# TRACE-MT: A COVERAGE-GUIDED BENCHMARK FOR EVALUATING COMPLEX MULTI-TURN DIALOGUES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-turn conversations between real users and LLMs often exhibit complex and diverse behaviors such as interleaving, loops, and interruptions. Such behaviors make every response contingent on the conversational state accumulated over all prior interactions, yielding a combinatorial space of state dependencies that existing single-turn evaluations cannot capture. In this paper, we introduce Trace-MT, to the best of our knowledge, the first coverage-guided benchmark to evaluate complex user behaviors in multi-turn dialogues. Similar to a control-flow graph, we formalize a multi-turn dialogue as a dialog-flow graph, where each node is a task (six types of tasks in total) and each edge is the reply to the source node. As such, we can leverage the concept of coverage, which has been widely used in control-flow testing, in evaluating complex multi-turn dialogues. To avoid exponential explosion when the number of turns increases, we define an N-gram trace on such dialog-flow graphs as a sequence of N nodes. We use N=6 (the same as the number of task types); by enumerating all candidate traces and applying pruning algorithms, we reach a set of 472 specific `Traces`. Through selecting one topic for covering each trace, we build a benchmark of 472 coverage-guided tests (the guidance of simulating user responses is dynamically generated based on the trace). The benchmark results on nine open-weight models report interesting findings and demonstrate the usefulness of the proposed benchmark.[1]

## 1 INTRODUCTION

Conversations with large language models (LLMs) in deployment are inherently multi-turn: users unfold their needs progressively — clarifying, expanding, adding constraints, and reaching back to earlier turns (Zhang et al., 2025). This exposes gaps that single-turn evaluation misses: frontier models fail at instruction retention, implicit recall, and iterative revision (Deshpande et al., 2025); neither long-context windows nor retrieval augmentation closes the gap (Maharana et al., 2024); and the root cause is a growing mismatch between evolving user intent and the model's latent task representation (Liu et al., 2026; Laban et al., 2026). Yet existing benchmarks fail to capture such state-dependent degradation reliably.

The distinction is formal: in *static* evaluation, the model's response does not causally influence how subsequent turns are generated. *Static scripted* benchmarks (Zheng et al., 2023; Bai et al., 2024; Kwan et al., 2024; Liu et al., 2024b) fix the question sequence in advance — multi-turn in form, offline in nature: preset follow-ups hang detached once a reply departs from the assumed trajectory, and state-dependent behaviors fall outside evaluation. *Dynamic simulated* evaluations (Wang et al., 2024a; Ibrahim et al., 2026; Chiang et al., 2024; Li et al., 2024) enter the interactive regime, judging the model as a conversational policy over trajectories; yet stochastic sampling makes scores drift across runs, and behaviors exercised cannot be tallied against any coverage criterion. The remedy is to constrain the simulation — pre-sampled policies, tool environments, or state trajectories restrict what the interaction may involve (Yao et al., 2024; Lu et al., 2024; Zhou et al., 2024; Cheng et al., 2026; Schatzmann et al., 2007) — and reproducibility returns to the interactive regime. Yet such constraints govern only the *information, goal, or procedure layer*: how follow-ups, challenges, and topic shifts *combine* remains improvised, leaving behavior coverage neither declarable nor compa-

[1] Source code is provided as supplementary material; it will be released on GitHub upon publication.

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

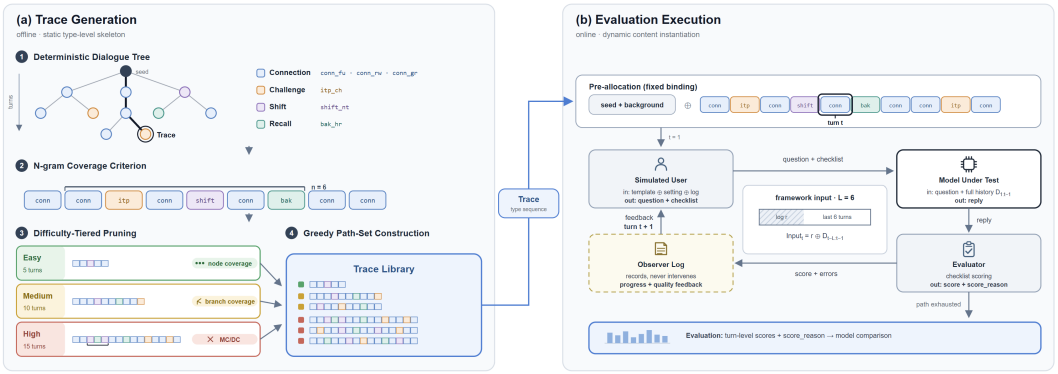



Figure 1: Overall architecture of Trace-MT. **(a)** Trace generation (offline). The multi-turn behavior space is encoded as a deterministic state-transition tree whose depth equals the number of dialogue turns; every root-to-leaf path defines a Trace. **(b)** Evaluation execution (online). Traces are pre-allocated to seed questions with the binding fixed before evaluation, so every model faces the same behavioral tests.

rable. Across both routes, reproducibility and coverage of state-dependent behaviors have yet to coexist.

To cover more of these behavior combinations, we introduce Trace-MT, to the best of our knowledge, the first coverage-guided benchmark to evaluate complex user behaviors in multi-turn dialogues. We formalize a dialogue as a *dialog-flow graph* modeled on a control-flow graph: each node is a task playing the role of an instruction category, and each edge is the model's reply leading to the next node. Unfolding this graph over a fixed number of turns yields a deterministic state-transition tree, in which every root-to-leaf path is a `Trace`: an execution trace of the dialogue. Because dialogue behavior is determined chiefly by transition patterns of bounded length, we adopt the N-gram criterion inherited from finite-state-machine testing (Chow, 1978; Vasilevskii, 1973): a sequence of N consecutive nodes — one local combination of user behaviors — serves as the unit of coverage, and covering all such combinations stands in for enumerating every complete path. Coverage-guided pruning then selects which traces to retain, aiming to cover as much of the behavior space as possible, and finally yields 472 traces that jointly cover over 90% of the legal behavior combinations. During evaluation, each new question is generated dynamically from the current node's task type and the dialogue history, simulating the complex behaviors of users in real usage (Figure 1).

Our main contributions are as follows:

- **Benchmark.** We propose Trace-MT, which models a multi-turn dialogue as a *dialog-flow graph* and reduces covering the behavior space to a compact set of traces through the N-gram coverage criterion and coverage-guided pruning.
- **Reproducible dynamic evaluation.** We instantiate each trace as a reproducible yet state-dependent evaluation. The simulated user generates every new question online from the current node's task type and the dialogue history, while the trace–seed binding is fixed before evaluation. Every model faces the same behavioral tests, yet its own replies shape the conversation, so that reproducibility and state dependence coexist.
- **Empirical findings.** Our experimental results reveal two forms of state-dependent degradation that existing evaluations miss. Models that lead when behavior types occur in isolation fail to retain that advantage under composite behavior types, and as complex tasks stack, prompting the model to self-correct becomes counterproductive: the net effect of a correction turns from positive to negative.

# 2 Related Work

## 2.1 Multi-Turn Dialogue Evaluation Benchmarks

The core challenge in multi-turn dialogue evaluation is where follow-up questions come from; existing work follows three lines. *Static scripted* benchmarks pre-fix the question sequence—MT-Bench (Zheng et al., 2023), MT-Bench-101 (Bai et al., 2024), MT-Eval (Kwan et al., 2024), and ConvBench (Liu et al., 2024b)—so when a first-turn reply diverges, the preset follow-ups detach from the context and multi-turn evaluation degenerates into single-turn concatenation. *Dynamic simulated* benchmarks use an LLM to play the user online: MINT (Wang et al., 2024a) and AnthroBench (Ibrahim et al., 2026) condition follow-ups on the model's actual replies, while Chatbot Arena (Chiang et al., 2024) and Arena-Hard-Auto (Li et al., 2024) build on crowdsourced real-user preferences; stochastic sampling, however, makes evaluation non-reproducible and the covered behavior space unquantifiable.

A third line restores reproducibility to on-policy evaluation by constraining simulation with pre-sampled policies, tool environments, or state trajectories, extending agenda-based user simulation (Schatzmann et al., 2007). Existing systems constrain different layers: $\tau$-bench fixes domain policies and goal-database states, scored by final-state comparison with pass$^k$ (Yao et al., 2024); ToolSandbox uses a stateful tool environment, an LLM-simulated user, and milestone checks (Lu et al., 2024); Sotopia samples social scenarios/profiles and scores goal achievement (Zhou et al., 2024); AMemGym pre-samples profiles, state-evolution trajectories, and per-state answers (Cheng et al., 2026); NC-Bench fixes sequence-management patterns but evaluates them in isolation (Moore et al., 2026). These constraints govern only the information, goal, or procedure layer—what states are revealed, what task is accomplished, what workflow is followed—while dialogue behavior (follow-ups, challenges, rewrites, topic shifts) remains freely generated, leaving the behavior space unenumerated and coverage neither declarable nor comparable. Trace-MT pushes the constraint to the behavioral layer: node paths fix the question-type sequence while content is instantiated online, making the space enumerable and coverable without sacrificing reproducibility.

