the identity level, Qwen3.5-9B Q8_0 violates four supplied-state and 17 history-only requirements in all three rounds; the corresponding counts for GLM-4.7-Flash MXFP4_MOE-no-MTP are six and three. Other violations are intermittent. These counts describe recurrence of nonconformance, not necessarily an identical timing category or tool sequence. Historical paired witnesses identify recorded failures; a witness need not fail in every fresh invocation for that original attribution to remain valid.

Unrealized waiting versus a missed business action. In the first new Qwen3.5-9B Q8_0 round, `d3_full_Qwen3.5-9B-Q8_0_llamacpp_20260913_085827.json`, `/results/77` reaches all three reference states, whose required actions are WAIT, WAIT, WAIT. After two wait calls, the model gives a final text response without a tool call. It correctly states that the rejected gate does not authorize the business action, but does not realize the final required wait. The registered interface requires waiting through `wait_for_event`; ordinary text is not an environment action. This is an observed execution omission, not missing response evidence and not a failure to recognize the rejected result. Its frozen binary nonconformance judgment is retained. The raw runner's `missed` label is explained as `unrealized_wait` in identity-level analysis: there was no required business action to miss. Raw records are unchanged.

C.4 Evidence units and matched supports

The registered selected-epoch manifest contains 288 response atoms, grouped into 72 four-answer units, per configuration and condition. The sequential manifest contains 81 trajectories per condition. Paired contrasts match the same identities under supplied and withheld state; aggregate scores use the validity layer defined in Appendix C.1.

Each remote main-table panel uses one OpenRouter serving profile across its three selected-epoch conditions and two sequential conditions. Sequential histories retain the model's actual preceding decisions. Transport recovery retains successful responses and fills only unavailable slots under the same profile; it does not replace observed behavioral violations. Registrations and HTTP receipts bind the requested model, routed provider and reasoning setting. For every configuration, conformance includes the explicit name-only output instruction as well as the action and timing obligations. The source-bound readout is `paper/analysis/analyze_frontier_panels.py`.

The same-source Qwen3.5-9B BF16, derived Q8_0 and Q6_K records separately support the selected/sequential results in Tables 16 and 17. Their selected counts reproduce; each format's 162 trajectories has no flagged prefix, and all recorded prompts, actions and trajectory scores pass replay. These are distinct from the pre-existing Q8_0 serving-profile follow-up in Table 1. The configuration-specific transport and repetition boundaries accompany Table 17. The ledger below includes fourteen remote configurations and all three Qwen follow-up rounds retained in Appendix F.7; Table 1 displays Qwen's first round. Rounds are not new configurations or independent structures. Complete-choice units require all four atoms to be admitted; an established action or output violation decides a strict unit's failure, as defined in Appendix C.1.

Run-level admission. The table below accounts for attempts, decision-level admission and scoring units within the displayed configurations. Initialization judgments retain attributable startup violations before conditional downstream scoring.

Condition	Unit	Tried	Judged	Denominator
Local anchor	response atom	4,896	4,896	1,224/1,224 strict units
Selected $+R$	response atom	4,896	4,896	1,224/1,224 strict units
Selected $-R$	response atom	4,896	4,896	1,224/1,224 strict units
Sequential $+R$	trajectory	1,377	1,377	1,377 trajectories
Sequential $-R$	trajectory	1,377	1,377	1,377 trajectories
Tool $+R$	tool trajectory	162	162	162 trajectories
Tool $-R$	tool trajectory	162	162	162 trajectories
Init.: framework	startup trace	24	24	24 startup; 23 downstream
Init.: model	startup trace	24	24	24 startup; 15 downstream

"Judged" denotes a determined contract verdict, not necessarily a uniquely selected semantic action. One history-only selected response makes no action selection but violates the response instruction; the other 14,687 selected responses have unique source-bound readings. Tool exposure has 162 checked pairs. Exact-set and clean-startup outcomes use every attempted task-initialization trace; downstream outcomes are defined only after clean startup and a valid worldline. These different startup and downstream supports reflect the stage-specific obligations, with upstream violations retained as end-to-end failures. Tool and initialization counts are checked from the hash-locked records.

D Complete descriptive results

Table 10 distinguishes the questions and controlled changes across the experimental groups. Presentation variants probe the same latent structures; fresh rounds probe recurrence on retained identities. Matched follow-ups additionally separate policy form from task bindings in Appendix F.3, initial framing from ongoing state provision in Appendix F.4, and per-epoch reconstruction from cumulative history in Appendix F.6.

Registered model display names are written without abbreviations throughout the descriptive tables. Table 11 binds each displayed system name to its exact endpoint identifier. Local-system names include checkpoint size and representation format; immutable model-file hashes and backend configurations remain in the machine-readable evidence.

OpenRouter configurations and response evidence. Gemini 3.8 Flash uses the pinned OpenRouter route `google-ai-studio/flex` with reasoning effort `low`; its endpoint requires reasoning. Tencent HY4 Preview uses `tencent/fp8` with reasoning disabled. Seed 2.1 Turbo and Mistral Medium 3.5 use `seed/fp8` and `mistral`, respectively, also with reasoning disabled. Each configuration retains one serving profile across 864 selected responses (72 four-answer units per condition) and 162 sequential trajectories (81 per condition), with temperature zero and a 2,048-token completion limit. The sequential input uses the cumulative, append-only history defined in Appendix F.6, with actual model decisions in each reached prefix. The Gemini panel's 1,260 response receipts include 1,210 with positive reasoning usage and 50 with zero reported reasoning tokens. All terminate normally and satisfy the action, timing and output obligations. All 1,260 HY4 receipts terminate normally and report zero reasoning tokens. Its eight history-only selected-unit failures are wrong actions, with no output-format violation; the corresponding state-only and supplied-state units pass. In sequential execution, supplied state also removes the six observed history-only violations: one missed action, one premature action and four spurious actions on no-action horizons. Both additional reasoning-off panels pass all state-only and supplied-state tests but retain history-only violations. Seed's ten selected-unit failures are wrong actions, and its three sequential failures are premature actions. Mistral's eleven selected-unit failures are wrong actions; its two sequential failures instead return "request decision", a history-event label outside the three allowed actions, when the oracle requires WAIT. They violate the explicit response contract, without establishing a selected action or a later timing outcome. All 1,257 receipts in each panel terminate normally and report zero reasoning tokens. The other complete remote panels use the same selected and sequential support. DeepSeek V4 Flash 0731 uses `deepinfra/fp8`; DeepSeek V4 Pro 0813 uses `baidu/fp8`; Kimi K3 uses `sail-research/fp4`; MiniMax M3 uses `coreweave/fp4`; and Tencent HY3 Release uses `deepinfra/fp4`. These profiles disable reasoning and set temperature zero with a 2,048-token completion cap. Claude Sonnet 5 uses `azure/global`, GPT-5.6 Luna uses `openai`, and GPT-5.6 Terra uses `openai/flex`; they likewise disable reasoning and use a 2,048-token cap, omitting the

Table 10: Experimental relationships. Fixed conditions apply within the named contrast; composition coverage changes specifications, while allocation pairs within each composition keep that specification fixed.

Question and support	Held fixed	Varied	Judgment supported
Local correctness versus interface conformance (Section 4.1); fourteen configurations, with three fresh Qwen rounds and a three-round history-construction control	Policy, matched identities and oracle within each configuration; serving profile within the history control	Selected: state-only, history-plus-state, history-only. Sequential: state exposure; per-epoch or append-only history as specified in Appendices D and F.6	Whole-contract and semantic judgments; whether history-interface failures persist across input constructions
Complete interface versus continuing state (Section 4.2); 81 pairs, local three-round and larger-model single-round studies	Model profile, tools, canonical startup and reference worldline; initial heading within continuing-state contrasts	Original interfaces; crossed headings and ongoing observations; original events-only promise as specified in Appendices F.4 and F.5	Factor-specific contrasts, repairs and regressions; recurrence only within the repeated comparison
Assigned startup and downstream behavior (Section 4.2); original 12 identities per allocation, three/ten rounds; two larger models, three rounds on 58 unique episodes	Policy, catalog, state provision and reference schedule within an allocation pair	Framework/model startup; fresh invocations under each registered profile in Appendices F.1 and F.5	Omitted startup, forbidden restarts, timing after aligned startup, and protected behavior
Startup across policy forms; six new compositions and retained controls, then a four-cell matched design; BF16/Q6_K, three rounds	Startup semantics and executor; matched forms also fix task names, startup set and worldline	Coverage extension in Appendix F.2; crossed phase context and Boolean form in Appendix F.3	New coverage versus retained passes; conditions under which role-specific violations recur
Reasoning configuration (Section 4.3); Flash/Pro, one four-atom unit in three interfaces, one call per mode and prompt	Release, provider, prompts, temperature, output cap and oracle	Default versus disabled reasoning as specified in Appendix F.8	Paired semantic and output-contract changes on a fixed diagnostic support
Low-reasoning conformance; Gemini 3.8 Flash, complete text panel; GLM 5.3 Flash, three diagnostic rounds of 46 prompts	Each model's low-reasoning serving profile, policy, identities, oracle and output obligation	Gemini: three selected interfaces and two sequential conditions; GLM: selected units and retained sequential prefixes	Gemini's full-panel conformance at low reasoning; recurrence of GLM output violations on the diagnostic support
Same-source formats (Section 4.3); Qwen3.5-9B, single recorded realizations	BF16 source, controller, allocation, identities and checks	BF16 and its derived Q8_0/Q6_K	Within-obligation repairs and regressions across decision and startup tests
Repeated formats (Section 4.3); Qwen3.8-27B, three formats, three rounds per layout	Layout, allocation, inputs, invocation settings and checks	Numerical-format realization	Repeated exchanges of passing identities despite equal totals
Fixed-weight execution (Section 4.3); Qwen3.8-27B, three rounds	Format and weights; T also retains I's partitioning	I/II: devices and partitioning; I/T: tail device only, Q4_K_M	Behavioral preservation under execution changes
Retail construction checks (Section 4.2); 320 inputs, three deterministic configurations	Contract, structured receipts, original return tool and database fixtures	Controller-to-wrapper responsibility transfer; yes-only-to-intent-bound checker repair	Stage judgments independently checked by assertions; repairs masked by unchanged terminal outcomes

Table 11: Displayed names and registered identifiers for the selected-epoch and sequential configurations.

Displayed model name	Registered endpoint
Claude Sonnet 5 (off)	`anthropic/claude-sonnet-5`
DeepSeek V4 Flash 0731 (off)	`deepseek/deepseek-v4-flash-0731`
DeepSeek V4 Pro 0813 (off)	`deepseek/deepseek-v4-pro-0813`
Gemini 3.8 Flash (low)	`google/gemini-3.8-flash`
GLM 5.3 Flash (low)	`z-ai/glm-5.3-flash`
GPT-5.6 Luna (off)	`openai/gpt-5.6-luna`
GPT-5.6 Terra (off)	`openai/gpt-5.6-terra`
Kimi K3 (off)	`moonshotai/kimi-k3`
MiniMax M3 (off)	`minimax/minimax-m3`
Mistral Medium 3.5 (off)	`mistralai/mistral-medium-3-5`
Seed 2.1 Turbo (off)	`bytedance-seed/seed-2-1-turbo`
Tencent HY3 Release (off)	`tencent/hy3`
Tencent HY4 Preview (off)	`tencent/hy4-preview`
Qwen3.5-9B Q8_0 (off)	`qwen3.5-9b` (local llama.cpp)

unsupported temperature parameter. All successful response receipts in these reasoning-off panels report zero reasoning tokens. GLM 5.3 Flash uses `inference-net/fp4`, temperature zero, an 8,192-token completion cap and low reasoning: 268 of its 1,218 successful receipts report positive reasoning usage and 950 report zero. Requested low therefore does not imply nonzero reasoning on every answer.

Registrations, prompt support, source snapshots and raw HTTP journals are retained under `infra_code/experiments/frontier_full_20260926/`, `infra_code/experiments/openrouter_full_panel_20260926/`, and `infra_code/experiments/remote_full_deepseek_v4_flash_0731_20260926_r2/`. The source-bound readouts in `paper/analysis/frontier_full_20260926/` are produced by `paper/analysis/analyze_frontier_panels.py`.

D.1 Point-level selected-epoch conformance

Table 12: Point-level text conformance, out of 288 responses per condition. Anchor is state-only. With state and Without state share the same history; only the former supplies authoritative current state. Remote rows use OpenRouter; Qwen rows report local fresh rounds.

Model	Anchor	With state	Without state
Claude Sonnet 5 (off)	285/288	287/288	251/288
DeepSeek V4 Flash 0731 (off)	288/288	286/288	221/288
DeepSeek V4 Pro 0813 (off)	288/288	283/288	271/288
Gemini 3.8 Flash (low)	288/288	288/288	288/288
GLM 5.3 Flash (low)	288/288	288/288	283/288
GPT-5.6 Luna (off)	288/288	288/288	278/288
GPT-5.6 Terra (off)	288/288	288/288	278/288
Kimi K3 (off)	288/288	288/288	281/288
MiniMax M3 (off)	280/288	270/288	239/288
Mistral Medium 3.5 (off)	288/288	288/288	277/288
Seed 2.1 Turbo (off)	288/288	288/288	278/288
Tencent HY3 Release (off)	288/288	288/288	285/288
Tencent HY4 Preview (off)	288/288	288/288	280/288
Qwen3.5-9B Q8_0 (R1–R2)	288/288	261/288	232/288
Qwen3.5-9B Q8_0 (R3)	288/288	260/288	232/288

Qwen R1–R3 are the serving-profile follow-up in Appendix F.7; the sequential breakdowns below use its append-only history arm. The earlier archived selected realization has point conformance counts $287/288$, $261/288$, $232/288$ and selected paired cells $(39, 13, 2, 18)$; it remains a separate record, not an additional follow-up round.

D.2 STRICT-UNIT PAIRED CELLS

Table 13: Paired text conformance on the same history identities. P/P and F/F denote pass/pass and fail/fail; With only and Without only identify passage exclusively in the With state and Without state conditions, respectively.

	Selected exposure ($N = 72$)				Sequential exposure ($N = 81$)			
Model	P/P	With only	Without only	F/F	P/P	With only	Without only	F/F
Claude Sonnet 5 (off)	43	28	0	1	57	24	0	0
DeepSeek V4 Flash 0731 (off)	30	40	1	1	18	37	1	25
DeepSeek V4 Pro 0813 (off)	50	17	5	0	41	23	10	7
Gemini 3.8 Flash (low)	72	0	0	0	81	0	0	0
GLM 5.3 Flash (low)	68	4	0	0	47	28	6	0
GPT-5.6 Luna (off)	63	9	0	0	62	19	0	0
GPT-5.6 Terra (off)	63	9	0	0	79	2	0	0
Kimi K3 (off)	66	6	0	0	81	0	0	0
MiniMax M3 (off)	30	28	8	6	63	10	6	2
Mistral Medium 3.5 (off)	61	11	0	0	79	2	0	0
Seed 2.1 Turbo (off)	62	10	0	0	78	3	0	0
Tencent HY3 Release (off)	69	3	0	0	70	11	0	0
Tencent HY4 Preview (off)	64	8	0	0	75	6	0	0
Qwen3.5-9B Q8_0 (R1)	39	13	2	18	52	24	0	5
Qwen3.5-9B Q8_0 (R2)	39	13	2	18	52	24	1	4
Qwen3.5-9B Q8_0 (R3)	40	12	1	19	54	21	0	6

Three-condition failure localization. Figure 3 shows DeepSeek V4 Flash 0731's 72 selected identities, all of which pass without history. Matching `abc_results` by case, domain, lexical form and atom verifies identical policy text across all three conditions, identical current-state text in the state-only and history-plus-state prompts, and identical history text in the two history conditions. Thirty strict units pass in both history conditions, 40 pass only with supplied state, one passes only with history alone, and one fails in both. Thus history plus state passes 70/72 units and history alone passes 31/72, despite identical state-only conformance.

The shared failure is `join_and_01/neutral_symbolic/neutral/t1_e1`. The rule requires both checks K and N to complete before action Q. The history starts K, N and an auxiliary task R, records completion of N, and requests a decision; K and R remain pending. State-only returns WAIT, while both history conditions return action Q, including the condition explicitly listing K as pending. This is a prerequisite violation, not a failure of the local state-only control. In `join_or_10/business_ops/far/t_e0`, one completed gate authorizes shipping while the other gate and the auxiliary report remain pending. State-only and history-only responses ship; history plus state waits. The latter is a lost passing identity within a net-positive exposure contrast. The source is `infra_code/experiments/remote_full_deepseek_v4_flash_0731_20260926_r2/passes/unreviewed/result.json`; the four answers per identity determine the strict-unit verdicts above.

D.3 SEQUENTIAL CONFORMANCE AND VIOLATION TYPES

Tables 14 and 15 classify each trajectory once per condition. OP denotes a violated output instruction at or before the first wrong semantic choice; a subsequent malformed prefix cannot contribute an additional timing error. Otherwise, the retained action/timing outcome supplies the category. OT and CNA are complete passes; the remaining columns are failures. OP alone does not determine semantic correctness: a response can violate the output instruction either with a uniquely interpretable action or without identifying an allowed action.

For interpretation, a correct label before/after its first required epoch is premature/delayed; a wrong label is a wrong-action violation with its epoch retained. No execution despite a required business action is missed. Execution when no business action is required within the horizon is spurious. Omitting a required wait-tool invocation is instead an unrealized-WAIT violation, even when a legacy record calls it missed: it does not establish a missed business action. Unavailable or undecidable responses retain their evidence status rather than receiving any of these semantic labels.

Table 14: Sequential conformance in the With state condition: history plus authoritative current state, on 81 trajectories per row. Qwen rows report fresh rounds.

	Conformant		First violation					
Model	OT	CNA	Pr	De	Mi	SE	WA	OP
Claude Sonnet 5 (off)	63	18	0	0	0	0	0	0
DeepSeek V4 Flash 0731 (off)	44	11	19	0	0	7	0	0
DeepSeek V4 Pro 0813 (off)	46	18	0	15	2	0	0	0
Gemini 3.8 Flash (low)	63	18	0	0	0	0	0	0
GLM 5.3 Flash (low)	59	16	0	0	0	0	0	6
GPT-5.6 Luna (off)	63	18	0	0	0	0	0	0
GPT-5.6 Terra (off)	63	18	0	0	0	0	0	0
Kimi K3 (off)	63	18	0	0	0	0	0	0
MiniMax M3 (off)	55	18	0	5	3	0	0	0
Mistral Medium 3.5 (off)	63	18	0	0	0	0	0	0
Seed 2.1 Turbo (off)	63	18	0	0	0	0	0	0
Tencent HY3 Release (off)	63	18	0	0	0	0	0	0
Tencent HY4 Preview (off)	63	18	0	0	0	0	0	0
Qwen3.5-9B Q8_0 (R1)	58	18	0	2	3	0	0	0
Qwen3.5-9B Q8_0 (R2)	58	18	0	2	2	0	0	1
Qwen3.5-9B Q8_0 (R3)	57	18	0	2	4	0	0	0

Table 15: Sequential conformance in the Without state condition: history only, on the same 81 trajectory identities per row.

	Conformant		First violation					
Model	OT	CNA	Pr	De	Mi	SE	WA	OP
Claude Sonnet 5 (off)	46	11	0	0	0	0	0	24
DeepSeek V4 Flash 0731 (off)	16	3	47	0	0	15	0	0
DeepSeek V4 Pro 0813 (off)	33	18	3	4	23	0	0	0
Gemini 3.8 Flash (low)	63	18	0	0	0	0	0	0
GLM 5.3 Flash (low)	43	10	0	1	0	0	0	27
GPT-5.6 Luna (off)	47	15	16	0	0	3	0	0
GPT-5.6 Terra (off)	61	18	2	0	0	0	0	0
Kimi K3 (off)	63	18	0	0	0	0	0	0
MiniMax M3 (off)	54	15	4	4	1	3	0	0
Mistral Medium 3.5 (off)	61	18	0	0	0	0	0	2
Seed 2.1 Turbo (off)	60	18	3	0	0	0	0	0
Tencent HY3 Release (off)	52	18	7	0	0	0	0	4
Tencent HY4 Preview (off)	61	14	1	0	1	4	0	0
Qwen3.5-9B Q8_0 (R1)	42	10	6	8	7	8	0	0
Qwen3.5-9B Q8_0 (R2)	43	10	6	7	7	8	0	0
Qwen3.5-9B Q8_0 (R3)	44	10	5	6	7	8	1	0

OT: on-time; CNA: correct no action; Pr: premature; De: delayed; Mi: missed; SE: spurious execution; WA: wrong action; OP: output-protocol violation. Qwen supplied-state R2 includes one complete response selecting no action. It violates the response obligation without an invented action or timing subtype.

D.4 TOOL-MEDIATED ADMISSION

Both reported tool-mediated systems pass all three tool anchors, and all 81 paired trajectories in each allocation satisfy the reference/runtime worldline audit. Main-text summaries appear in Table 2; complete paired cells are retained here, and no non-admitted realization is presented as a behavioral result.

D.5 TASK-INITIALIZATION ALLOCATION-SPECIFIC FIRST DEVIATIONS

Historical clean-conformance scores for framework/model startup are 11/12 and 10/12 for the pre-existing Qwen3.5-9B Q8_0, and 12/12 and 5/12 for GLM-4.7-Flash. Model-assigned exact-set scores equal the respective clean scores. Framework setup itself starts the required set in all 12 episodes; the scores above judge the model's assigned behavior. These records are separate from the state-exposure repetitions and factorial study in Table 2.

Evaluated system	Allocation	Registered first deviation
Qwen3.5-9B Q8_0 (pre-existing GGUF)	Framework	none: 11; unexpected framework spawn: 1
Qwen3.5-9B Q8_0 (pre-existing GGUF)	Model	none: 10; invalid tool call: 2
GLM-4.7-Flash MXFP4_MOE-no-MTP	Framework	none: 12
GLM-4.7-Flash MXFP4_MOE-no-MTP	Model	none: 5; over-initialization: 6; invalid tool call: 1

The two model-allocation invalid calls from Qwen3.5-9B Q8_0 (pre-existing GGUF) cross the startup boundary before the required set has been realized; the stored trace also records the resulting under-initialization and boundary-before-initialization diagnostics. The table uses the registered primary first-deviation field and does not double count those consequences.

D.6 BOUNDED SAME-SOURCE FORMAT SENSITIVITY AUDIT

Table 16: Same-source Qwen3.5-9B repairs and regressions, from single recorded realizations using BF16-derived formats. Rows use different evidence units and must not be summed.

		BF16 → Q8_0		BF16 → Q6_K	
Test condition	Shared support	Repairs	Regressions	Repairs	Regressions
Selected, state exposed	72 strict units	0	3	1	7
Sequential, state supplied	81 trajectories	0	0	0	1
Sequential, state withheld	81 trajectories	0	0	0	2
Tool-mediated, state supplied	81 trajectories	0	0	3	0
Tool-mediated, state withheld	81 trajectories	0	10	10	0
Model initialization, clean startup	12 episodes	0	2	0	3

This bounded audit compares three admitted GGUF realizations of one frozen Qwen3.5-9B source checkpoint. BF16 and Q8_0 form the original pre-result pair; Q6_K was later derived directly from the same frozen BF16 file and registered before any Q6_K model output. The Q6_K extension is therefore prospective relative to itself but is not an independent confirmation of the original comparison. Here Q8_0 denotes the registered Qwen3.5-9B-DELEGATE-Q8_0 BF16-derived GGUF (SHA-256 prefix `5848262d`), distinct from the pre-existing Qwen3.5-9B-Q8_0 GGUF (`78aca7d0`) used in the cross-model tables.

All three use the same server executable, embedded chat template, generation and request settings, prompts, tools, scheduler, oracle, and RX 7900 XT. The older BF16/Q8_0 commands additionally set the logging-only flag `-lv 4`; the Q6_K command omits it, and the evidence audit normalizes only this verbosity difference while requiring every behavioral serving field to match. Hash-locked receipts bind source and model files. Q6_K additionally passes its preregistered preflight, and every displayed format passes its registered tool anchors, identity, replay, response-evidence, worldline, and admission checks. Temperature-zero settings freeze the request contract but do not establish repeated-output stability.

	State exposure		Within-format pair			
Format	$+R$	$-R$	PP	$+R$ only	$-R$ only	FF
BF16	77/81	71/81	71	6	0	4
Q8_0	77/81	61/81	61	16	0	4
Q6_K	80/81	81/81	80	0	1	0

Comparison	Condition	PP	BF16 only	Comparator only	FF
BF16/Q8_0	State exposed	77	0	0	4
BF16/Q8_0	State withheld	61	10	0	10
BF16/Q6_K	State exposed	77	0	3	1
BF16/Q6_K	State withheld	71	0	10	0

BF16 and Q8_0 have identical state-exposed outcome labels on all 81 identities; with state withheld, five BF16 on-time identities become delayed and five become missed under Q8_0. Relative to BF16, Q6_K changes three exposed-state delays to on-time and leaves one delay unchanged; with state withheld, all nine BF16 delays and its one miss become on-time. Prompts and reference τ^* values match exactly across all three formats. Terminal reference-state hashes may differ because that stored endpoint is evaluated where each realization stops; every run nevertheless has runtime/reference equality at every reached epoch, so such endpoint differences are behavioral rather than design drift.