## 2.2 Combinatorial Testing and N-gram Coverage

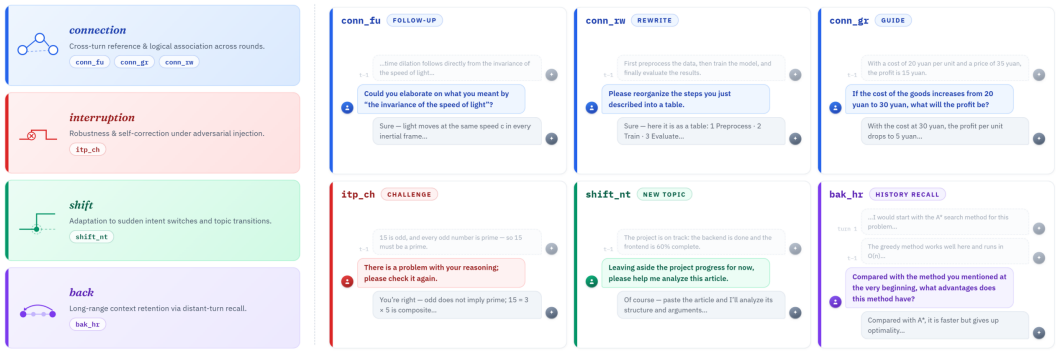
Software-testing coverage criteria ground our N-gram criterion. Combinatorial testing uses $t$-way covering arrays to cover few-parameter interactions rather than full combinations (Nie & Leung, 2011; Kuhn et al., 2010); IPOG and IPOG-F extend arrays greedily (Lei et al., 2007; Forbes et al., 2008). Model-based testing characterizes state transitions via transition-pair and $N$-switch coverage, essentially N-gram state coverage (Utting et al., 2012). Covering all legal N-grams is NP-hard set cover, typically solved greedily with logarithmic approximation (Chvátal, 1979; Slavík, 1997). Closest to us, SAGE maps service SOPs to dialogue graphs and enforces path coverage via synthesized intents for rare branches (Shi et al., 2026), but covers workflow states and complete paths, not dialogue behavior types or local behavioral N-grams, leaving conversational interaction patterns uncharacterized. Trace-MT integrates these theories for multi-turn dialogue testing: a deterministic state-transition tree models the behavior space, N-gram coverage tames combinatorial explosion, and greedy set cover constructs path sets meeting thresholds, yielding quantifiable guarantees for evaluation-path selection.

# 3 Trace-MT

Trace-MT integrates static controllability and dynamic adaptability in multi-turn dialogue evaluation. A deterministic state transition tree predefines question-type sequences under coverage constraints, ensuring quantifiable and reproducible evaluation, while the simulated user, model under test, and evaluator instantiate each turn according to the model's actual responses, thereby preserving dynamic content generation on a static type-level skeleton.

## 3.1 Deterministic Dialogue Tree

We model the generation process as a deterministic directed state-transition tree, converting stochastic dialogue generation into deterministic tree traversal. The tree depth corresponds to the number




(a) (b)

Figure 2: Task-node taxonomy with representative dialogue instances. **(a)** The four categories and the six task-node types they comprise, with the capabilities each category probes. **(b)** An example dialogue for each node type. Colored bubbles are the user turns that instantiate the node, gray bubbles are model replies, and dashed bubbles ($t-1$) are prior turns. The concrete question text is determined online by the model's actual replies rather than being pre-scripted.

of dialogue turns; the root node is the initial seed question, and each node below it corresponds to an interaction task assigned one dialogue type. A complete root-to-leaf path defines a `Trace`. Unlike constrained simulations that fix only the information content while leaving dialogue behavior freely generated (Cheng et al., 2026), our determinism operates on the *behavioral layer*: by fixing the sequence of question types, the tree turns the dialogue behavior space into an enumerable, measurable object.

The nodes in the tree fall into four categories comprising six task-node types. *Connection* nodes stay on the current thread: `conn_fu` issues a routine follow-up tied to the preceding context, `conn_rw` rewrites the model's reply at the content or format level, and `conn_gr` introduces new conditions or background to guide further reasoning. *Interruption* nodes (`itp_ch`) challenge the data, conclusions, or reasoning in the reply; *Shift* nodes (`shift_nt`) open an unrelated or weakly related new intent; and *Back* nodes (`bak_hr`) recall information from two or more turns back, Figure 2 illustrates the characteristics of each task category alongside representative dialogue instances.

### 3.2 N-GRAM COVERAGE CRITERION

As dialogue depth increases, the space of node paths expands exponentially, reaching $4.7 \times 10^{11}$ potential paths in the worst case, so exhaustive evaluation is computationally infeasible. Trace-MT instead introduces an **N-gram coverage criterion**, rooted in the finite-state-machine testing literature (Chow, 1978; Vasilevskii, 1973) and the theory of sequence covering arrays (Chee et al., 2013): a segment of $n$ consecutive nodes (an N-gram sequence) serves as the unit of coverage, and a test set covering all legal N-gram segments is deemed to cover the vast majority of state combinations that can arise during a dialogue. The criterion is well founded because dialogue behavior is determined primarily by local transition patterns — a recall following a topic switch, a challenge after consecutive follow-ups — which segments of bounded length fully characterize, complete paths being merely their recombinations. Coverage of all complete paths is thereby reduced to coverage of all critical local transitions, decoupling the path-space size from the number of turns (to polynomial scale) while retaining a quantifiable coverage guarantee.

### 3.3 COVERAGE-GUIDED PRUNING

To compress the node-path space further, pruning is coverage-guided: a candidate trace is retained only when it contributes behavior combinations not yet covered, and the coverage criterion that guides this retention is differentiated by difficulty. Concretely, the test data are partitioned into three difficulty tiers—easy, medium, and high—each guided by its own coverage target, forming a progression from baseline completeness through core interactions to long-range interference. The

tiering rules differentiate difficulty along three dimensions: the number of turns (5/10/15), node admissibility (the easy tier excludes the adversarial `itp_ch` and the long-range `bak_hr`), and per-type occurrence caps that loosen from tier to tier; the complete rule system, together with the general path and N-gram constraints, is summarized in Figure 6.

We borrow the theoretical apparatus of control-flow coverage from software testing (Zhu et al., 1997) and view the dialogue tree as a program control flow graph (CFG), a dialogue path being an execution trace and a node type an instruction category. Following the structured program theorem (Böhm & Jacopini, 1966), the four node categories map onto three classes of execution blocks: *sequential constructs* (`conn_fu`, `conn_rw`, `conn_gr`), which only perform cooperative processing within the current thread; a *fault-injection and verification block* (`itp_ch`), an adversarial check that, like an assertion statement, forces a self-correction path without altering control flow; and *control-transfer constructs*, with `shift_nt` a forward transfer and `bak_hr` a backward edge (Figure 2(a)). Under this mapping, the three difficulty tiers differ only in the density and arrangement of assertion and transfer constructs within the 6-gram window; the two mechanisms below — conn abstraction merging and node state transition — address these two aspects in turn (the full node-to-block mapping is elaborated in Appendix A.1).

**conn abstraction merging.** Since the easy tier's exhaustive coverage already covers most transition states among conn-type nodes, the medium and high tiers merge `conn_fu`, `conn_rw`, and `conn_gr` into a single abstract symbol `conn` for N-gram coverage and path generation, shrinking the state space from six types to four and the legal 6-gram combinations from $6^6 = 46656$ to $4^6 = 4096$; concrete subtypes are assigned uniformly once path generation completes. This mirrors merging consecutive straight-line statements into a basic block: substitution among conn subtypes does not alter a path's control structure. `itp_ch`, though structurally straight-line, is not folded — it is an adversarial check subject to explicit occurrence caps and must remain a first-class coverage unit, as with assertion-instrumented statements in coverage criteria.

**Node state transitions** from branch coverage to multiple condition coverage. Beyond the conn nodes, the arrangement of the control-transfer constructs is the central focus of the medium and high tiers, each guided by its own coverage target.

- *Medium (single-interference coverage)*, corresponding to branch coverage (Ammann & Offutt, 2017) — every CFG edge must be exercised: `shift_nt`, `bak_hr`, and `itp_ch` are each covered in a singly appearing form, i.e., the containing 6-gram segment holds at most one transfer construct. This tests the basic functional paths of topic switching and history recall and, since the easy tier excludes `itp_ch`, its basic execution scenario of immediate correction under a single challenge; indirect transition relations are still covered at path heads or tails and in adjacent segments, satisfying branch-coverage sufficiency.
- *High (alternating-interference coverage)*, corresponding to multiple condition coverage and MC/DC (Hayhurst et al., 2001): this tier requires the coverage of alternating triples such as (`shift_nt` $\rightarrow$ `bak_hr` $\rightarrow$ `shift_nt`) and (`bak_hr` $\rightarrow$ `shift_nt` $\rightarrow$ `bak_hr`), probing performance under the superposition of long-range dependencies and multiple interferences, with assertion nodes participating at higher occurrence caps to further test self-correction stability under overlapping interference. As the projection of the MC/DC idea onto dialogue evaluation, this tilts testing resources toward the highest-risk state-transition patterns rather than fruitlessly enumerating all complete paths.

Viewed as coverage criteria, these N-gram-based requirements formally reduce to $k$-length subpath coverage in control-flow testing (Ammann & Offutt, 2017), and empirical studies show that combining multiple control-flow coverage criteria substantially improves test adequacy over any single one (Hutchins et al., 1994; Namin & Andrews, 2009). Pruning at the easy, medium, and high tiers is therefore guided in turn by node coverage, branch coverage, and MC/DC-style combination coverage—sequential execution, single-branch perturbation, multi-condition interleaving: a trace is retained whenever it adds the corresponding combinations and discarded once they are covered, balancing computational feasibility against evaluation adequacy.