Disjunctive timing regression across presentations. All ten BF16-to-Q8_0 regressions above occur in `join_or_10` with state withheld. The affected presentations are software infrastructure (near and far), data research (near), and support access (near and far), each under `base` and `e1`. At epoch 1, `gate_b` completes and satisfies the OR rule; BF16 acts, while Q8_0 waits. In `base`, `gate_a` later completes, and Q8_0 executes at epoch 2 or 3. In `e1`, `gate_a` remains pending; the auxiliary completion still occurs, but Q8_0 waits through the terminal no-event observation. This produces a delay or miss from the same first-deviation pattern. The paired cells over all nine presentations of each schedule are $(1, 0, 5, 3)$: one retained failure, no repair, five regressions, and three retained passes. All other history-only trajectory outcome labels are unchanged, and every state-supplied disjunctive trajectory conforms in both formats.

These are cross-presentation and cross-schedule witnesses from the recorded format comparison. Their raw locations are `/results/137`, `/results/138`, `/results/140`, `/results/141`, and `/results/142` for `base`, and `/results/155`, `/results/156`, `/results/158`, `/results/159`, and `/results/160` for `e1`, in the BF16 and Q8_0 artifacts named by `paper/analysis/quantization_extension_evidence.json`. The traces retain the first completed gate, the subsequent wait, and the matched runtime/reference states. Fresh-invocation recurrence is established separately for the initialization and deployment witnesses below.

	Framework initializes			Model initializes		
Format	Exact	Clean	Downstream	Exact	Clean	Downstream
BF16	12/12	11/12	10/11	12/12	12/12	12/12
Q8_0	12/12	10/12	10/10	10/12	10/12	10/10
Q6_K	12/12	12/12	11/12	9/12	9/12	9/9

All 24 task-initialization identities have identical prompts, reference τ^* values, and required startup sets across formats. In the model-initialized allocation, BF16 passes all 12 exact/clean startups, Q8_0 passes 10 with one invalid call and one over-initialization, and Q6_K passes 9 with three invalid calls. Each invalid call is the registered first deviation before startup completion; it is not propagated into a downstream timing label. Framework-realized and model-realized results remain separate, and downstream denominators remain conditional on clean initialization.

These tool-mediated results contain one frozen response realization per admitted format of one source checkpoint. Q6_K improves both tool-mediated state-exposure conditions relative to BF16 while reducing direct task-initialization conformance, so the observed profile is not consistently ordered by nominal bit width or by a single notion of model capability. These are transitions between recorded responses, supplying concrete repair/regression candidates. Attribution to format rather than invocation variability requires repeated matched comparisons of the corresponding obligations. The

initialization repetitions and deployment study provide such evidence only for their own registered comparisons.

The cross-responsibility comparison is directly recoverable from the BF16 and Q6_K source pairs listed in `paper/analysis/quantization_extension_evidence.json`. For the timing repair, select `case_id=join_and_01`, `variant=base`, `condition=sequential_with_state`, `domain=software_infra`, and `confusability=near`. The common required epoch is 2; BF16 executes at epoch 3 after the auxiliary completion, whereas Q6_K executes at epoch 2. For the initialization regression, select `case_id=phase_selection`, `domain=software_infra`, and `allocation=model_initialized`. BF16 calls `spawn_task(review_gate)` before waiting; Q6_K first calls `wait_for_event`, which is rejected with `unspawned_task_event`. The raw response logs and tool traces retain both calls and their environment results.

D.7 Foundational conformance under format changes

Table 17: Same-source foundational format comparison from audited recorded responses. Selected columns count all-four strict units (72 each); sequential columns count conformant trajectories (81 each). Formats are directly tied to the frozen Qwen3.5-9B source.

	Selected			Sequential	
Model	Anchor	State exposed	History only	State exposed	History only
Qwen3.5-9B BF16	72	54	41	74	51
Qwen3.5-9B Q8_0	72	51	41	74	51
Qwen3.5-9B Q6_K	72	48	41	73	49

The foundational supplement freezes the same source formats, renderer, policy, and scoring rules before its behavioral calls, using two slots and concurrency two on RX 7900 XT. For each of BF16, Q8_0 and Q6_K, all 864 selected responses replay and the strict-unit counts reproduce. The source-bound prefix audit flags none of the 162 trajectories per format. Sequential replay verifies all 405, 405 and 408 recorded epochs, respectively, including prompts, oracle actions, model choices and trajectory scores; no response is unavailable. Independent prompt interpretation also agrees throughout. The Q8_0 and Q6_K request journals additionally reconcile all 1,269 and 1,272 HTTP attempts, respectively. BF16 retains its original semantic evidence without retrospective full-HTTP attestation. The source hashes match the response and prefix audits in `paper/analysis/text_action_evidence_screen_20260905.json` and `paper/analysis/historical_action_prefix_review_20260905.json`.

Relative to BF16, the history-plus-state selected cells $(00, 01, 10, 11)$ are $(18, 0, 3, 51)$ for Q8_0 and $(17, 1, 7, 47)$ for Q6_K. Q8_0 retains every sequential pass identity. Q6_K's supplied-state sequential cells are $(7, 0, 1, 73)$, and its history-only cells are $(30, 0, 2, 49)$. All formats share the same local-anchor and history-only selected pass identities. These are supported recorded-response transitions under the registered comparison. Each format contributes one response realization; recurrence is established separately for the registered deployment comparisons in Appendix G.

Source records and offline replay. The three raw artifacts are in `infra_code/experiments/`, named `responsibility_full_Qwen3.5-9B-DELEGATE-`*format*`_llamacpp_`*timestamp*`.json`: BF16 uses `20260905_071701`, Q8_0 uses `20260905_080639`, and Q6_K uses `20260905_081711`. Each artifact contains `abc_results` and `d_results`; indices in the witness exports address these arrays directly. The paired exports are `paper/analysis/foundational_bf16_q8_paired_20260905.json` and `paper/analysis/foundational_bf16_q6_paired_20260905.json`. From the repository root, the following commands replay all three artifacts and the two retained HTTP journals without invoking a model:

```
export PYTHONPATH=infra_code/src:infra_code/experiments
for entry in BF16:071701 Q8_0:080639 Q6_K:081711; do
  format=${entry%%:*}
  stamp=${entry##*:}
  artifact=infra_code/experiments/responsibility_full_
```

```
  artifact=${artifact}Qwen3.5-9B-DELEGATE-${format}
  artifact=${artifact}_llamacpp_20260905_${stamp}.json
  set -- "$artifact"
  if [ "$format" != BF16 ]; then
    journal=infra_code/experiments/foundational_quant_
    journal=${journal}${format}_20260905_requests.jsonl
    set -- "$@" --journal "$journal"
  fi
  infra_code/.venv/bin/python \
    paper/analysis/audit_foundational_requests.py "$@"
done
```

For these artifacts, the audit reports legacy-parser replay rather than retrospective certification under a newer output protocol. Their available responses also match the supplied action labels exactly. BF16's absent HTTP journal remains an unestablished transport-evidence field, not a failed behavioral obligation.

E PROVENANCE AND RECONSTRUCTION

Quoted Chinese prompts and responses throughout the paper are rendered in English; the artifacts retain the original wording.

A violation witness retains the specification predicate or startup obligation, triggering state or event prefix, first incorrect action, original request and response, and matched controls. Response-reading summaries report adjudicated, pending, and unavailable answers together with adjudication coverage; a pending reading is not counted as an error or as established undecidability.

The versioned artifacts connect reported outcomes to source records, registered test identities, prompts, and executable oracles. Provenance is recorded by evidence type: OpenRouter profiles and HTTP receipts identify remote API configurations; local derivation receipts bind source and quantized checkpoints; tool traces retain calls, observations, and reference/runtime state witnesses. The archive records which identity and transport fields are available for each run. Evidence compilers replay the source records, verify hashes and matched grids, apply the relevant admission requirements, and derive paper-facing counts from normalized JSON rather than prose. Retained revisions, file digests, serving configurations, and the exclusion ledger support reconstruction of the reported numerators, denominators, paired cells, and admission decisions.

For request-journal-enabled runs, the evidence chain links a manifest row or decision epoch to a logical request identifier, then to numbered HTTP attempts. Each attempt retains the submitted messages and decoding parameters, followed by the raw response body and status or a transport exception. The independent audit reconstructs prompts and gold actions, checks the request-to-result correspondence, validates retry order, reparses responses, and recomputes trajectory outcomes. Hashes bind the retained files; authorization headers are excluded. Historical semantic replay and full HTTP-attempt attestation are separate evidence fields, so richer subsequent instrumentation does not retroactively attest an unavailable historical record.

Bad-case reconstruction retains source identities, original responses, and the first oracle-disagreeing epoch. Deterministic replay verifies the recorded deviation against the specification. Fresh model invocations test whether that deviation recurs and require separately registered response slots. The bad-case export enumerates all qualifying deviations under its declared rule, rather than selecting examples by apparent explanatory value.

A replayable decision violation. The first selected mismatch in source order in the foundational Qwen3.5-9B BF16 export is row `/abc_results/40`: `gating_01`, data-research rendering, far lexical form, atom `t_e0`, with history and state supplied. The policy requires advancing to the next phase during preparation; starting data parsing is authorized during execution when schema validation has passed. The displayed state is still preparation, with validation passed and the background quality report pending. The model answers "start data parsing," whereas the executable oracle returns "enter the next phase." The answer names a valid action but violates its phase guard. Replaying the history and policy reproduces this disagreement. The export retains the full prompt, response, source pointer, and source-file SHA-256 for direct reconstruction.

F SCOPE OF STATISTICAL INTERPRETATION

Empty admitted support gives an undefined rate or contrast, reported as NA. For engineering acceptance, missing evidence on a required test leaves that requirement unresolved; passage on the remaining admitted subset does not discharge it. This separates measurement admission from release acceptance while retaining attributable failures as failures.

All reported system configurations and manifest units are fixed. The nine semantic renderings are nested within a latent unit, and the primary archive contains one recorded response per registered evaluated-configuration–condition–row. Counts and paired cells describe those finite response realizations. Recurrence and repeated-call acceptance are assessed in the separately registered repeated-invocation studies, with invocation conditions, intended use and acceptance rules fixed before responses. The initial three rounds and subsequent seven-round extension retain their separate registrations. Structure, transition, domain and lexical-distance slices localize behavior within the tested support; interface contrasts use the specified renderer R. Conformance is judged from observable actions against the oracle. A violation refutes fulfillment of its registered requirement; extending coverage to new structures requires corresponding tests.

F.1 REGISTERED REPEATED-INVOCATION QUALIFICATION EXTENSION

The single-realization interpretation above describes the original corpus. A separate follow-up is registered before its new invocations in `infra_code/config/d4_qualification_20260911.json`. It retains the original initialization manifest and historical results as baseline evidence. The intended use is acceptance of the specified workflow startup and subsequent controller contract under a declared framework-initialized or model-initialized allocation.

The registration covers Qwen3.5-9B BF16 and its directly derived Q8_0 and Q6_K formats, together with GLM-4.7-Flash MXFP4_MOE-no-MTP. Each configuration receives three fresh-server rounds of all 24 registered episodes, for 288 new episodes. Historical responses define protected obligations but do not count toward these new rounds. Each configuration retains its recorded template, context and slot arrangement, greedy decoding, tool contract, scheduler, and oracle on RX 7900 XT. Protocol anchors remain separate from the episode denominators.

Absolute acceptance is assessed separately for each allocation and for startup and end-to-end conformance. It requires a pass on every registered identity in every new round. Regression acceptance instead checks every explicitly protected baseline pass: the BF16 baseline defines the protected sets for the three Qwen3.5-9B formats, and the historical GLM-4.7-Flash result defines its own protected sets. The registration enumerates these identities before the new responses. Preserving the protected set can coexist with an absolute failure on another mandatory obligation; the two verdicts therefore remain distinct.

The intended use is a mandatory-contract regression check under the recorded serving conditions. All-pass is appropriate to that decision because each listed obligation is required, rather than exchangeable for another successful episode. An application instantiates this rule by mapping required and forbidden behavior to cases exercising its enabling and disabling conditions. Existing passing identities on which callers rely define the protected set before the candidate is observed. A new phase rule or responsibility therefore changes required coverage; preserving an older passing set alone does not test the added duty. The obligation-to-case mapping determines what the acceptance judgment covers.