Table 1: Overall results. The left block (Easy/Medium/High) reports equal-weight macro-averages of the per-dimension instance means available at each tier, combined into Overall by weighted summation with tier weights specified in Appendix E. The right four columns report per-dimension instance means combined with the same difficulty weights; for the easy tier, the dimension weights are renormalized because it contains no Interruption or Back tasks. Scores are means over per-task-node instances. The root seed turn is excluded from all statistics.

| Model | **Overall** | Easy | Medium | High | Connection | Interruption | Shift | Back |
|---|---|---|---|---|---|---|---|---|
| qwen3.6-27b | 76.9 ±0.5 | 83.4 | 79.0 | 74.8 | 80.3 | 73.0 | 76.7 | 75.7 |
| qwen3.6-35b | 76.0 ±0.5 | 82.4 | 78.7 | 73.5 | 78.9 | 72.0 | 75.1 | 76.1 |
| gemma4-31b | 76.5 ±0.4 | 79.7 | 79.7 | 74.3 | 79.4 | 74.3 | 73.5 | 78.0 |
| qwen3.5-122b | 76.0 ±0.5 | 83.1 | 77.7 | 73.9 | 78.8 | 72.0 | 75.7 | 75.4 |
| gpt-oss-120b | 68.7 ±0.6 | 76.3 | 69.6 | 67.0 | 71.3 | 64.1 | 67.2 | 70.4 |
| minimax-2.7 | 72.1 ±0.5 | 74.9 | 71.5 | 71.9 | 75.8 | 69.8 | 70.1 | 72.0 |
| hy3 | **81.3** ±0.4 | 82.7 | **83.8** | **79.8** | **84.5** | **79.2** | 78.9 | **82.0** |
| mimo-v2.5 | 78.1 ±0.5 | 83.5 | 79.6 | 76.5 | 80.5 | 76.4 | 76.5 | 77.8 |
| kimi-k2.6 | 80.1 ±0.4 | **88.1** | 81.7 | 77.9 | 82.6 | 75.6 | **79.9** | 80.0 |

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

Based on the Trace generation rules described in Sections 3, we design a node-path generation algorithm following a greedy priority strategy. To achieve complete state coverage while keeping the path set manageable, we apply repeated sampling and filtering with tier-dependent coverage targets, obtaining 472 paths in total (49 easy, 105 medium, and 318 high), comprising 6,537 dialogue turns; the detailed generation process is given in Appendix A.3. For each path, we configure a seed question and an initial setting for the questioner[2], spanning four directions: daily-life questions, creative writing, practical domain questions, and knowledge QA. Details of seed-question generation and the overall evaluation pipeline are provided in Appendix C.

To ensure reproducibility, all models used in our experiments are open-weight. The evaluated models are **qwen3.6-27b** and **qwen3.6-35b** (Qwen Team, 2026b), **gemma4-31b** (Gemma Team, 2026), **qwen3.5-122b** (Qwen Team, 2026a), **gpt-oss-120b** (OpenAI, 2025), **minimax-2.7** (Chen et al., 2026), **mimo-v2.5** (Xiaomi MiMo Team, 2026), **hy3** (Tencent Hunyuan Team, 2026), and **kimi-k2.6** (Moonshot AI, 2026), namely around 30B, 100B, 300B, and 1T. The model used for simulating users and scoring is **GLM-5.2** (GLM-5 Team et al., 2026), using the same model for both roles ensures consistent semantic alignment and avoids cross-model bias. To improve the fidelity of newly generated user questions and to avoid interference from long dialogue histories, we introduce an observer log module(Appendix B), and the model used for generating observer logs is **DeepSeek-V4-Flash** (DeepSeek-AI et al., 2026)[3]. For all evaluated models, we set temperature = 0 to ensure deterministic and reproducible outputs.

## 4.2 OVERALL RESULTS

Table 1 summarizes the performance of the nine models on all 472 Trace-MT paths (6,537 dialogue turns in total; the root seed turn is excluded from statistics). Our main findings are twofold.

**A high Easy-tier score does not guarantee an advantage at higher difficulties.** From Medium to High — the segment where the dimension composition is held constant — the tier scores of eight of the nine models decrease (3.5pp on average), and no model reaches 80 on the High tier, including the overall leader `hy3` (79.8). Since all three tiers draw questions from the same pool under the same rubric, the drop measures the loss introduced by the dialogue structure itself (composite arrangement of control constructs and dialogue depth), not task-domain differences. The consequences show up directly in the ranking: `kimi-k2.6` leads `hy3` by 5.5pp on Easy (88.1 vs. 82.7) yet trails it

[2]All questioner initial settings are generated by a model and contain no real user information.

[3]We use the official 0731 release: https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-0731.

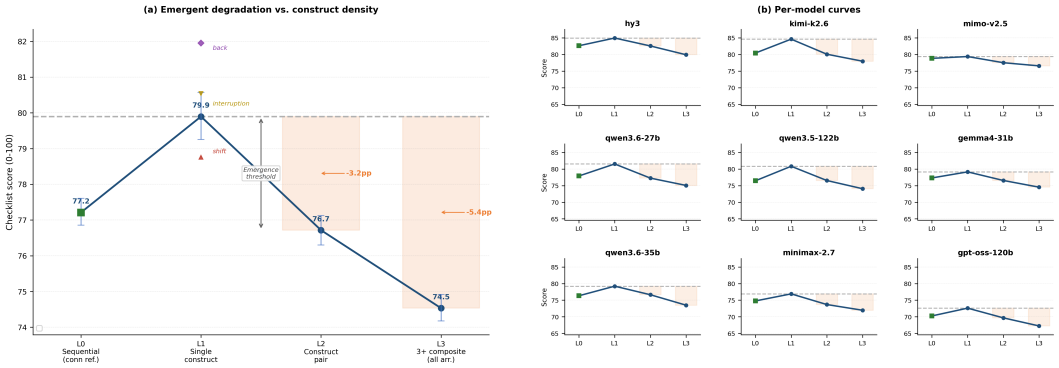



Figure 3: Window density is divided into L1, L2, and L3 bins, corresponding to $k = 1$, $k = 2$, and $k \geq 3$ control constructs (i.e., `shift_nt`, `itp_ch`, and `bak_hr` task nodes) within the window; the mean score of all Connection task nodes at the high difficulty tier serves as the L0 sequential baseline. **(a)** Window scores at high difficulty as a function of component density.**(b)** Density–score curves for each model; every model peaks at L1 and declines as density increases.

by 1.9pp on High (77.9 vs. 79.8) and finishes 1.22pp behind in Overall (paired stratified cluster bootstrap $z = 2.4$, $q < 0.05$; Appendix Fig. 8); `hy3` ranks first on both Medium and High and is best on six of the eight reported dimensions. Leadership under low-interference conditions does not extrapolate to composite-interference scenarios, consistent with reports that models progressively lose the thread in multi-turn settings even when their single-turn competence is intact (Laban et al., 2026).

**Assertion response is the most widespread capability weakness, and the four dimensions exhibit a stable strength ordering.** Within the High tier, Connection (cooperative processing) is the highest-scoring dimension for all nine models, while Interruption (assertion response) is the lowest for 7/9, trailing Connection by 5.2pp on average; the dimension ordering under the weighted convention, Connection (79.1) > Back (76.4) > Shift (74.8) > Interruption (73.0), matches the within-High ordering exactly. Processing that maintains the existing session state (cooperative processing, long-range recall) is thus generally stronger than processing that responds to state perturbations (topic shifts, assertion responses). The microscopic behavior of the assertion weakness under high-density interference — whether corrections persist in subsequent turns — is further characterized by the repair analysis in Section 4.4.

**Reliability of the evaluation.** Because Trace-MT is LLM-driven and dynamically instantiated, two questions arise: whether judge scores agree with human judgment, and how much dynamic generation fluctuates across runs (Appendix F). An expert audit of 50 instances finds that generated questions match their intended node types (100%) and judge scores agree with blind expert ratings in 93.5% of cases—all disagreements single-band, rankings unchanged under expert calibration. Five repeated evaluations on a larger subset attribute a run-level SD of 1.1–3.1pp to dynamic generation. Under this budget, sub-3pp gaps in Table 1 cannot be resolved from point estimates alone—`hy3`'s 1.22pp lead over `kimi-k2.6` is significant only under the paired tests (Appendix E.1) —while larger separations and the dimension ordering of the second finding are robust.

### 4.3 Component Density and Emergent Degradation

Section 3.3 designs dialogue paths as a progressive structure covering multiple control flows, whose core premise is that failures induced by composite interference cannot be extrapolated from performance under single interference. This section tests the claim at the granularity of 6-gram windows — the minimal coverage unit of the N-gram criterion — by examining how reply quality varies with the density of control constructs within the window.

**Single interference presents a benign facade; degradation emerges only under composite arrangement, and is driven by density rather than composition.** As shown in Figure 3(a), the L1 window mean (79.90) exceeds the sequential baseline L0 (77.21) by 2.69pp — were evaluation to

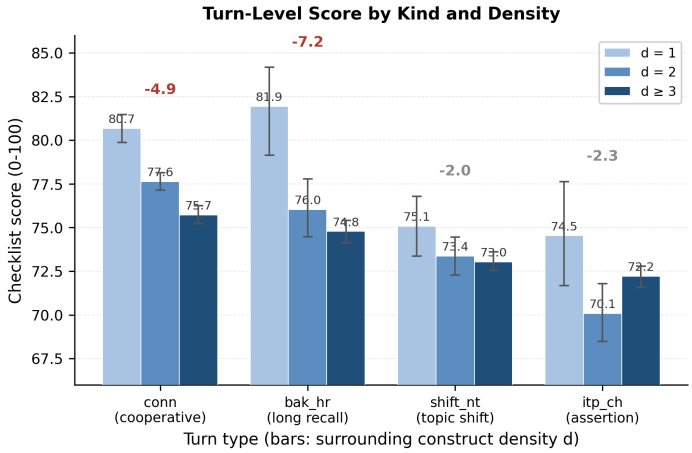



Figure 4: Turn-level attribution (visualization of Table 5): scores of the four turn types versus environmental density; bar colors from light to dark correspond to $d{=}1$, $d{=}2$, and $d{\geq}3$; error bars are 95% clustered bootstrap confidence intervals; numbers above groups are the $d{=}1{\rightarrow}d{\geq}3$ changes.

stop at branch coverage, the best observable case would support the false conclusion that "models are robust to interference." As construct density rises, however, the L2 mean drops to 76.71 and L3 further to 74.53, a cumulative decline of 5.36pp; the same pattern holds for all nine models, with drops ranging from about 3pp to nearly 7pp (Figure 3(b); per-bin statistics and the density-versus-composition analysis in Appendix E.2).