Repetition addresses a separate source of uncertainty. The invocation plan specifies which operating conditions may vary and fixes the repeat budget before candidate responses; failure consequences guide how extensively those conditions are exercised. Here each fresh server repeats the same 24 identities with the serving configuration fixed, testing whether their judgments recur across new server realizations. The initial three-round study and the prospectively added seven rounds retain their separate registrations. Additional structures extend coverage, as in Appendix F.2; additional restarts exercise the existing support without enlarging it.

A verified attributable violation gives a violated verdict, with any unresolved cells still reported. Complete passing evidence gives a pass; without a verified violation, missing or inadmissible required evidence gives an unresolved verdict. Repairs cannot cancel a protected regression, and failures are not replaced by score-selected reruns. Startup failure identifies the first violated stage and precludes a

downstream timing label. These rules define a finite-contract acceptance check under the registered invocation conditions; the three rounds are repeated executions, not additional independent latent tasks. All 288 registered episodes were collected, and every registered startup and whole-contract judgment is decidable; Table 18 reports all 32 verdicts.

Table 18: Registered finite-check assessment across three fresh-server rounds. Counts give passing episodes out of 36 per allocation (12 identities per round). A denotes absolute acceptance and R preservation of the protected baseline set; P means pass and V violated. Startup uses clean conformance. The whole contract includes startup and the aligned controller requirements, so its failures are not an additional disjoint error count.

		Startup			Whole contract		
Model	Initializes	Passed	A	R	Passed	A	R
Qwen3.5-9B BF16	Framework	33/36	V	P	30/36	V	P
	Model	36/36	P	P	36/36	P	P
Qwen3.5-9B Q8_0	Framework	33/36	V	P	30/36	V	V
	Model	34/36	V	V	34/36	V	V
Qwen3.5-9B Q6_K	Framework	32/36	V	V	29/36	V	V
	Model	28/36	V	V	28/36	V	V
GLM-4.7-Flash MXFP4_MOE-no-MTP	Framework	36/36	P	P	23/36	V	V
	Model	15/36	V	V	14/36	V	V

The protected startup/whole-contract sets contain 11/10 identities for framework-initialized Qwen3.5-9B and 12/12 for model-initialized Qwen3.5-9B. For GLM-4.7-Flash, the corresponding sizes are 12/10 and 5/5. Each protected identity is required in every round. The Qwen3.5-9B entries are the BF16 realization and its directly derived Q8_0 and Q6_K formats, distinct from the pre-existing Q8_0 in the cross-model table.

Of the 288 episodes, 247 satisfy clean startup and enter downstream controller evaluation. The other 41 violate startup and retain NA downstream timing labels; their whole-contract failure is already established by the startup obligation.

The derived Qwen3.5-9B Q8_0 completes its second model-initialized round with all episodes conformant but violates a protected startup requirement in rounds 1 and 3. The phase-selection case in the software-infrastructure domain provides a direct repeat-level witness. In model-initialized execution, Qwen3.5-9B Q8_0 starts the review gate together with the two inactive-phase gates in rounds 1 and 3, but starts only the required review gate in round 2. Qwen3.5-9B Q6_K fulfills this startup obligation in rounds 1 and 2, then waits without starting a task in round 3, reproducing the recorded initialization violation discussed above. A passing repetition therefore coexists with a verified protected violation under the registered conditions. These within-configuration changes describe invocation variability, separately from the cross-format repair/regression comparisons.

Repeated initialization-before-wait violations. Two business-operations identities have a stronger recurrence witness: `phase_selection` and `disjunctive_join`, both under `model_initialized`. Their required sets are respectively {`review_gate`} and {`gate_a`, `gate_b`}. In all three Qwen3.5-9B Q6_K rounds, the first call is `wait_for_event`, the realized set is empty, and the environment rejects the call with `unspawned_task_event`. The registered first deviation is `invalid_tool_call` at the startup boundary; downstream timing remains NA. The corresponding BF16 rounds realize the required sets and conform downstream. Under this allocation, conjunctive-join and nested-result-guard startup traces also conform in both formats across all three domains and rounds, locating the repeated omission within the registered structural support.

The two repeated witnesses are at `/results/5` and `/results/17` in each raw artifact linked by the `rounds` entries of `infra_code/experiments/d4_qualification_20260911_decisions.json`; select `model_key=qwen_bf16` or `qwen_q6` and repeats 1–3. Each entry retains the original request IDs. The software-infrastructure phase-selection variation above and these consistently failing business-operations cases are different identities; both remain in the complete verdict ledger.

All 1,041 model-request/response pairs, including protocol anchors, are linked to retained journals; all 288 episodes have complete replay evidence. Per-round serving-process witnesses and request identities distinguish fresh runs from reused responses. The machine-readable decisions in `infra_code/experiments/d4_qualification_20260911_decisions.json` retain every violated identity, its repeat, raw-artifact row, first-deviation evidence, and request references. The phase-selection witness is located by `case_id=phase_selection`, `domain=software_infra`, and `allocation=model_initialized` in the corresponding rounds.

The collection command below composes the existing protocol qualifier and initialization runner; it does not replace their prompt, scheduler, or scoring code. It requires the registered GPU to be idle and records each fresh server, preflight, raw artifact, request journal, and replay audit. The analysis command regenerates allocation-specific startup and end-to-end verdicts, retaining individual violated and unresolved identities. From the repository root, using the configured Python environment:

```
export PYTHONPATH=infra_code/src:infra_code/experiments
python infra_code/experiments/run_d4_qualification.py
python paper/analysis/qualify_initialization.py \
  infra_code/experiments/d4_qualification_20260911_runs.jsonl
```

The collection ledger and per-round evidence remain under `infra_code/experiments/`. Re-analysis is read-only by default. Resuming collection requires an explicit `--resume` and preserves recorded rounds, including unsuccessful ones; it does not rerun them according to observed scores.

Read-only re-analysis locates archived experiment files in the current checkout. Recorded checkpoint and serving paths remain provenance fields, so replay does not require loading the model weights or reproducing the original machine's directory layout.

Ten-round recurrence extension. After the original three rounds, `infra_code/config/d4_qualification_20260912_ten_rounds.json` registers seven additional fresh-server rounds for Qwen3.5-9B BF16 and Q6_K. The extension keeps all 24 identities per configuration, both allocations, the original protected sets, and the same serving and experimental conditions. It adds 336 episodes to the 144 retained episodes for these two configurations. Thus the ten-round evidence comprises three previously observed and seven prospectively registered rounds; the original four-configuration report above remains unchanged.

Table 19 enumerates every identity with a whole-contract violation in these ten rounds. Within each listed identity, every failing round has the same first-violation witness: the offending tool and arguments, lifecycle result, and execution epoch agree. Passing and failing rounds can still alternate. In particular, Qwen3.5-9B Q6_K's business-operations disjunctive case passes startup and downstream execution in round 10 after the same omitted-startup violation in rounds 1–9. Its business-operations phase-selection case repeats that violation in all ten rounds.

Table 19: Identity-level recurrence over ten initialization rounds. Entries count violated whole contracts in the original three and additional seven rounds, separately. Every identity with any violation is shown; the remaining 38 configuration–allocation–identity combinations pass all ten rounds. Wait before spawn and repeated spawn violate startup; early action follows admitted startup.

Model	Initializes	Obligation / presentation	Original	Added	First violation
Qwen3.5-9B BF16	Framework	AND / business	3/3	7/7	Early action
	Framework	Nested guard / neutral	3/3	7/7	Repeated spawn
Qwen3.5-9B Q6_K	Model	Phase / business	3/3	7/7	Wait before spawn
	Model	OR / business	3/3	6/7	Wait before spawn
	Model	Phase / neutral	1/3	2/7	Wait before spawn
	Model	Phase / software	1/3	3/7	Wait before spawn
	Framework	OR / business	3/3	7/7	Early action
	Framework	Nested guard / neutral	2/3	6/7	Repeated spawn
	Framework	Phase / neutral	1/3	1/7	Repeated spawn
	Framework	Phase / software	1/3	1/7	Repeated spawn

For the intermittent Qwen3.5-9B model-initialized phase cases, the neutral presentation violates startup in rounds 2, 5, and 10, and the software-infrastructure presentation in rounds 3, 6, 7, and 8.

Under framework initialization, the nested neutral case repeats an already-started task in rounds 1, 3, 4, and 6–10; the neutral and software phase cases do so in rounds 1 and 9. The two always-violated framework-initialized timing witnesses act at epoch 0: the BF16 business-operations AND case requires epoch 2, and the Q6_K business-operations OR case requires epoch 1.

Qwen3.5-9B BF16 passes all twelve model-initialized identities in every round. Qwen3.5-9B Q6_K passes all eight model-initialized identities outside the four listed cases in every round, including every conjunctive and nested-guard presentation. The two configurations respectively retain ten and eight always-passing framework-initialized identities. These all-ten-pass observations and the violation sequences use the complete registered support. All allocation- and stage-specific absolute and regression verdicts for these two configurations remain unchanged from the three-round report; the extension refines the recurrence evidence underlying them.

All 480 episodes pass request/response linkage and semantic replay, and every required outcome is decidable. Their 1,822 distinct HTTP requests comprise 547 from retained rounds and 1,275 from new rounds, including the separately audited protocol anchors. Each of the twenty model-rounds uses a distinct serving process. `infra_code/experiments/d4_qualification_20260912_ten_rounds_decisions.json` maps every identity and repetition to its original artifact, row, request IDs, and stage-specific verdicts. The per-identity all-ten-pass indicator describes these observed executions under the registered conditions, rather than a population reliability guarantee.

The same collection and analysis entries accept the additional registration:

```
cd infra_code
export PYTHONPATH=src:experiments
python experiments/run_d4_qualification.py \
  --config config/d4_qualification_20260912_ten_rounds.json
python ../paper/analysis/qualify_initialization.py \
  experiments/d4_qualification_20260912_ten_rounds_runs.jsonl \
  --config config/d4_qualification_20260912_ten_rounds.json
cd ..
```

F.2 Prospectively fixed composition extension

To test reuse beyond the original fixtures, a separate pre-response registration fixes six new policy compositions: a three-way conjunction, a disjunction of two conjunctions, a conjunction of two disjunctions, two conjunctive result guards, phase selection with a nested result guard, and two branches sharing a task reference. The last case requires starting each distinct referenced task once, not once per textual occurrence. Two original business-operations presentations, the conjunctive join and nested result guard, remain as model-startup passing controls. All eight cases use that one rendering, both initialization allocations, and three fresh-server rounds for Qwen3.5-9B BF16 and its derived Q6_K: 96 episodes, of which 72 concern new compositions and 24 retain the controls. The same RX 7900 XT, serving profiles, and concurrency of two are used for both formats. Format order alternates across rounds; allocation submission order is reversed in round two. No new-case responses were inspected before registration. This is a prospective extension within the existing policy language.

Extension-specific code supplies case definitions, independent checks and a collection/admission adapter. The policy renderer, initialization oracle, reference-trajectory constructor, tool executor, transport, replay and stage classifier are reused without changes to their production source. The extension checks its own eight-case identity grid rather than relaxing the original twelve-identity admission gate. Independently written Boolean expectations and a separate interpreter of the actual policy text agree with the oracle on 79,086 finite states, including pre-completion results, both phases and background-task variation. The sixteen initial prompts also pass independent startup-set interpretation. These are instrument checks, not additional model-response units; the complete pre-collection test suite has 302 passing tests.

Registration protects clean model-initialized startup on the two original control identities. The six new obligations are mandatory tests, not retroactively protected historical passes. Their absolute conformance is reported separately from control preservation. Paired format transitions on the new cases describe the recorded comparisons. The control prompts are byte-identical to their original

records, but the new mixed suite is a different collection workload, not an exact repetition of the original twelve-identity grid. Violations are retained, failed startup leaves downstream timing NA, and missing evidence does not establish passage. No score-selected reruns are permitted.

The specification, independent checks and collection entry points are `infra_code/experiments/route_c_cases.py` and `infra_code/experiments/run_route_c_local.py`. The frozen `infra_code/experiments/route_c_local_20260919/registration.json` binds case definitions, prompts, model and source hashes, serving settings, run order and analysis rules. Per-run artifacts retain tool traces and full responses; separate HTTP journals connect each recorded response to its invocation.

Results and interpretation. All six runs completed: 96 episodes and 36 protocol anchors on six distinct serving processes. All 486 HTTP requests have matched responses, without transport retries or unavailable turns; every episode passes manifest and classification replay. Table 20 retains the full case grid. Exact and clean startup agree throughout: all framework-initialized episodes pass startup; model-initialized BF16 passes 15/18 new and 6/6 control episodes, while Q6_K passes 18/18 new and 5/6 control episodes. Downstream admission is conditional on these startup passes.

Table 20: Whole-episode conformance in the composition extension (passing/evaluated rounds). Columns identify the startup actor. Each cell covers three episodes of one identity, not independent cases. The bottom two retained controls protect model-assigned startup only.

	Qwen3.5-9B BF16		Qwen3.5-9B Q6_K	
Composition	Framework	Model	Framework	Model
Three-way AND	3/3	3/3	3/3	3/3
OR of two ANDs	3/3	3/3	3/3	3/3
AND of two ORs	3/3	3/3	3/3	3/3
Two result guards	3/3	3/3	3/3	3/3
Phase plus nested result guard	3/3	0/3	3/3	3/3
Shared task reference	3/3	3/3	3/3	3/3
Original conjunctive join	0/3	3/3	3/3	3/3
Original nested result guard	3/3	3/3	3/3	2/3

For `phase_nested_release`, the active RELEASE clause requires an accepted `release_gate` result (inventory confirmation) and completion of either `approval_gate` (shipping approval) or `legal_review` (compliance review). The declared all-references startup contract derives exactly these three tasks, excluding the two REVIEW-phase tasks and the background task. The initial prompts state the allocation explicitly:

> *Framework initialized:* The environment has completed canonical startup according to InitOracle. Do not call `spawn_task`; continue the workflow directly from the current state.
>
> *Model initialized:* The environment has not started any catalog task. You must first call `spawn_task` according to InitOracle to realize the exact startup set; before that, you must not wait or execute.

The same prompts subsequently state the common operating convention: "When the policy returns WAIT, call `wait_for_event`; when it permits ACT, call `execute_action(action_id="ACT")`; ordinary text is not an environment action." The model-started allocation explicitly requires completing startup before either business-policy operation. In `route_c_local_20260919/qwen_bf16_r1.json`, JSON pointers `/results/8` and `/results/9` locate the framework- and model-initialized records. The former begins with all three tasks PENDING, calls `wait_for_event`, and subsequently acts on time. The latter begins with all three NOT_STARTED. Its first response states:

> All relevant tasks are NOT_STARTED. Therefore, rule 2's IF condition is unsatisfied, and the policy returns WAIT. According to the operating convention, when the policy currently returns WAIT, call `wait_for_event`.

Its first native tool call is `wait_for_event({})`, with no prior `spawn_task`. The scheduler returns `unspawned_task_event`. Thus the business-rule judgment is correct but the assigned startup obligation is violated; downstream timing is NA. The same paired stage outcomes recur in

rounds two and three (rows 9/8 and 8/9, respectively). Both retained model-startup controls pass in all three rounds.

The original phase-selection and nested-guard business presentations each pass model-initialized startup and downstream execution in all ten earlier rounds. The former selects the REVIEW clause and starts `review_gate`; the latter starts `risk_check`, `legal_review` and `manager_override` before waiting. The retained nested-guard control also follows this ordering in all three extension rounds. The new fixture uses the RELEASE phase, a nested active clause, different task names and a larger catalog. Here the response correctly selects the active clause and evaluates its business condition, but proceeds directly from WAIT to the waiting tool, skipping the required startup. The observed break is therefore between business rule evaluation and the prior action obligation, rather than an incorrect WAIT judgment. Appendix C.3 derives the corresponding allocation-specific judgments from this pair.

Q6_K's original nested-guard model-startup control fails in round one and passes in rounds two and three. The failure also begins with waiting before any spawn, leaving downstream timing NA. Its new-case passage therefore does not satisfy the registered preservation requirement. BF16 preserves the two protected startup controls but fails the expanded mandatory set. The sole admitted downstream violation in this extension is BF16's framework-initialized conjunctive control, analyzed next. Historical control prompts match, but workload and invocations differ, so the new control failure is not an isolated causal effect of composition or numerical format.

Aligned-startup timing contrast. BF16's original business-operations conjunctive control has the opposite allocation contrast. Both allocations pass clean startup and admit downstream scoring, but framework initialization is followed by execution at epoch 0, while model initialization is followed by waiting until the required epoch 2. This paired outcome holds in all ten rounds indexed by `infra_code/experiments/d4_qualification_20260912_ten_rounds_decisions.json` and all three composition-extension rounds. In each pair, the framework's epoch-0 task-state hash equals both the reference hash and the model's hash after its last startup call: both gates are PENDING and the auxiliary task is NOT_STARTED. Their common decision oracle requires WAIT at that state. The different timing outcomes therefore follow aligned startup, not a failure of the framework to establish the required task set.

In `route_c_local_20260919/qwen_bf16_r1.json`, rows 12/13 locate the framework/model pair; the corresponding rows are 13/12 in round two and 12/13 in round three. The framework-condition response in round one describes each gate as "status=PENDING (completed)," then executes. The model-condition response instead identifies the initial unmet condition, starts both gates, and waits for their completion. The common task state and oracle make the downstream decisions comparable, while the allocation instructions and prior tool interactions remain different parts of the execution path.

`infra_code/experiments/route_c_local_20260919/summary.json` records each identity, repetition, stage outcome and source-row pointer; per-run `_audit.json` files retain replay and request-linkage checks.

F.3 Startup across matched policy forms

The follow-up holds the startup set, task catalog, task names, initial facts, event schedule, and complete reference worldline fixed while crossing two policy factors. Phase context either retains the original RELEASE guard and inactive REVIEW clause or makes the active clause unconditional. The business condition either requires a passing `release_gate` and completion of both `approval_gate` and `legal_review`, or requires a passing `release_gate` and completion of either of those two tasks. The latter is the original nested AND/OR condition. Every cell requires the same three tasks to start, and every reference trajectory has the action sequence WAIT, WAIT, WAIT, ACT. The full phase-plus-nested cell reproduces the earlier composition's initial prompt byte for byte within each presentation and allocation.

The four cells are crossed with neutral-symbolic, software-infrastructure and business-operations presentations, both startup allocations, BF16 and its same-source Q6_K, and three fresh-server rounds. Original phase-selection, conjunctive-join and nested-result-guard controls remain in every run, yielding 42 episodes per format/round and 252 in total. These controls retain their original

obligations; they are not assumed to pass merely because they are controls. Independent truth checks cover 268,912 finite states across the four new policy forms and separately interpret their rendered conditions and startup instructions.

The comparison localizes which explicit policy forms preserve startup and downstream fulfillment on this fixed execution support. The Boolean factor changes AND/OR semantics away from the observed worldline as well as syntactic nesting; the phase factor includes the inactive clause. Consequently, neither factor is a pure text-length or nesting-depth manipulation. A combination-only failure requires the full cell to fail while all three simpler cells pass in the same format, presentation, allocation and round. The registration and complete source-bound records are under `infra_code/experiments/local_completion_20260925/composition_controls/`; `paper/analysis/analyze_composition_controls.py` retains each cell and control, rather than inferring an interaction from marginal totals.

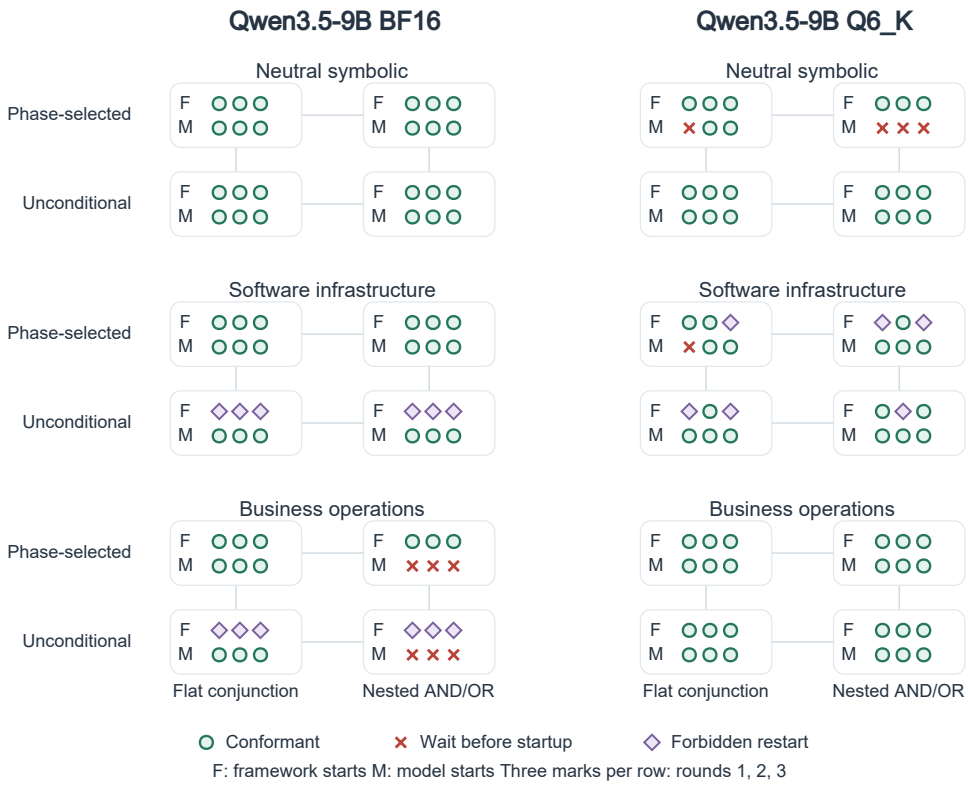


Figure 5: Startup across matched policy forms. Horizontal changes replace a flat conjunction with nested AND/OR; vertical changes introduce phase selection and the inactive clause. The startup set and reference action sequence stay fixed. Each node shows the framework/model allocations in three fresh rounds. All startup passes in these four-cell grids also conform downstream. The 108 retained-control episodes are described separately below.

The BF16 business-operations omission persists in all three rounds with the nested condition, both with and without phase context; both flat-conjunction model-startup cells pass throughout (Figure 5). Without phase context, its response again enumerates the NOT_STARTED tasks, correctly concludes that the policy returns WAIT, and calls `wait_for_event` before any spawn. Thus phase composition is not necessary for the original omission. In the software and business framework-started cells, the same simplification instead introduces forbidden startup calls in all three rounds, even though the tasks are already PENDING. These are distinct obligations: start when assigned, and refrain from restarting when the framework has already started the tasks.

Q6_K localizes differently. All four business-operations model-startup cells pass, but the neutral-symbolic phase-plus-nested cell omits startup in all three rounds. Its three simpler matched cells all pass in rounds two and three; in round one, the phase-selected flat cell also omits startup. The

same combined fixture therefore exhibits recurring failure and a combination-only contrast in two of three rounds, not a universal ordering by policy complexity. No task renaming or worldline change is needed to observe these differences within a presentation.

The retained controls provide 108 additional episodes. Both formats violate the neutral-symbolic framework-started nested control by restarting a task in all three rounds. BF16's framework-started business conjunctive control retains clean startup but acts prematurely in all three rounds. Q6_K omits model-assigned startup in the software and business phase controls in all three rounds; its neutral-symbolic framework-started phase control acts prematurely only in round two. Every other retained-control episode conforms. All 252 episodes pass source, response, prompt and replay checks; the request journals account for 1,030 logical calls including protocol anchors. These checks establish the evidence used by the stage judgments, not passage of the behavioral requirements.

F.4 Separating initial framing from continuing state

The original tool contrast changes both the initial heading and subsequent state exposure. To separate these components, the follow-up crosses two truthful headings—authoritative current state and initial snapshot—with two ongoing observation modes—events with current state and events alone. Within each heading, the two modes have identical initial prompts. Policy, initial facts, tools, startup, reference trajectories and scoring remain fixed on the original 81 identities. The registered historical history-only heading additionally promises that subsequent changes arrive through events only; that promise is not crossed with ongoing state supply. An exact copy of this history-only interface is retained as a fifth arm. The authoritative-heading/state arm likewise retains the historical supplied-state interface.

Both original local configurations, Qwen3.5-9B's pre-existing Q8_0 and GLM-4.7-Flash MXFP4_MOE-no-MTP, execute all five arms in each of three fresh-server rounds: 405 episodes per configuration/round, or 2,430 overall. The registration audits all 405 initial prompts and 306 matched observation prefixes before collection. Analysis reports ongoing-state contrasts within each heading, heading contrasts within each observation mode, the exact historical-interface contrast, and the exclusive-events wording contrast. All are paired on the same 81 identities within a configuration and round; the comparison does not subtract rates on different admitted subsets. The registration and journals are under `infra_code/experiments/local_completion_20260925/framing_controls/`; `paper/analysis/analyze_framing_controls.py` retains paired outcomes for each contrast, separating the effects of interface components at the observation boundary under the tested renderings.