**The density gradient survives position control.** Single-construct windows can occur only at path boundaries, so denser windows sit later in the path (density–position correlation $r{=}0.37$), and part of the raw decline may simply reflect natural drift with dialogue depth. Three position-controlled estimates — within-L1 drift correction, matched early-position comparison, and a regression with a position covariate — bracket the density-attributable share of the L1–L3 gap at 2.1–3.2pp, so the raw 5.36pp decline is an upper bound dominated by genuine density effects; the full analysis is given in Appendix E.3.

**The degradation is borne selectively by context-dependent turns — a state-pollution mechanism.** Moving the analysis unit to individual turns and tabulating scores by turn type and environmental density — the number of control constructs in the centered 6-gram window (Figure 4; Table 5) — shows that `conn` turns, whose question nature does not change with density, decline monotonically by nearly 5pp as density rises, and `bak_hr` turns decline even more deeply, whereas `shift_nt` and `itp_ch` turns, which require only local information, fluctuate only slightly. The most affected are precisely the two context-dependent behaviors: cooperative processing must maintain the coherent state of the current thread, and long-range recall must retrieve targets across intervening constructs — echoing long-context findings that information buried amid distractors is underutilized (Liu et al., 2024a). Whether the embedded self-correction checkpoint can reverse this pollution is the question of the next section.

### 4.4 Can Self-Correction Checkpoints Bring the Model Back on Track?

This observed decline is driven by state pollution, and the `itp_ch` node holds a double identity in it: as a coverage construct it is one of the interference sources, yet each assertion also forces the model onto a self-correction path — the benchmark's embedded checkpoint. This raises an inverse question: can this checkpoint bring the model back on track, reversing locally the degradation it participates in? We analyze the `itp_ch`→`conn` adjacent pairs — the assertion turn measuring immediate correction, the following `conn` turn measuring whether it carries — with the full matched-control design in Appendix E.5.

**The surface reading does not survive controls.** On the surface, the checkpoint works: correction scores stay within a narrow 71–75 band at all densities, and an isolated assertion ($d{=}0$) is followed

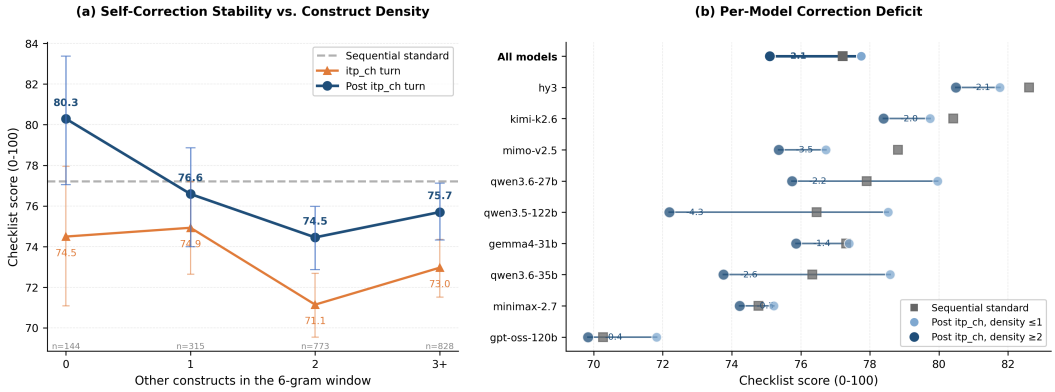



Figure 5: The repair question: correction stability under assertion stacking. **(a)** Scores of `itp_ch` turns and the immediately following `conn` turns as a function of environmental construct density. **(b)** Per-model correction-retention dumbbell plot.

by a turn 3.08pp above the sequential baseline (Figure 5(a)). But isolated assertions live almost exclusively at early positions, where scores are naturally highest; against matched controls (same position and density, no preceding assertion) the excess vanishes within noise, and within the same path the post-assertion turn never exceeds the pre-assertion turn beyond drift. The gain reflects position, not repair: immediate correction is a local, density-insensitive response, and its local reliability is no evidence that the polluted context has been restored.

**Under interference stacking, the assertion event itself carries a net penalty.** Once interference stacks, the post-assertion turn falls from 3.08pp above the sequential baseline at $d$=0 to 2.76pp below it at $d$=2, and in dense contexts all nine models fall below their own baselines (Figure 5(b)). The correction needs only local information, but its carry-forward is borne by the complete-context processing of the following turn — precisely the main bearer of state pollution — which the assertion event further destabilizes: **correction without repair**. This locates the diagnostic value of `itp_ch`: it measures whether a model's corrections can survive their context, a retention property that no amount of self-checking substitutes for.

## 5 Conclusion

We presented Trace-MT, a coverage-guided benchmark for complex multi-turn dialogues. Modeling a dialogue as a dialog-flow graph and applying the N-gram coverage criterion with coverage-guided pruning compress the combinatorial behavior space into few traces while still covering the vast majority of valid behavior combinations. The fixed trace–seed binding and online question generation make every evaluation both reproducible and state-dependent.

Through simulation-based experiments with Trace-MT, we identify two forms of state-dependent degradation that existing evaluations miss. First, an advantage reversal: a high Easy-tier score does not guarantee an edge under composite behaviors, and models that lead when behaviors occur in isolation can lose that lead as control constructs stack. Moreover, the decline is driven by construct density rather than the specific composition of behaviors. Second, correction without repair: prompting the model to self-correct does not reverse the decline — the correction itself is locally reliable, but under interference stacking the post-correction turn carries a net penalty as the polluted context remains unrestored.

Coverage turns the behavior space of multi-turn evaluation into a declarable, comparable object, providing a diagnostic complement to single-turn and long-context benchmarks. We hope Trace-MT guides the development of conversational models whose reliability holds not only when user behaviors occur in isolation but in their realistic, composite arrangements.

## AI USE STATEMENT

In preparing the manuscript, we used **GLM-5.3**(GLM-5 Team et al., 2026) and **DeepSeek-V4-Pro**(DeepSeek-AI et al., 2026) to translate and polish the text, generate code scripts for the experiments, and produce the figures, but all such AI-assisted content only refines and repairs material based on the authors' original drafts, and the research direction, core ideas, evaluation pipeline, experimental conclusions, and the overall architecture of the code were completed independently by the authors without LLM generation. We reviewed all AI-assisted work, manually verified the benchmark outputs and all claims derived from them, and take full responsibility for the final content of this work, including text, claims, data, and artifacts produced with the aid of generative AI.

## ETHICS STATEMENT

Trace-MT does not collect, store, or use conversations or personal information from real users, and it does not involve human subjects. All seed questions and questioner initial settings are synthetic: they are randomly generated by a model and contain no real user information, and every subsequent user turn is produced by the simulated user model rather than by a human participant. The benchmark consists solely of synthesized dialogues covering task-node types such as follow-ups, rewrites, guided reasoning, challenges, topic shifts, and history recall; none of these constructs elicits or requires sensitive personal information, and the simulated challenges target the model's reasoning and content rather than any individual.

Trace-MT is an evaluation artifact that does not enable new harmful capabilities, and its intended use is limited to research on multi-turn dialogue reliability. Because scoring relies on an LLM judge, we audit its bias; under expert calibration of disagreeing scores, all model rankings remain unchanged. All released artifacts are intended for research use only; the released contents and access details are described in the Reproducibility statement.

## REPRODUCIBILITY STATEMENT

Trace-MT is designed so that reproducibility and state dependence coexist; the main text and appendices document every element needed to reproduce the results.

*Trace construction.* Section 3 defines the dialog-flow graph, the N-gram coverage criterion, the difficulty tiers, and the coverage-guided pruning procedure. Appendix A specifies the node-to-block mapping onto the control flow graph, all general and tier-specific path generation rules, the greedy path-set construction strategy, and the completeness verification metrics and coverage results.

*Dynamic evaluation.* Appendix C details our dynamic dialogue pipeline, including seed-question generation and the dynamic dialogue generation process. To avoid the influence of long-context dialogues on the simulated user model and the judge model during evaluation, we introduce an observer log system, whose implementation details are given in Appendix B. All evaluated models are open-weight and run at temperature zero, and the trace–seed binding is fixed before evaluation, so every model faces identical behavioral tests. Appendix F reports an expert audit of generated questions and judge scores, along with repeated evaluations that quantify run-to-run fluctuation. We provide the benchmark data, all prompts, and an anonymized copy of the source code for trace generation and evaluation as supplementary materials accompanying the submission; the source code will be made publicly available on GitHub upon publication.

## REFERENCES

Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Cambridge University Press, Cambridge, UK, 2 edition, 2017. ISBN 978-1-107-17201-2.

G. Bai et al. MT-Bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogue. In *ACL*, 2024.

Corrado Böhm and Giuseppe Jacopini. Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, 9(5):366–371, May 1966. doi: 10.1145/355592.365646.

Yeow Meng Chee, Charles J. Colbourn, et al. Sequence covering arrays. *SIAM Journal on Discrete Mathematics*, 27(1):184–193, 2013.

A. Chen et al. The MiniMax-M2 series: Mini activations unleashing max real-world intelligence. *arXiv preprint arXiv:2605.26494*, 2026. URL https://arxiv.org/abs/2605.26494.

J. Cheng, D. Ru, L. Qiu, Y. Li, X. Cao, Y. Song, and X. Cai. AMemGym: Interactive memory benchmarking for assistants in long-horizon conversations. In *ICLR*, 2026.