All six runs complete the registered support without unavailable responses, and all 6,456 logical requests, including anchors, pass source and replay checks. The original full-interface contrast repeats with opposite signs: supplied-minus- withheld differences are $+17, +20, +18$ passing identities for Qwen3.5-9B Q8_0 and $-5, -2, -6$ for GLM-4.7-Flash, each out of 81 pairs. The factorial comparisons separate this observation from an isolated ongoing-state interpretation (Figure 6). With the authoritative heading fixed, adding state yields $+9, +9, +10$ for Qwen and $+1, 0, +3$ for GLM; with the snapshot heading fixed, the differences are $+16, +15, +13$ and $+6, +8, +11$. GLM's negative complete-interface contrast therefore does not recur as a negative net effect of adding state under either fixed heading. Removing the explicit events-only promise also changes direction across configurations: Qwen gains $3, 7, 6$ passes, whereas GLM loses $8, 4, 10$.

Net gains do not establish obligation preservation. Under the authoritative heading, GLM gains and loses $(6, 5)$, $(4, 4)$ and $(8, 5)$ passing identities in the three rounds; the zero-total-change round thus exchanges four passes in each direction. Under the snapshot heading, its corresponding counts are $(10, 4)$, $(10, 2)$ and $(13, 2)$. Qwen's authoritative-heading contrast loses one pass in round one while gaining ten; its other five within-heading state contrasts lose none. Interface choice must therefore consider which obligations are repaired or violated, not only the sign of the net contrast. The registered analysis retains every paired identity for all six contrasts and rounds.

F.5 Larger-model tool follow-ups

DeepSeek V4 Pro 0813 (Baidu FP8) and MiniMax M3 (CoreWeave FP4) each execute the complete 81-identity tool support under three interfaces: authoritative heading with ongoing state, that same

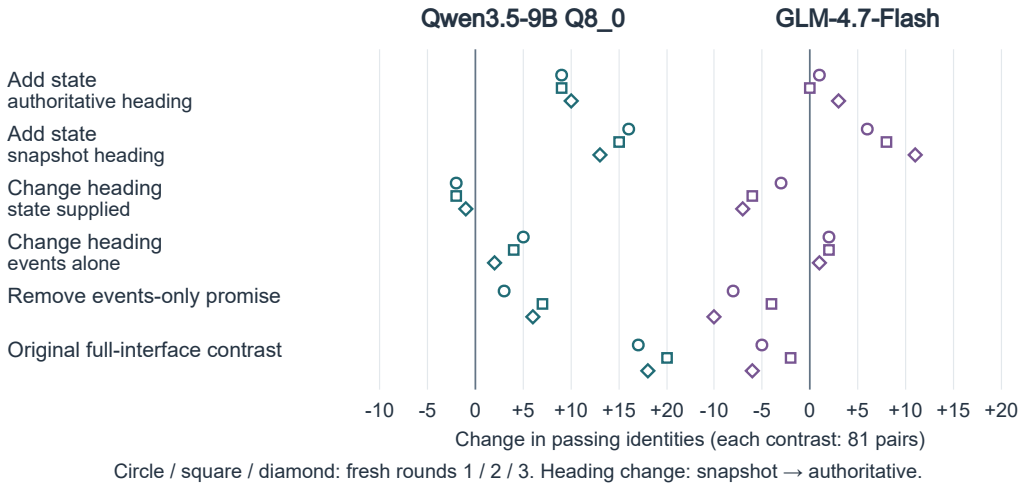


Figure 6: Separating interface components on the same 81 tool-execution identities. Each mark is one fresh round; positive values denote a net increase in passing identities, not absence of regressions. Heading changes replace "initial snapshot" with "authoritative current state." Removing the events-only promise compares the original history interface with the snapshot-heading/events arm. Rows share experimental arms; neither rows nor rounds are new independent cases.

heading with events only, and the original history-only interface. Each release uses one fixed provider route, temperature zero, a 4,000-token response cap and disabled reasoning; receipts report zero reasoning tokens. The first two arms have identical initial prompts. Each model contributes 243 episodes in one invocation round, with the same tools, canonical startup and reference worldlines.

In that arm order, DeepSeek V4 Pro passes $73, 75, 75$ episodes and MiniMax M3 passes $79, 80, 81$. With the authoritative heading fixed, the supplied/events-only paired cells (both pass, supplied only, events only, both fail) are $(72, 1, 3, 5)$ for DeepSeek V4 Pro and $(78, 1, 2, 0)$ for MiniMax M3. Against the original history interface they are $(71, 2, 4, 4)$ and $(79, 0, 2, 0)$. These are complete identity-matched contrasts, not comparisons of differently selected admitted subsets.

MiniMax's two supplied-state failures are delayed OR actions in the far business-operations and software-infrastructure renderings of `join_or_10/base`. After the second gate completes, one completed branch already authorizes the business action at epoch 1; the model waits again and acts at epoch 2. Its response describes the OR as requiring both checks. Both corresponding events-only interfaces act on time. The authoritative-heading/events-only arm instead misses the business action on the business-operations `e1` variant, where the second branch alone still suffices. Thus fixing the heading does not eliminate configuration-specific state-support failures or make the passing identities interchangeable.

DeepSeek V4 Pro's eight supplied-state failures and five failures in each events-only arm are execution omissions on never-act trajectories: the model ends normally without realizing the next required WAIT through its tool. There is no business action to miss. The binary violations are retained, while the raw classifier's `missed` label is interpreted as `unrealized_wait`, as in the earlier witness above. Each events-only arm additionally has one delayed OR action.

Both larger models also complete three rounds of the initialization union: the 24 original episodes, 16 extension/control episodes and 42 matched-constituent roles share identities, giving 58 unique episodes per round and 174 per model. DeepSeek V4 Pro passes all 174 episodes across both allocations. MiniMax M3 passes every original and extension role in all three rounds, and every constituent role in rounds 1 and 2. In round 3, the neutral-symbolic `phase1_nested0` model-started episode fails. The current phase selects the release rule, whose startup set is {`approval_gate`, `legal_review`, `release_gate`}. All three are NOT_STARTED. The response correctly states that their conjunction is unsatisfied and the business policy returns WAIT, but says "the current phase is already V, indicating that initialization has passed." Its first call is `wait_for_event`; the environment rejects the call because required tasks are unstarted. The same identity passes under

framework startup and in the first two model-started rounds. The first violation is omitted startup, leaving downstream timing unscored.

Registrations, retained responses and request journals are under `infra_code/experiments/remote_extension_20260925/`. `paper/analysis/analyze_remote_tool_study.py` verifies full support, prompt/profile binding, all 2,720 logical requests (including tool anchors), and shared semantic replay. It retains raw outcomes and the source-linked execution-omission interpretations separately.

F.6 Sequential evidence construction

A sequential input comparison uses the same pre-existing Qwen3.5-9B Q8_0 checkpoint and one recorded serving profile for two history constructions. The historical renderer reconstructs an event description from the state at each epoch; the append-only renderer retains the actual cumulative event prefix, including wake boundaries without new facts. Both reconstruct the same reference states and share the action oracle. Across the 612 conditional request positions, 252 prompt strings differ and all initial prompts match. Independent text interpretation and state reconstruction check the full 1,224-prompt manifest before responses.

Each of three fresh-server rounds collects the complete 81-trajectory supplied-state and history-only supports under both constructions, yielding 972 trajectories. Construction order alternates across rounds. Previous decisions in collected prompts come from the actual responses, not the oracle; the conditional manifest enumerates reachable WAIT prefixes without supplying those decisions during collection. The comparison preserves the checkpoint, server, device, template, decoding, policy, action vocabulary, observation horizon and transcript representation. Its registration, immutable source snapshot and raw journals are under `infra_code/experiments/local_completion_20260925/history_bridge/`. `paper/analysis/analyze_history_bridge.py` reports identity-level changes between input constructions and the supplied/withheld contrast within each. The two input versions remain separately identified in both collection and replay; these fresh observations do not replace the historical responses.

The supplied-state advantage recurs under both constructions (Table 21). Changing the history input changes individual outcomes but does not remove the history-only violations. On the history-only support, the append-only construction repairs three, four and two identities and regresses one, zero and zero, respectively. These are within-round input contrasts, not additions to the earlier tool-interface repetitions.

Table 21: Whole-contract conformance under two sequential history constructions, each on 81 registered identities. S/H denotes supplied-state/history-only. Differences are paired numerators on that complete support, not pooled trials.

	Per-epoch state-derived history			Append-only event history		
Round	S	H	S−H	S	H	S−H
1	74/81	50/81	+24	76/81	52/81	+24
2	74/81	49/81	+25	76/81	53/81	+23
3	74/81	52/81	+22	75/81	54/81	+21

In round two's append-only supplied-state arm, `join_or_10/e1/neutral_symbolic/neutral` returns only "action name:" at epoch 1. The complete HTTP 200 response has `finish_reason=stop`; source-bound reading establishes that no action was selected. This is one response-obligation failure, with action and timing judgments unavailable, rather than a substituted WAIT or ACT. All 81 whole-contract outcomes are determined; 80 admit action/timing interpretation. The paired history-only identity passes. The original collector record retains its pending-reading fields, and `paper/analysis/analyze_reviewed_history_bridge.py` applies the separately recorded judgment without changing that source. The linked review retains the oracle-free reading packet, assisted outputs, final readings and completion receipt separately.

F.7 Complete local Qwen serving-profile follow-up

Three further fresh-server selected rounds use the exact model file, server binary, device, chat template, launch arguments and environment of the history-construction study. Each collects the full 864 responses: 72 four-answer units in each of the state-only, history-plus-state and history-only conditions. The registration and source snapshot precede all 2,592 responses. Every round passes 72, 52 and 41 units, respectively. The paired supplied/withheld cells (both pass, supplied only, withheld only, both fail) are $(39, 13, 2, 18)$ in rounds 1 and 2 and $(40, 12, 1, 19)$ in round 3. Equal marginal counts therefore do not imply identical passing identities.

Two round-three responses need explicit readings: "action name: wait" selects WAIT; an explanation ending "should execute start data parsing" selects ACT. The readings use the action labels and retained answer without an oracle, are recorded separately as assistant interpretations, and are linked to original prompt, response and request hashes. Replay uses these choices without another model invocation; the original responses and pending-reader fields are unchanged. Both responses violate the name-only output obligation. One also selects the wrong action; the other reduces round-three point conformance from 261 to 260 supplied-state answers, without changing the 52 passing strict units or their paired identities. All selected unit judgments are determined; the semantic readings and complete response judgments are retained separately.

Table 1's Qwen entry reports round 1 of these selected observations and the append-only sequential arm of Table 21. Across all three sequential rounds, supplied passes are $76, 76, 75$ and withheld passes $52, 53, 54$ out of 81, giving paired differences $24, 23, 21$. For each condition, pass^3 counts the intersection of passing identities across all three rounds, while Mean averages the three passing counts on that same support. Selected pass^3 counts are $72/72$, $51/72$, and $41/72$ for state-only, history-plus-state, and history-only; the corresponding means are 72, 52, and 41 out of 72. Sequential pass^3 counts are $74/81$ and $52/81$, with means $75.67/81$ and $53/81$. Thus equal selected totals need not preserve identical passes across rounds. These are separate invocation series and separately paired supports, not itemwise selected-to-sequential matches or replacements of historical responses. Appendices D.1 and D.2 distinguish these rounds from the archived selected realization. The analyzer `paper/analysis/analyze_qwen_profile_followup.py` checks serving-profile identity, all three completed-run receipts, source-bound interpretations and full support. `paper/analysis/check_completed_text_tables.py` verifies the first-round display and repeated summaries against the source-bound identity judgments without rescoring responses.

F.8 Matched reasoning-mode diagnostic

The diagnostic retains the `gating_01` four-atom unit in the neutral-symbolic rendering under state-only, history-plus-state and history-only inputs: twelve prompts per release and mode, not twelve independent structures. DeepSeek V4 Flash 0731 uses the DeepInfra FP8 route; DeepSeek V4 Pro 0813 uses Baidu FP8. Each release retains its route, temperature zero, 2,048-token output cap, prompts and action oracle. One fresh call per prompt under each mode gives 48 responses. The default arm omits the reasoning parameter; the disabled arm adds only `reasoning.enabled=false`. All default receipts report positive reasoning-token usage; all disabled receipts report zero. Every response ends normally with a uniquely interpretable action, distinguishing wrong choices from output truncation.

State-only and history-plus-state each select all four correct actions in both modes and releases. History-only semantic correctness is $4/4$ under default reasoning, versus $2/4$ for disabled DeepSeek V4 Flash 0731 and $3/4$ for disabled DeepSeek V4 Pro 0813. During preparation, a later-stage result must not change the required phase advance. Disabled DeepSeek V4 Flash 0731 instead selects the business action on the base atom; disabled DeepSeek V4 Pro 0813 waits on its changed-result counterpart. Disabled DeepSeek V4 Flash 0731 additionally advances instead of waiting on the later-stage failed-result atom. Default Flash omits an internal space in two listed action names: their meanings are clear but the exact output requirement is violated. Consequently Flash has $10/12$ complete response passes in each mode, with different failure identities and types; Pro has $12/12$ versus $11/12$. The same-identity comparisons distinguish changed policy decisions from output compliance.