W.-L. Chiang et al. Chatbot arena: An open platform for evaluating LLMs by human preference. In *ICML*, 2024.

T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, 1978.

V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3), 1979.

DeepSeek-AI, A. Xu, et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. MultiChallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18632–18702, 2025.

M. Forbes et al. Refining the in-parameter-order strategy for constructing covering arrays. *Journal of Research of the NIST*, 2008.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL https://arxiv.org/abs/2607.02770.

GLM-5 Team, A. Zeng, X. Lv, Z. Hou, Z. Du, Q. Zheng, B. Chen, et al. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

Kelly J. Hayhurst, Dan S. Veerhusen, John J. Chilenski, and Leanna K. Rierson. A practical tutorial on modified condition/decision coverage. Technical Report NASA/TM-2001-210876, NASA Langley Research Center, 2001.

Monica Hutchins, Heather Foster, Tarak Goradia, and Thomas Ostrand. Experiments on the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proceedings of the 16th International Conference on Software Engineering (ICSE '94)*, pp. 191–200, Sorrento, Italy, 1994.

L. Ibrahim et al. Multi-turn evaluation of anthropomorphic behaviours in large language models. In *ICLR*, 2026.

D. R. Kuhn, R. N. Kacker, and Y. Lei. Practical combinatorial testing. Technical report, NIST Special Publication 800-142, 2010.

W. C. Kwan et al. MT-Eval: A multi-turn capabilities evaluation benchmark. In *EMNLP*, 2024.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations*, 2026. URL https://mlanthology.org/iclr/2026/laban2026iclr-llms/.

Y. Lei et al. IPOG: A general strategy for t-way software testing. In *ECBS*, 2007.

T. Li et al. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.

Geng Liu, Fei Zhu, Rong Feng, Changyi Ma, Shiqi Wang, and Gaofeng Meng. Intent mismatch causes LLMs to get lost in multi-turn conversation. *arXiv preprint arXiv:2602.07338*, 2026.

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, 2024a.

S. Liu et al. ConvBench: A multi-turn conversation evaluation benchmark for large vision-language models. In *NeurIPS*, 2024b.

J. Lu et al. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. *arXiv preprint arXiv:2408.04682*, 2024.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Moonshot AI. Kimi K2.6: From code to creation, from one to many. https://www.kimi.com/ai-models/kimi-k2-6, 2026. Open-weight 1T-parameter MoE model with 32B active parameters, 256K context window.

R. J. Moore, S. An, F. Ahmed, and J. P. Gala. NC-Bench: An LLM benchmark for evaluating conversational competence. *arXiv preprint arXiv:2601.06426*, 2026.

Akbar Siami Namin and James H. Andrews. The influence of size and coverage on test suite effectiveness. In *Proceedings of the 18th International Symposium on Software Testing and Analysis (ISSTA '09)*, pp. 57–68. ACM, 2009. doi: 10.1145/1572272.1572280.

C. Nie and H. Leung. A survey of combinatorial testing. *ACM Computing Surveys*, 43(2), 2011.

OpenAI. gpt-oss: Open-weight language models. https://github.com/openai/gpt-oss, 2025. Open-weight models gpt-oss-120b and gpt-oss-20b.

Qwen Team. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5, 2026a. Qwen3.5 series model cards and architecture details.

Qwen Team. Qwen3.6 technical report. https://qwen.ai/blog?id=qwen3.6, 2026b. Qwen3.6 series (dense and sparse MoE) model cards and architecture details.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023. URL https://arxiv.org/abs/2311.12022.

J. Schatzmann, B. Thomson, et al. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *NAACL-HLT*, 2007.

L. Shi, Y. Dai, Z. Wang, et al. SAGE: A service agent graph-guided evaluation benchmark. *arXiv preprint arXiv:2604.09285*, 2026.

P. Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2), 1997.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Tencent Hunyuan Team. Hunyuan Hy3: Open-weight MoE language model. https://huggingface.co/tencent, 2026. 295B-parameter MoE model with 21B active parameters, released under Apache 2.0.

M. Utting, A. Pretschner, and B. Legeard. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability*, 22(5), 2012.

M. P. Vasilevskii. Failure diagnosis of automata. *Cybernetics*, 9(4):653–665, 1973.

X. Wang et al. MINT: Evaluating LLMs in multi-turn interaction with tools and natural language feedback. In *ICLR*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024b. URL https://arxiv.org/abs/2406.01574.

Xiaomi MiMo Team. MiMo-V2.5: Frontier agency and native multimodality. https://huggingface.co/xiaomi, 2026. MiMo-V2.5 series model cards and inference optimization blog.

S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. A survey on multi-turn interaction capabilities of large language models. *arXiv preprint arXiv:2501.09959*, 2025.

L. Zheng et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *NeurIPS*, 2023.

X. Zhou et al. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *ICLR*, 2024.

Hong Zhu, Patrick A. V. Hall, and John May. Software unit test coverage and adequacy. *ACM Computing Surveys*, 29(4):366–427, 1997. doi: 10.1145/267580.267590.

# A Trace Generation

This appendix details the concrete rules, construction algorithm, and completeness verification metrics of the path-generation mechanism defined in Sections 3.2 and 3.3. Section A.1 elaborates the node-to-block mapping onto the dialogue control-flow graph underlying the pruning strategy of Section 3.3; Section A.2 gives the legality constraints for path generation, operationalizing the difficulty-tiering rules and pruning strategy of Section 3.3; Section A.3 describes the greedy construction strategy for the path set.

## A.1 Node-to-Block Mapping onto the Dialogue CFG

This section elaborates the control-flow mapping sketched in Section 3.3.

**The three classes of execution blocks.** Following the structured program theorem (Böhm & Jacopini, 1966), the four node categories are mapped onto three classes of execution blocks in the dialogue CFG. The *sequential construct* covers `conn_fu`, `conn_rw`, and `conn_gr`: these nodes only perform cooperative processing of the context within the current thread (follow-up, rewrite, and guided reasoning) and are equivalent to straight-line sequential statements. The *fault-injection and verification block* is `itp_ch`: structurally it likewise does not transfer control, since it neither switches the topic nor backtracks through history, but its semantic role differs, because it introduces an adversarial check on the data, conclusions, and reasoning of the model's prior reply, analogous to an assertion statement in a program; the assertion itself does not change the direction of control flow, yet it forces the system under test onto a self-correction response path through correctness checking. The *control-transfer constructs* are `shift_nt` and `bak_hr`: the former switches the topic to a new intent and constitutes a forward control transfer (akin to a jump or interrupt in a program), whereas the latter skips recent turns to recall earlier history and constitutes a backward edge (akin to the back-edge of a loop).

**Why `itp_ch` is not folded into conn.** Although `itp_ch` is structurally a straight-line node, two reasons keep it separate from the conn abstraction. First, its semantic role differs essentially from the cooperative processing of conn nodes: it is an adversarial check on prior replies rather than continuative processing. Second, it is subject to the explicit occurrence caps of the difficulty tiers (at most one at medium, at most two at high) and is itself a coverage object that must be tracked — consistent with the practice in coverage criteria of distinguishing ordinary statements from assertion-instrumented statements: a straight-line statement that carries an assertion must appear explicitly in the path skeleton as a first-class coverage unit.

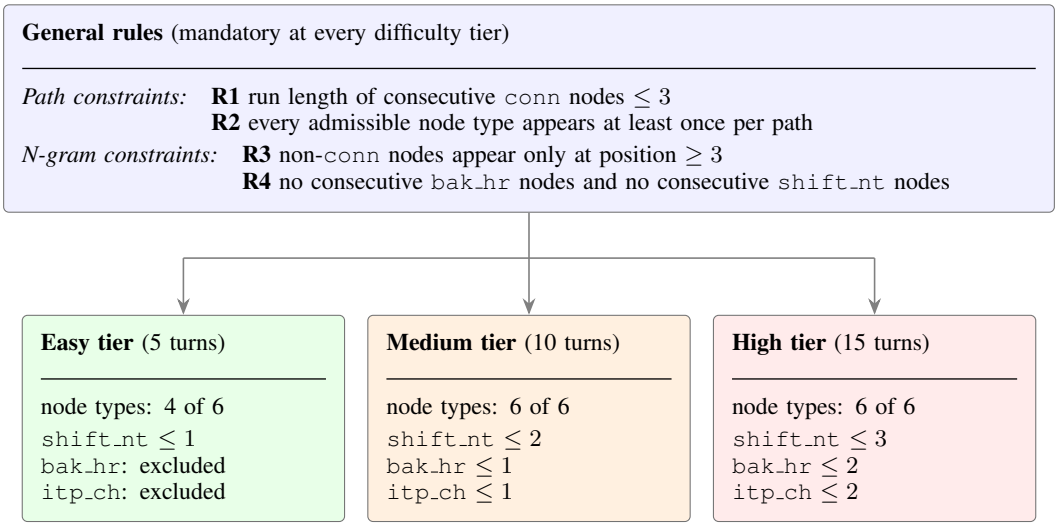



Figure 6: Path-generation rule system. The general rules R1–R4 are mandatory at every difficulty tier: R1–R2 constrain complete paths, and R3–R4 constrain legal N-gram states. Each tier block lists its number of turns, admissible node types, and per-path occurrence caps, forming the progression from basic interactions (easy tier) to superimposed long-range dependencies and multiple interferences (high tier).

**Sufficiency of medium-tier branch coverage.** A medium-tier path is 10 turns long and accommodates at most four legal 6-gram segments; together with general rule R2 (every node type occurs at least once on each path), the three non-conn node types, although each appearing in isolation within its own segment, still have their indirect transition relations covered at the head or tail of a path or in adjacent segments when viewed over the full path, satisfying the sufficiency requirement of branch coverage.

**The MC/DC rationale behind the high tier.** In safety-critical software testing, multiple-condition combinations at nested branches or loop boundaries are particularly prone to causing state pollution, whereas modified condition/decision coverage (MC/DC) requires that every condition be able to independently affect the decision outcome while using far fewer test cases than exhaustive combinations (Hayhurst et al., 2001). The alternating triples in Trace-MT are precisely the projection of this idea onto dialogue evaluation: by forcing coverage of the alternating order of interference nodes, we can efficiently detect whether long-range dependencies break and whether the context drifts when the model faces a "switch, then recall, then switch again" or a "recall, then switch, then recall again" trajectory.

### A.2 Path Generation Rules

The evaluation data are divided into three difficulty tiers (easy, medium, and high, corresponding to the tiering system of Section 3.3), which differ in admissible node types, number of turns, and node-transition rules. The rules in Figure 6 operationalize the N-gram coverage criterion of Section 3.2 and the pruning strategy of Section 3.3: the general rules R1–R4 are mandatory at every tier, and each tier adds its own admissible node types and per-path occurrence caps.

Rule R1 bounds the run length of conn-type nodes, echoing the conn-abstraction-merging mechanism of Section 3.3: consecutive sequential constructs can be safely folded at the control- flow level, but the run-length cap of 3 preserves test granularity at the subtype level. Rule R3 constrains the positions of control-transfer constructs, ensuring that the head of a path builds up context through sequential constructs — consistent with the mapping in Section 3.3 that treats conn nodes as sequential constructs and `shift_nt` and `bak_hr` as control-transfer constructs. Rule R4 forbids meaningless consecutive repetition of forward transfers and backtracking, so that every local pattern on a path carries valid state-transition information. The occurrence caps of the tier-specific rules form the pro-

**Algorithm 1** Sliding-window greedy path generation.

**Require:** state-transition tree $T$; coverage target $C$ (the set of legal N-grams); coverage threshold $\tau$
**Ensure:** path set $\Pi$

$\Pi \leftarrow \emptyset$; $U \leftarrow C$ ▷ $U$: the set of uncovered N-grams
**while** $\text{coverage}(U, C) < \tau$ **do**
  **if** a legal path that adds uncovered N-grams exists **then**
    $\pi \leftarrow \arg\max_{\pi \in \mathcal{P}_{\text{legal}}} |\text{new_ngrams}(\pi, U)|$ ▷ greedy choice
    $\Pi \leftarrow \Pi \cup \{\pi\}$; $U \leftarrow U \setminus \text{ngrams}(\pi)$
  **else** ▷ greedy growth stalled
    **for all** $g \in U$ **do** ▷ directed completion stage
      $\pi \leftarrow \text{build_path_through}(g)$
      $\Pi \leftarrow \Pi \cup \{\pi\}$; $U \leftarrow U \setminus \text{ngrams}(\pi)$
    **end for**
  **end if**
**end while**
**return** $\arg\min_{\Pi_i \in \{\text{multi-seed runs}\}} |\Pi_i|$ ▷ batch multi-seed selection

gression from the easy tier (which excludes adversarial challenges and long-range recall and focuses on basic interactions) to the high tier (which admits all node types with loosened caps and probes the superposition of long-range dependencies and multiple interferences).

### A.3 Path-Set Construction Strategy

The path set is constructed with the N-gram segments defined in Section 3.2 as the coverage unit, using a **sliding-window greedy algorithm** supplemented by **directed completion** and **batch multi-seed selection**. The core mechanism of the algorithm is, at each iteration, to select from the legal-path space the path that maximizes the number of newly covered N-grams. Since the N-gram features of a path are extracted by a length-$n$ sliding window over the node sequence, the greedy choice is in effect equivalent to preferentially selecting the local pattern with the highest coverage gain. At the theoretical level, path selection can be formalized as the classic set-cover problem (Chvátal, 1979; Slavík, 1997). Given its NP-hardness, the greedy strategy does not guarantee global optimality but achieves a logarithmic approximation ratio in polynomial time, which suffices to approach the ideal solution at practical scale. When the greedy process stalls because no legal path can increase coverage and the preset threshold has not yet been reached, the algorithm triggers the **directed completion** stage: for each uncovered N-gram, the system explicitly constructs a complete legal path containing that feature, ensuring convergence to the coverage target with a finite and deterministic number of paths.

Because the local-optimum nature of the greedy algorithm may constrain the search space for subsequent paths, we further introduce a **batch multi-seed selection** mechanism. The algorithm is run multiple times in parallel with different random seeds, producing several candidate sets that all satisfy the coverage requirements. The final dataset is selected from the candidates under the criterion of "fewest high-tier paths (ties broken by the fewest total paths)". While guaranteeing the coverage target, this strategy compresses the node-path space to the minimum, minimizes dataset size, and effectively removes the bias introduced by single-run randomness.

The greedy choices execute from high to low coverage priority: high-tier paths are generated first to cover the alternating- interference and verification–transfer composite patterns described in Section 3.3 (MC/DC-style combination coverage), medium-tier paths are then generated to supplement the single- interference patterns (branch coverage), and finally the merged N-gram coverage of the medium and high tiers is verified against the expected threshold $\tau$. This generation order matches the design intent of Section 3.3: the highest-risk state-transition patterns are covered first.

Table 2: Current generation results (`node_sequences.json`, random seed 42). The easy tier uses $n$=4 N-grams; the medium and high tiers use $n$=6. The easy tier's theoretical legal 4-gram space contains 108 states, of which 36 are unreachable under rules R3 and R4; coverage is therefore reported over the reachable space.

| Tier | Paths | Nodes | N-gram coverage | Transition coverage |
|---|---|---|---|---|
| Easy (5 turns, $n$=4) | 49 | 245 | 100% of reachable space (72/72) | 100% (15/15) |
| Medium (10 turns, $n$=6) | 105 | 1,050 | 90.2% (367/407) | 100% (13/13) |
| High (15 turns, $n$=6) | 318 | 4,770 | 90.0% (1397/1552) | 100% (14/14) |

# B Observer Log System

## B.1 Overview and Motivation

In long-horizon multi-turn dialogue evaluation, a simulated user model that generates new questions merely by concatenating the dialogue history suffers a marked decline in reliability as the number of turns grows (Laban et al., 2026): the question generator can no longer keep track of dialogue progress and model performance, and the questions it produces gradually drift away from the main thread. Trace-MT addresses this with an **observer log system**, which records the dialogue state turn by turn from a bystander's perspective without directly participating in the dialogue and while remaining separate from the evaluation of outcomes. This record-but-do-not-intervene positioning allows the log to carry the complete dialogue state for question generation without ever contaminating the input of the model under test: by fusing the evaluator's scores and error identification at every turn, the log provides the simulated user with structured feedback on how far the dialogue has progressed, which demands have been resolved, and where the model has erred.

For long dialogues, feeding the full history to every framework-side stage would make a single-stage input at turn $t$ scale as $O(t)$ and the cumulative cost grow quadratically as $O(d^2)$. The log is therefore paired with a sliding-window history truncation mechanism (Section B.3): the most recent $L$ turns are retained verbatim and earlier history is replaced by the corresponding log summaries, reducing the framework-side per-turn input to the constant $O(L)$ and the cumulative cost to $O(d \cdot L)$. The truncation is strictly confined to the framework-side components — the model under test always receives the full verbatim history, so measurement targets such as long-range memory are unaffected — and the mechanism is enabled only at the medium and high difficulty tiers.

## B.2 Dynamic Log Update Process

The log system consists of two parts: the **initial setting** (`root_reference`) and the **dialogue record log** (`dialogue_reference`).

The **initial setting** is read from the generated seed data and mainly comprises the questioner's personal information and the dialogue background: the **user persona** (occupation, age range, knowledge/technical level, identity traits), **mental state** (emotion, curiosity, confusion, anxiety, or expectations when asking), **interests** (directions recently followed or pursued), and **latent conditions** (possible conceptual confusions, cognitive biases, or expression limitations). A typical initial setting, for example, describes "a 28-year-old software engineer who feels anxious about career development under remote work because a colleague who regularly goes to the office received a promotion". The initial setting supplies character motivation and cognitive-bias cues for the subsequent dialogue: the simulated user generates questions consistent with the persona, so the question distribution carries individualized color rather than impersonal neutral phrasing.

Although the simulated user's question generation must preserve individualized character, the dialogue log should be written from an objective third-party observer's perspective. This perspective simulates the external influence of an objective environment while avoiding the information omissions and perspectival limitations of a single user's stance, and prevents the dialogue from overfocusing on any particular domain. The log-update prompt specifies four writing requirements: (1) use an objective narrative observer's tone, with concise language resembling a game character's personal log updated after an event; (2) summarize the current turn's user request and the model's reply,

retaining core data and final conclusions; (3) incorporate the evaluator's score and reasons, summarizing reply quality — which aspects meet expectations and which are deficient — and explicitly identifying errors or omissions; (4) based on the evaluation results and the initial setting, naturally indicate, from the third-party perspective, directions the user may subsequently pursue. The output is structured JSON containing a `reference` field, namely the full text of the updated observer log.

The observer log enhances dialogue coherence along two pathways. The first is **progress awareness**: the reference explicitly records how far the dialogue has progressed and which demands have been resolved, so that new questions connect naturally with what precedes them. The second is **quality feedback**: the log fuses the evaluator's scores and error identification, allowing the simulated user to generate follow-ups that better match real-user psychology — for instance, digging further into points the model answered incompletely or incorrectly. The two pathways make the log the framework-side "working memory" of the evaluation: this memory rolls forward turn by turn, always summarizing the entire history at a fixed length.