The frozen paired registrations, profiles and HTTP receipts are under `infra_code/experiments/endpoint_continuations_20260925/reasoning_control/`.

paper/analysis/analyze_text_reasoning_control.py checks the controlled settings, original prompt identities, action readings, normal completion and reported reasoning usage. These diagnostic observations are separate from the full historical conformance rows.

Low-reasoning follow-ups. GLM 5.3 Flash uses its provider-supported reasoning.effort=low setting in three fixed diagnostic rounds of 46 prompts: three selected units across the three interfaces (36 responses), plus ten retained sequential-prefix prompts. The units include two historical failure identities and a passing control. The rounds respectively contain three, one and four name-only output violations, including explanatory text and duplicated action names. All 138 responses complete normally. Their complete response pass counts are $43/46$, $45/46$ and $42/46$; sequential-prefix probes are not counted as complete trajectories.

Gemini 3.8 Flash's complete low-reasoning panel, described in Appendix D, passes all three selected conditions and both sequential conditions. Its full-panel result is separate from the GLM diagnostic support and from the matched DeepSeek reasoning-mode comparison. paper/analysis/analyze_reasoning_output_obligations.py applies the same response rule to GLM and both DeepSeek modes; the Gemini 3.8 panel uses paper/analysis/analyze_frontier_panels.py.

G Registered format-by-layout deployment extension

G.1 Design, format provenance, and execution identity

Execution configurations I and II of the custom heterogeneous inference framework denote the two-GPU and three-GPU layouts, respectively, in Table 22. These are realizations within one framework, not two independently developed inference engines. The fixed-weight control retains the two-GPU model partition and replaces only its tail device, as specified in Appendix G.4.

The deployment registration is infra_code/config/qwen38_27b_factorial_20260909.json, with execution profile batched128_parallel2 registered before its formal responses. It crosses Qwen3.8-27B Q4_K_M, Q6_K, and Q8_0 with the two layouts in Table 22. Each cell has three complete fresh-process runs. Every run uses the same 864 selected rows and 162 sequential trajectories: 72 four-atom strict units in each of three selected conditions, and 81 trajectories in each sequential condition. This extension retains the original obligations and renderings; the repeated runs measure execution stability on those identities.

The registration identifies the retained Qwen3.8-27B GGUF files. Q6_K is directly quantized from the local BF16 file by the recorded llama-quantize implementation, with GPU visibility disabled during quantization. Its derivation log, Qwen3.8-27B-Q6_K.quantize-20260909.log, is included in the artifact's serving/evidence/ directory. It records 851 source tensors, 498 quantized tensors, and 353 byte-preserved F32 tensors. Q4_K_M and Q8_0 are the pre-existing format files bound by the registration. Their tensor names and shapes agree with BF16 and Q6_K. Q4_K_M contains 433 Q4_K and 65 Q6_K tensors alongside 353 F32 tensors; Q8_0 uses 498 Q8_0 and 353 F32 tensors. Q8_0 denotes integer eight-bit GGUF quantization, not FP8. Across formats the comparison concerns these registered GGUF realizations; within a format, every layout reuses the same weight tensors.

Table 22: Ordered GPU assignments and zero-indexed layer placement for the Qwen3.8-27B deployment extension. All layer cuts preserve the model's four-layer hybrid attention cycle.

Layout	Device	Architecture	Layers
Two GPUs	AMD Radeon RX 9070 XT	gfx1201	0–31
	AMD Radeon RX 9060 XT	gfx1200	32–63
Three GPUs	AMD Radeon RX 9070 XT	gfx1201	0–31
	AMD Radeon RX 9060 XT	gfx1200	32–47
	AMD Radeon RX 7900 XT	gfx1100	48–63

The split engine copies shard tensor payloads byte for byte, applies contiguous layer segments, and transfers the intermediate hidden states as F32. Each worker has independent sequence state for two request slots. Prefill uses chunks of up to 128 tokens, with per-worker RPC serialization and up to two

concurrent requests. Each request has a 4,096-token context, greedy decoding, reasoning disabled, and native end-of-generation stopping. The original chat template is applied by the native Jinja path before tokenization. The collector serially loads one inference engine at a time without mmap and records readiness before collecting responses. Policy, history construction, action vocabulary, oracle, and scorer are unchanged.

The runtime inventory is recorded before each run: WSL kernel `6.18.33.2-microsoft-standard-WSL2`, ROCm 7.2.0, HIP runtime `7.2.26015.70200-43~24.04`, and AMD display driver `32.0.31041.1004` on all three cards. The two- to three-GPU contrast jointly changes the device set and the tail partition. Its estimand is the registered layout change; the ordered assignments make subsequent device-replacement or order-swap contrasts separately specifiable.

G.2 Stable phase-guard witness and matched controls

The `gating_01` unit instantiated in Appendix A.3 has a recurring violation at `t_e1`, the rejected-result preparation atom: Qwen3.8-27B answers `WAIT` where `NEXT` is required. This response occurs in the history-only near and far renderings of business operations, software infrastructure, data research, and support access. The same eight presentations exhibit this exact violation in every Q4_K_M, Q6_K, and Q8_0 run on both registered layouts, including all three repetitions per cell, and in the three additional tail-device runs. All corresponding local-anchor and history-plus-state atoms, and the neutral-symbolic history-only atom, correctly return `NEXT`.

For the business-operations far rendering, the raw selected rows are `/abc_results/72` (preparation, approved), `/abc_results/81` (preparation, rejected), `/abc_results/90` (execution, approved), and `/abc_results/99` (execution, rejected). Across the repeated deployment configurations, the model's actions are respectively `NEXT`, `WAIT`, `ACT`, and `WAIT`, against oracle actions `NEXT`, `NEXT`, `ACT`, and `WAIT`. The preparation pair therefore violates required invariance, while the execution pair preserves the required sensitivity. The original prompts, submitted inputs, and responses are retained in every run's artifact and request journal; the engine journals and repeat contrasts establish fresh-execution recurrence.

These presentations instantiate one registered latent case. The witness identifies a shared failure under an inactive phase guard, with its support and passing controls explicit. The complete exports additionally retain other violations, including an approved-result preparation error in the far data-research rendering under Q6_K and Q8_0; recurrence of the highlighted rejected-result failure does not replace those records.

G.3 Complete paired results and repeated witnesses

Every format/configuration cell passes all 72 local-anchor units, 72 history-plus-state units, and 81 supplied-state trajectories in each round, providing controls for the history-only witnesses. Corresponding response-token sequences, including native end-of-generation, are identical across repetitions. Across configurations I and II, Qwen3.8-27B Q4_K_M changes only the approved-result response; Q6_K and Q8_0 preserve every response-token sequence.

Table 23: Qwen3.8-27B conformance across numerical formats and execution configurations of the custom inference framework. Counts are per run; all three runs reproduce the same pass identities. Selected counts use four-atom strict units and sequential counts use trajectories. Every row also passes all 72 local-anchor units, all 72 history-plus-state units, and all 81 supplied-state trajectories.

	Configuration I, history only		Configuration II, history only	
Model format	Selected	Sequential	Selected	Sequential
Qwen3.8-27B Q4_K_M	62/72	74/81	63/72	74/81
Qwen3.8-27B Q6_K	61/72	74/81	61/72	74/81
Qwen3.8-27B Q8_0	61/72	74/81	61/72	74/81

Table 24 gives every format and layout contrast for one round. The same cells and changed identities recur in rounds 2 and 3. All local-anchor and history-plus-state selected comparisons have cells

$(0, 0, 0, 72)$; all supplied-state sequential comparisons have $(0, 0, 0, 81)$. These supports remain separate from history-only conformance.

Table 24: Qwen3.8-27B history-only paired outcomes: F/F and P/P denote retained failures and passes. An arrow specifies the before-to-after direction; an unarrowed format or layout is held fixed. Counts are per round, reproduced in all three rounds.

		Selected ($N = 72$)				Sequential ($N = 81$)			
Format	Layout	F/F	Repair	Regress	P/P	F/F	Repair	Regress	P/P
Q4_K_M	Two → three	9	1	0	62	7	0	0	74
Q6_K	Two → three	11	0	0	61	7	0	0	74
Q8_0	Two → three	11	0	0	61	7	0	0	74
Q4_K_M → Q6_K	Two	10	0	1	61	6	1	1	73
Q4_K_M → Q8_0	Two	10	0	1	61	6	1	1	73
Q6_K → Q8_0	Two	10	1	1	60	7	0	0	74
Q4_K_M → Q6_K	Three	9	0	2	61	6	1	1	73
Q4_K_M → Q8_0	Three	9	0	2	61	6	1	1	73
Q6_K → Q8_0	Three	10	1	1	60	7	0	0	74

For format f, layout l, and repeat r, let $Y_{i,f,l,r}$ be the conformance indicator for registered unit i. Within each condition, the format-by-layout interaction for formats f and g is

$$I_{f,g,r} = \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} \left[(Y_{i,g,3,r} - Y_{i,g,2,r}) - (Y_{i,f,3,r} - Y_{i,f,2,r}) \right],$$

where $\mathcal{J}$ is the shared complete support of strict units or trajectories for that condition. Relative to Q4_K_M, both Q6_K and Q8_0 give $-1/72$, or -1.39 percentage points, for history-only selected conformance in each repeat. Q6_K versus Q8_0 gives zero, as do all other conditions. The two nonzero format comparisons share one underlying changed unit, identified below. Repetitions reproduce this witness rather than supplying additional independent latent structures.

A layout-dependent phase-guard decision. The changed Q4_K_M unit is `gating_00`, data-research domain, far lexical form, history-only condition. Atom `t_e0` is at source pointer `/abc_results/292`. The event history enters preparation, starts schema validation and a background quality report, and records both completions, with schema validation passed. It contains no transition out of preparation. The policy requires entering the next phase during preparation; parsing is authorized only in the execution phase. The two-GPU answer is "start data parsing," whereas the three-GPU answer is "enter the next phase." The former selects a valid action outside its phase guard. All three independent process runs reproduce their respective answers. The submitted request, rendered prompt, and input tokens match across the two layouts; 1,260 of the 1,261 output-token sequences also match, with this response the sole difference. Q6_K and Q8_0 each have 1,262 corresponding requests and identical outputs across their two layouts.

The Q6_K-to-Q8_0 selected repair and regression involve the near software-infrastructure and near data-research renderings of that same latent case, at `/abc_results/291` and `/abc_results/293`. Both require the phase-advance action. Q8_0 repairs Q6_K's unauthorized deployment action in the former rendering but substitutes a parsing action for Q6_K's correct phase advance in the latter. These are two surface realizations of one structure. In sequential execution, changing Q4_K_M to Q6_K or Q8_0 repairs `join_and_01`, support-access, near, variant `e1`, while regressing `gating_10`, software-infrastructure, near, variant `e1`. The first becomes correct-no-action; the second misses the action required at epoch 2 throughout the remaining observation horizon. Their raw trajectories are at `/d_results/124` and `/d_results/48`, respectively.

Request evidence and complete deviation exports. The 18 runs contain 15,552 selected responses and 2,916 trajectories, with 7,158 reached sequential decision epochs. All 22,710 formal HTTP request/response pairs pass semantic replay and transport linkage, with zero prompt or ground-truth mismatches. The engine journals bind the rendered prompt, 3,849,312 total input tokens, emitted token sequences, native stopping, and 250,266 worker-stage trace events. Forty-five distinct GPU-worker process identities witness fresh two- or three-worker realizations across the 18 runs. The three

runs within each format-layout cell have identical corresponding input and output tokens, including end-of-generation.

The deviation exports enumerate all 213 selected-action mismatches and 126 trajectory first deviations, for 339 records across the runs. These are response and trajectory records, including their repetitions, not 339 independent tasks. Each export retains the source pointer, prompt, response, oracle comparison, and linked request. The single-run exports establish replayability; the cross-run token comparisons and fresh-worker witnesses supply the separate evidence that the same deviations recur under new executions.

G.4 Fixed-partition tail-device control

The separate `tail_control_parallel2` profile in the same registration specifies three new Qwen3.8-27B Q4_K_M runs. Its reference consists of the three completed `dual_01` runs from `batched128_parallel2`; the new `dual_02` layout retains RX 9070 XT on layers 0–31 and replaces RX 9060 XT with RX 7900 XT on layers 32–63. Both layouts reuse the same two shard files. The layer cuts, engine binary, serving settings, runtime versions, prompt construction, oracle, and scorer remain fixed. The contrast evaluates tail-device replacement under this recorded software stack. Its run count and stopping rule were fixed before the new responses; the three additional runs are reported separately from the 18-run crossed design.