### B.3 Sliding-Window History Truncation

In long-dialogue evaluation, the framework-side stages of evaluation scoring, log updating, and new-question generation all rely on the dialogue history as input. If the full history is concatenated turn by turn, the input of a single stage at turn $t$ scales as $O(t)$, and the cumulative cost of an entire evaluation grows quadratically with depth ($\sum_{t=1}^{d} O(t) = O(d^2)$); moreover, overly long contexts degrade the instruction-following and information-extraction stability of the auxiliary models (Liu et al., 2024a). To address this, Trace-MT introduces a sliding-window history truncation mechanism built on the observer log. Its core is a two-level input structure of "log prefix plus verbatim window": early history is compressed while recent history is retained verbatim, controlling computational cost while maintaining contextual coherence.

Specifically, the window length is set to $L = 6$ (aligned with the N-gram sequence length used in path generation). At turn $t$ ($t > L$), the history $D_{1:t-1}$ is split into two parts: the most recent $L$ turns $D_{t-L:t-1}$ are retained verbatim, while the earlier segment $D_{1:t-L-1}$ is replaced by the observer log $r_{t-L-1}$ updated at the end of turn $t-L-1$ (which covers exactly turns 1 through $t-L-1$). The framework-side input at turn $t$ is

$$\text{Input}_t = r_{t-L-1} \oplus D_{t-L:t-1}, \tag{1}$$

where $\oplus$ denotes sequential concatenation. When $t \leq L+1$, the window has not crossed the history boundary, the prefix degenerates to empty, and the input is the full concatenation $D_{1:t-1}$, exactly as without truncation. Figure 7 illustrates the two-level input structure and the growth of the log prefix across these regimes.

Notably, the truncation is strictly confined to framework-side components: the model under test always receives the full verbatim history. This design both ensures the validity of measurement targets such as long-range memory and guarantees that "the information on which a question is grounded is a subset of the information visible to the model under test", so that unanswerable questions caused by an information gap are avoided. The mechanism is enabled only at the medium and high difficulty tiers. Its central advantage is balancing computational efficiency against information retention: by reducing the framework-side per-turn input length from $O(t)$ to the constant $O(L)$, the cumulative cost of an entire dialogue drops from $O(d^2)$ to the linear $O(d \cdot L)$, substantially cutting token consumption and effectively suppressing the instruction-forgetting risk faced by the simulated user model on long sequences. The observer logs analyzed in this appendix are generated by the simulated user model under the system prompt below, reproduced in its original Chinese form.

**Observer-log generator system prompt**

你是一个多轮对话评测系统的"观察者日志"生成器。你的职责是根据初始设定，对话历史和当前轮的对话及其评估结果，以旁观者视角撰写一段类似游戏角色日志的叙述性文字，记录当前轮的完成情况，并适当引导后续对话。

## 日志撰写要求

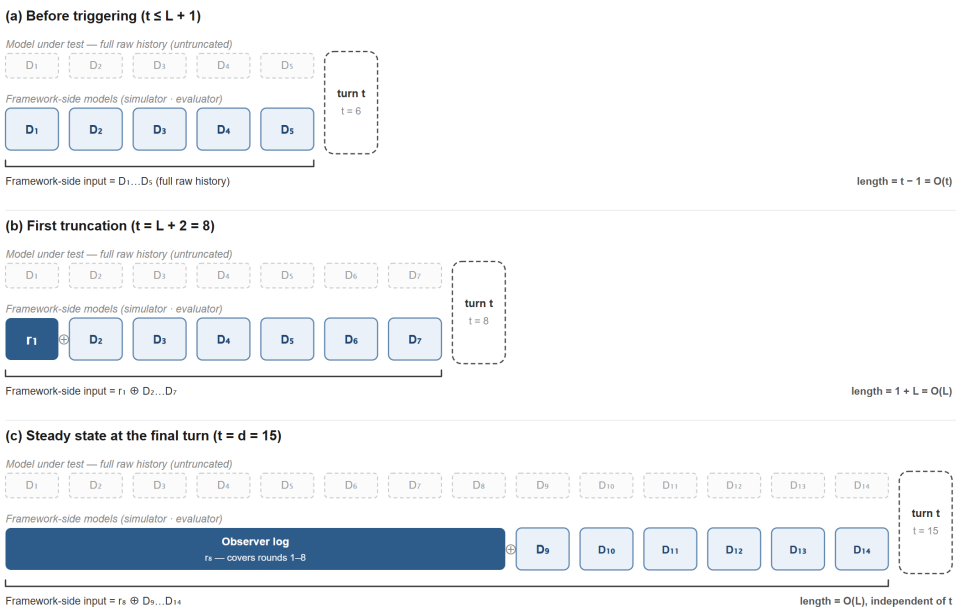



Figure 7: Sliding-window history truncation for framework-side models (window length $L = 6$).

1. **风格**：采用客观、叙事的观察者口吻，就像游戏角色在经历事件后更新的个人日志，语音应该简单凝练，输出本轮对话带来的进展与变化。

2. **内容**：

- 简要概述本轮用户提出了什么需求或问题，模型给出了怎样的回复，你需要保留一些核心数据和最终结论。

- 结合裁判模型的打分及原因，总结回复的质量表现：哪些方面符合预期，哪些存在不足。

- 若模型回复未能解决用户问题，或是回复中存在错误或遗漏的内容，应该指出这些不足。

- 基于评估结果，以及初始设定内容，自然地引出后续对话中用户可能更关注的事物，或想要进一步了解的方向，但是是从第三者视角来看的，而不是用户或评测的视角。

3. **形式**：在进行一步步思考后，最终输出一个Markdown 格式的json体，如下所示：

...[一些思考内容]...

“‘json ”reference”: ”更新后的参考文本，包含本轮对话的完成情况和后续对话的引导。” “‘

# C Test Data Construction

## C.1 Initial Seed Questions

Seed questions form the root nodes of the deterministic state- transition tree described in Section 3.1, and their quality determines the examination space that each Trace can unfold. Corresponding to the deterministic type sequence of a path, a seed question must satisfy two requirements: first, it must be embedded in a realistic scenario with multi-turn extensibility, able to support path unfolding over 5 to 15 turns; second, it must come with a rich user- background narrative, which serves as the direct source of the initial setting (`root_reference`) described in Section B.2. To this end, seed questions are constructed through two complementary channels: augmentation from existing evaluation data and domain-directed generation. Both channels are carried out by an auxiliary large

model under dedicated prompts; a single call outputs a JSON array in which each element contains two fields, `new_question` (the user question) and `background` (the background narrative). To scale up production, after the first generation round a fixed follow-up instruction ("generate some more different data") is used for multiple iterations, parsing and accumulating results round by round and improving sample diversity through deduplication constraints.

**Augmentation from existing evaluation sets.** This channel seeds on items from public benchmarks and instruction datasets: Arena-Hard items originate from real users' high-difficulty dialogue requests (Li et al., 2024), MMLU-Pro and GPQA-Diamond cover expert-level STEM reasoning tasks (Wang et al., 2024b; Rein et al., 2023), and Alpaca provides diverse everyday instruction tasks (Taori et al., 2023); together they form a seed pool with complementary capability dimensions and difficulty levels. The design intent is to inherit the capability-dimension coverage and difficulty calibration of mature benchmarks while escaping their textbook-like form by "germinating new questions" rather than rewriting the originals, so that root questions resemble the way real users actually ask.

**Domain-directed generation.** This channel relies on no seed items; instead, a domain-instruction slot specifies the target domain, and a model cast as an "evaluation engineer" with cross-domain professional experience generates questions from scratch. The covered domains and their typical sub-directions include finance, healthcare, aerospace, transportation, materials science, non-programming computer applications, and creative writing. Directed generation imposes four requirements on questions: **scenario grounding**, i.e., no concept is asked directly — a concrete work scenario must be set; **high complexity**, i.e., the question contains multiple conditions or parameters, forcing multi-step reasoning and analysis; **multi-turn extensibility**, i.e., the question must leave natural follow-up "openings" that can lead to parameter-adjustment advice, troubleshooting steps, option comparisons, code or formula details, and other subsequent interactions; and **realism**, i.e., textbook-style generic questions are avoided and difficulty is not excessive, matching the thorny problems ordinary practitioners encounter in real work. In addition, the initial question contains only one or two core demands, avoiding intent overload that would compress the space for later turns.

**Background narratives and their connection to the log system.** Every sample produced by both channels comes with a coherent background narrative (`background`) whose structure corresponds strictly to the four elements of the initial setting described in Section B.2: the **user persona**, **mental state**, **interests**, and **latent conditions**, typically in no fewer than 3 to 5 sentences. This narrative enters the dynamic dialogue pipeline of Section C.2 as the root node's initial setting (`root_reference`): on one hand it supplies the simulated user with character motivation and cognitive-bias cues, giving the questions of later turns an individualized distribution (Section B.2); on the other hand, the conceptual confusions and cognitive biases preset in the "latent conditions" element provide natural leads for error-inducing follow-ups at `itp_ch` (challenge) nodes, while the extensibility openings reserved in the questions ensure that nodes such as `conn_gr` (guided reasoning) and `bak_hr` (history recall) can unfold layer by layer along paths as long as 15 turns. The two channels complement each other: the augmentation channel inherits the capability coverage and STEM quantitative-reasoning orientation of public benchmarks, while the directed channel fills in the vertical- industry scenarios and creative-writing tasks that are scarce in benchmark data, together ensuring the breadth of the root-node set in both domain distribution and difficulty level.

### C.2 Dynamic Dialogue Pipeline

The test data are generated online through a three-party collaboration of "**model under test — simulated user model — evaluation model**": the node path determines "what type of question to ask", the observer log determines "whether the question is coherent", and the evaluation model determines "how good the answer is".