In each reference-to-candidate comparison, history-only selected conformance changes from 62/72 to 63/72, with paired cells $(9, 1, 0, 62)$. History-only sequential conformance changes from 74/81 to 73/81, with cells $(7, 0, 1, 73)$. Cell order is retained failure, repair, regression, retained pass, as in Table 24. Local-anchor and history-plus-state comparisons each retain $(0, 0, 0, 72)$, and supplied-state sequential comparisons retain $(0, 0, 0, 81)$. Every cell and changed identity recurs in all three comparisons.

The selected repair is the phase-guard witness at `/abc_results/292` described above. The sequential regression is a distinct conjunctive-join obligation: `join_and_01`, business-operations domain, far lexical form, variant `e1`, history-only condition, at `/d_results/117`. Shipping requires both fraud review and inventory confirmation to complete. At decision epoch 3, inventory confirmation and an auxiliary demand-forecast report have completed, but fraud review remains pending. The reference answers "wait" throughout the four-epoch horizon; the replacement answers "ship" at epoch 3 after three waits. Because the shipping condition never becomes true within this trajectory, its outcome is spurious execution rather than execution before a later required epoch. The phase-guard repair and conjunctive-join regression concern two distinct latent structures.

Each comparison has 1,261 corresponding requests with identical submitted inputs, rendered prompts, and prompt tokens. Of these, 1,259 have identical output-token sequences; the two differences are precisely the selected repair and sequential regression above. Every corresponding input and complete output sequence, including native end-of-generation, is identical across the three new runs. All 3,783 new HTTP request/response pairs pass semantic replay, transport linkage, and engine evidence audits, with zero prompt or ground-truth mismatches. The new runs contain 2,592 selected responses and 486 trajectories with 1,191 reached epochs. Their deviation exports retain all 27 selected mismatches and 24 trajectory first deviations, including repetitions.

The retained analysis is `infra_code/experiments/qwen38_27b_tail_control_parallel2_analysis_20260912.json`. Its `runs` entries identify the three raw artifacts and their request and engine journals; `contrasts` verifies within-configuration repetition; `baseline_comparisons` links each new run to its retained reference and gives paired cells, changed identities, and response-token witnesses. The `_bad_cases.json` exports share each request journal's collection prefix. Using the environment defined below, the existing analysis entry point replays this control without model invocation:

```
python paper/analysis/compare_foundational_runs.py matrix \
  "$DEPLOYMENT_CONFIG" --profile tail_control_parallel2 \
  --output "$AUDIT_DIR/tail-control.json"
```

For each retained `baseline_comparisons` entry, the same module's `compare` function reconstructs the paired results from `paired.baseline.path`, `paired.candidate.path`, `baseline_journal`, and `candidate_journal`. Calling `response_evidence` on each

request/engine-journal pair and then `response_contrast` reconstructs the two changed responses. The source pointers above locate their original prompts, answers, and oracles in every paired raw artifact.

G.5 Retained native and execution-profile controls

The original Qwen3.8-27B Q4_K_M native single-GPU run on RX 7900 XT remains an independent baseline: it passes 63/72 history-only selected units and 75/81 history-only trajectories, as well as every supplied-state unit and trajectory. The raw artifact is `infra_code/experiments/responsibility_full_Qwen3.8-27B-Q4_K_M_llamacpp_20260905_122940.json`; its journal is `infra_code/experiments/foundational_qwen38_27b_q4_20260905_requests.jsonl`. All 1,260 requests pass the available HTTP and semantic audits. Compared with this baseline, the batched three-GPU Q4_K_M run retains every selected pass identity, while the two-GPU run loses the phase-guard unit above. Both batched layouts have history-only sequential cells $(5, 1, 2, 73)$ relative to native execution. The comparison evaluates the native versus partitioned deployment stacks, including their execution profiles.

Three completed serialized-prefill runs are also retained as distinct execution-profile controls. Q4_K_M on three GPUs passes 63/72 history-only selected units and 74/81 history-only trajectories; Q6_K on three GPUs passes 61/72 and 74/81; Q8_0 on two GPUs passes 62/72 and 74/81. Their original records and audits are named in the registration. Relative to its serialized-prefill control, batched Q4_K_M preserves selected outputs but exchanges two repaired and two regressed history-only trajectory identities. Batched Q6_K preserves all corresponding input and output tokens. Batched Q8_0 loses the near data-research phase-guard unit while retaining every sequential pass identity. These valid historical realizations are used as implementation comparisons, separately from the three fresh repetitions of the batched profile.

G.6 Measured runtime and reproduction entry points

Table 25 gives measured inference spans from the first acquired worker RPC to the last completed worker RPC in each full run. These spans include the two-slot workload and its scheduling, and exclude model loading and the final offline audit. They describe elapsed collection performance rather than GPU utilization. Each Q4_K_M run processes 213,690 prompt tokens; each Q6_K and Q8_0 run processes 213,931, with the difference arising from reached sequential epochs.

Table 25: Qwen3.8-27B full-run inference span in seconds: median and observed minimum–maximum over three runs. Every run includes the complete selected and sequential manifests.

Format	Two GPUs	Three GPUs
Q4_K_M	408.6 (407.9–411.8)	405.7 (404.9–408.7)
Q6_K	597.3 (595.9–597.4)	578.3 (572.4–591.5)
Q8_0	1195.9 (1192.0–1200.3)	1100.7 (1096.3–1109.2)

The complete machine-readable analysis is `infra_code/experiments/qwen38_27b_factorial_batched128_parallel2_full_analysis_20260911.json`. Its `runs` entries map each format, layout, and repeat to the raw artifact, HTTP journal, engine journal, runtime inventory, and collection audit. The `contrasts` entries give all 45 paired comparisons: 18 format, 9 layout, and 18 repeat comparisons. The nine `interactions` retain the changed identities for the three format pairs in each round. The corresponding `_bad_cases.json`, `_native_paired.json`, and `_legacy_paired.json` files share the collection prefix of their request journal. Readers can follow case identities and source pointers directly, without manually comparing file digests.

`paper/REPRODUCING.md` gives environment setup and a question-indexed guide to the independent oracle checks, repeated exposure, composition extension, and implementation comparisons. Its commands regenerate paired results from retained records without invoking a model, using new output paths to preserve archived analyses. The source pointers above locate the original prompts, responses, and oracles within those records.

Serving sources and build. The review artifact includes the measured worker, controller, and HTTP bridge in `serving/worker.tar.gz` at revision `f4d2e7e`. The splitter `scripts/split_gguf.py` and modified layer-graph source `src/models/qwen35.cpp` are in `serving/layer-graph-overlay.tar.gz`, from revision `a536efe`. The remaining vendor sources are in `serving/llama.cpp-base.tar.gz`, at llama.cpp revision `2c6b141e`. The graph modification exposes contiguous layer segments and their intermediate hidden states. Native-versus-split CPU/GPU checks and request-reset checks are in the worker's `src/test_batched.py`; `serving/evidence/` retains the registered validation logs, including the 27B dual- and triple-layout HTTP checks.

The retained vendor tree and worker can be built in a separate directory using the registered three-architecture HIP backend. Extract the base and overlay archives into the same vendor directory and the worker into a separate directory; set `VENDOR_SRC` and `SPLIT_SRC` to those directories. `serving/SOURCES.json` records their source revisions and archive hashes. The commands below leave the measured binary and existing sources untouched; they use installed dependencies and local source files:

```
: "${SPLIT_SRC:?Set to the retained worker source checkout}"
: "${VENDOR_SRC:?Set to the patched llama.cpp source tree}"
SPLIT_BUILD=$(mktemp -d)
cmake -S "$VENDOR_SRC" -B "$SPLIT_BUILD/vendor" \
  -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=OFF \
  -DCMAKE_HIP_COMPILER=/opt/rocm-7.2.0/lib/llvm/bin/clang++ \
  -DGGML_HIP=ON -DAMDGPU_TARGETS='gfx1100;gfx1200;gfx1201' \
  -DGGML_HIP_NO_VMM=ON -DGGML_HIP_GRAPHS=ON \
  -DGGML_HIP_MMQ_MFMA=ON -DGGML_HIP_RCCL=OFF -DGGML_BLAS=OFF \
  -DLLAMA_BUILD_TESTS=OFF -DLLAMA_BUILD_EXAMPLES=OFF \
  -DLLAMA_BUILD_TOOLS=OFF -DLLAMA_BUILD_SERVER=OFF \
  -DLLAMA_BUILD_MTMD=OFF -DLLAMA_BUILD_UI=OFF
cmake --build "$SPLIT_BUILD/vendor" --target llama --parallel 16
GGML_LIB="$SPLIT_BUILD/vendor/ggml/src"
c++ -O2 -std=c++17 -no-pie \
  -I"$VENDOR_SRC/include" -I"$VENDOR_SRC/ggml/include" \
  "$SPLIT_SRC/src/srv.cpp" -Wl,--start-group \
  "$SPLIT_BUILD/vendor/src/libllama.a" \
  "$GGML_LIB/libggml.a" "$GGML_LIB/libggml-base.a" \
  "$GGML_LIB/libggml-cpu.a" "$GGML_LIB/ggml-hip/libggml-hip.a" \
  -Wl,--end-group -L/opt/rocm/lib -lamdhip64 -lhipblas -lhipblaslt \
  -fopenmp -pthread -ldl -Wl,-rpath,/opt/rocm/lib \
  -o "$SPLIT_BUILD/srv-hip-batched"
```

The splitter takes positional arguments `SOURCE OUT_A OUT_B 32` for two shards and `SOURCE OUT_A OUT_B 32 OUT_C 48` for three. It preserves tensor payloads and requires new output paths. The configuration records the original weight and shard locations; the review artifact distributes source and evidence, not weight tensors.

The collection entry point is `infra_code/experiments/run_qwen38_factorial.py`. Its `--print-plan` option displays the registered order and serving settings without loading models:

```
python infra_code/experiments/run_qwen38_factorial.py \
  --profile batched128_parallel2 --print-plan
```

On the registered, idle devices, omitting `--print-plan` starts a new full collection with fresh output prefixes. The collector uses the existing responsibility runner, preserves failed behavioral responses, audits every completed run, and stops on a driver or measurement fault. Re-analysis of the retained records requires neither these devices nor new model invocations.

H Information and measurement foundations

Information state and function placement shape the boundary at which requirements are tested. Partially observed control motivates decision-facing information state, and MDP theory supplies policy and decision epochs (Åström, 1965; Kaelbling et al., 1998; Puterman, 1994). State abstraction

studies what decisions survive a change of representation; sequential-team and sufficient-information work identify information that can be omitted while preserving optimality (Li et al., 2006; Mahajan & Tatikonda, 2015; Tavafoghi et al., 2022). Within each DUT allocation comparison, the executable policy remains fixed. End-to-end reasoning motivates function placement (Saltzer et al., 1984); this supplies context for testing what the component must fulfill when surrounding software changes its promised support.

Observed behavior also needs a basis for interpretation. Construct-validity theory links evidence to score interpretation and use (Cronbach & Meehl, 1955; Messick, 1995; Kane, 2013; Borsboom et al., 2004); language-model evaluation critiques examine the corresponding risks in benchmark design and behavioral assessment (Bowman & Dahl, 2021; Reuel et al., 2024; Bean et al., 2025; Kostić et al., 2026; Antoine, 2026). Operational qualification requires repeatability and intended-use evidence beyond recorded-response conformance. Criterion-referenced measurement evaluates a specified behavioral domain rather than relative standing (Glaser, 1963; Popham & Husek, 1969; Hambleton & Novick, 1973), explaining why retained requirements can remain useful even when comparative scores saturate.

Behavioral and agent-system lineages. Metamorphic testing and contrastive examples develop transformations and local behavioral hypotheses (Segura et al., 2016; Gardner et al., 2020). Neural coverage, logic-grounded and counterfactual checks offer executable probes (Pei et al., 2017; Zhou et al., 2026; Roewer-Després et al., 2025); adaptive and generated suites extend them across candidate behaviors (Ribeiro & Lundberg, 2022; Zhao et al., 2024; Efrat et al., 2023). Long-horizon task environments expose prerequisite and phase behavior (Gonzalez-Pumariega et al., 2025; Froger et al., 2026). Reasoning–action loops and agent-computer interfaces expose the execution boundary (Yao et al., 2023; Yang et al., 2024); externalized progress, causal structure and programmatic control offer different placements of work around that boundary (Jiang et al., 2026; Wu et al., 2026; Qi et al., 2026).