Before evaluation starts, node paths are pre-allocated cyclically to each seed question by difficulty from the pre-generated path library (the output of Section A.3): the binding between seed questions and paths is fixed before evaluation begins, so evaluating the same dataset at any time uses the same path sequences. The path sequences determine the question type and difficulty gradient of every turn, which means different models under test face the same set of dialogue-behavior tests,

and between-model score differences can be attributed to model capability rather than sampling fluctuation.

Thereafter, for each seed question, the system executes the following five steps turn by turn along the pre-allocated node path until the path is exhausted (the number of dialogue turns equals the path length):

1. The simulated user model generates a new question and that question's `expected_checklist` from the **current node-type template + initial setting + observer log**;
2. The model under test receives the user question and generates a reply;
3. The evaluation model scores the reply against the checklist and produces a `score_reason`;
4. The simulated user model updates the observer log following the procedure of Section B.2;
5. All information of the turn (question, node type, reply, score, scoring reason, and the turn's reference) is recorded, and the input for the next turn is constructed.

The three-party division of labor corresponds to the two classes of cost: the model under test produces only the dialogue being evaluated; the costs of the simulated user and evaluation models belong to the evaluation framework and are controlled by the sliding-window mechanism of Section B.3.

### C.3 New-Turn Question Generation

The simulated user model's input comprises the construction template for the current turn's node type (the six node prompts under `prompt/node_sys/`), the initial setting, the dialogue history, and the latest observer log. The output is structured JSON with three fields:

- `scenario`: the scenario description of the current turn's question;
- `question`: the new user question;
- `expected_checklist`: the expected answer points and their score allocation, with checkpoint weights assigned by importance and total weight $\sum_i w_i = 100$.

Each of the six node types defined in Section 3.1 has a dedicated prompt template that specifies the question-construction directions, generation requirements, and output format for that node type. The construction directions are the behavioral core of the node type: the template prescribes the dimensions along which questions of that type should be constructed, while the observer log constrains the content connection of the questions (what to ask and which thread to follow); the two together make the fixed type sequence instantiate into a natural and coherent concrete dialogue.

The construction directions of the `itp_ch` template deserve particular note: its challenges may target data and facts (direction ①), derivation logic (direction ②), or confirmed errors (direction ③), so that although this node is structurally a sequential construct (Section 3.3), it triggers the model's self-correction path through adversarial checking, corresponding to an assertion statement in software testing. The `bak_hr` template is dominated by direction ①, shifting to the cross-turn integration of direction ② when the historical context is complex — the former tests single-point recall and the latter information aggregation across multiple recall points, consistent with this node's positioning as a control-flow back-edge. The two construction directions of the `shift_nt` template correspond to the "completely unrelated" and "weakly related" classes of new intents: the former tests intent understanding after a hard topic switch, while the latter tests whether a soft topic switch induces misunderstandings in the reply.

The `expected_checklist` is produced synchronously by the simulated user model when it generates a question, and its score allocation provides the anchor for subsequent evaluation. For questions with objective answers, the checklist lists key information and reasoning steps; for open-ended questions, it indicates the core dimensions the answer should cover; in both cases weights are assigned by importance. This design generates the scoring anchor and the question semantics within the same model, minimizing cross-model semantic alignment bias. The six node-type system

prompts are reproduced below in their original Chinese form, ordered by node type; the Markdown markers are kept as in the source files.

**node-type system prompt: conn_fu (Connection)**

## 任务：构造多轮对话新一轮问题测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型对**追问**的处理能力。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的初始设定、对话记录以及对话日志，生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **使用指代词进行追问**在后续轮次中，用「他/ 她/ 它」「这个/ 那个」「上述」「该+名词」等方式指代前文出现的实体、概念或数据，并提出新的问题。

2. **询问类似事务，检验关联能力**提出一个与原始问题相似但不同的新事务，观察模型是否能识别出与前一话题的关联，并在回答中合理引用前文。

3. **从模型回复中选择部分内容，让模型深入解答** 在模型回答中提取关键信息，构造新问题，让模型进一步解释或计算。

4. **根据之前提供的材料或问题，继续询问** 在模型回答中寻找与当前问题相关的信息，并构造新问题。

在构造时，可以选择其中一种或多种方式进行构造，也可以根据实际需要自由发挥。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 确保生成的问题都或多或少的需要依赖前文信息进行解答，可以是前文的直接引用，也可以用指代或类比的方式模糊引用。

- 新的问题应该具有一定难度，避免过于简单或直接。

**node-type system prompt: conn_rw (Connection)**

## 任务：构造多轮对话新一轮问题测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型对**改写**的处理能力，包括对内容及格式的改写。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的所有历史对话生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **将模型回复中的内容进行定向修改**针对模型的回复内容，提出新的要求。

2. **格式化输出**将模型回复，以json，markdown等规范的格式输出，并确保格式正确。

3. **对模型回复内容的风格，语言等进行改写** 将模型回复进行整体层面的改动，包括风格，语种等。

4. **给定固定的字词，让模型使用或不使用**针对模型的回复内容，让模型必须使用某个词，或不允许出现xxx。

5. **自定义格式输出**将模型回复的内容换一种非结构化的形式输出，如合并为一段，分成n个小点，改成日常聊天的对话形式等。

在构造时，可以选择其中一种或多种方式进行构造，也可以根据实际需要自由发挥。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 确保生成的问题都或多或少的需要依赖前文信息进行解答，可以是前文的直接引用，也可以用指代或类比的方式模糊引用。

- 新的问题应该具有一定难度，避免过于简单或直接。

**node-type system prompt: conn_gr (Connection)**

## 任务：构造多轮对话追问测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型对**引导推理**的处理能力。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的所有历史对话生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **引入新的条件信息，引导模型进行推理**-提出与原始问题相关但未在对话中明确提及的条件，引导模型进行推理。

2. **引入新的背景信息，引导模型进行综合分析**-提出与原始问题相关但未在对话中明确提及的背景信息，引导模型进行综合分析。

3. **修改原题目中的部分条件，引导模型进行假设推理**-提出与原始问题相关但未在对话中明确提及的假设，引导模型进行假设推理。

在构造时，可以选择其中一种或多种方式进行构造，也可以根据实际需要自由发挥。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 确保生成的问题都或多或少的需要依赖前文信息进行解答，可以是前文的直接引用，也可以用指代或类比的方式模糊引用。

- 新的问题应该具有一定难度，避免过于简单或直接。

**node-type system prompt: itp_ch (Interruption)**

## 任务：构造多轮对话新一轮问题测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型的回复在被用户**质疑**后的处理能力。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的所有历史对话生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **质疑回复中的具体数据/结果/事实基础等具体信息**针对模型的回复结果中的数据，结论等，给出错误诱导或反驳进行质疑。

2. **质疑回复中的推导逻辑**针对模型的回复结果，使用逻辑漏洞，偷换概念或扩大范围等诡辩的方式，引入新的结论，观察模型能否正确分析。

3. **纠正模型在上一轮回复中的错误** 根据给定材料参考中的信息，纠正模型在上一轮回复中的错误。

在构造时，可以选择其中一种或多种方式进行构造，也可以根据实际需要自由发挥。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 确保生成的问题都或多或少的需要依赖前文信息进行解答，可以是前文的直接引用，也可以用指代或类比的方式模糊引用。

- 新的问题应该具有一定难度，避免过于简单或直接。

**node-type system prompt: shift_nt (Shift)**

## 任务：构造多轮对话新一轮问题测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型对于用户**新意图**问题的处理能力。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的初始设定、对话记录以及对话日志，生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **开启一个与之前无关的全新的话题**新一轮的问题与之前完全无关。

2. **开启一个与之前对话相似，但是没有直接关系的话题**新一轮的问题与之前轮的对话有一定的联系，但是不直接相关。

在构造时，可以选择其中一种或多种方式进行构造，也可以根据实际需要自由发挥。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 确保生成的问题与之前轮次的对话基本没有直接关系。

- 新的问题应该具有一定难度，避免过于简单或直接。

**node-type system prompt: bak_hr (Back)**

## 任务：构造多轮对话新一轮问题测试数据

你是一名专业的测试数据构造专家，擅长设计用于评估大模型对话能力的多轮交互场景。

### 任务目标

根据已有的历史对话信息，生成新一轮**多轮对话**测试数据，重点考察模型对于**历史轮对话**中信息的记忆和处理能力。每个数据样本会包含一个初始设定，一组或多组用户与模型之间的对话记录。你需要根据提供的初始设定、对话记录以及对话日志，生成新一轮的问题，以及该问题的理想回答要点（或完整答案）。

### 构造方向

在构造测试数据时，可以从以下方面进行构造：

1. **根据在上一轮之前的历史轮次中的一些信息进行提问**跳过上一轮对话，询问上上一轮，甚至更早之前的历史对话中的一些内容。

2. **整合之前轮的信息，再提出问题** 构造的问题需要涉及前面多个轮次的对话，模型需要将前面多个轮次对话中的信息整合起来再回复。

数据构造以第一点为主导方向；若历史上下文复杂度高、更适于生成第二类数据，则可转向第二点。具体视对话语境灵活选用。

### 生成要求

- 保持自然对话风格，生成的问题应像真实用户一样简洁、口语化（但不必加语气词），逻辑上自然衔接上文。

- 生成的问题需要追溯到至少两轮之前的问答记录，不能只依赖上一轮的对话。

- 新的问题应该具有一定难度，避免过于简单或直接。

# D EVALUATION

Because a unified scoring rubric across dialogue scenarios easily induces rating-scale drift and poor interpretability, we attach a checklist to every question. At the implementation level, the simulated user model produces the checklist synchronously when generating each turn's question: each checkpoint corresponds to a core fact, reasoning step, or contextual constraint that the reply must cover, and is assigned a weight according to its importance. This same model serves as both the simulated user and the scoring judge, because one model holds the most consistent understanding of the semantic boundaries and weight granularity of the checkpoints, which minimizes cross-model semantic alignment bias. The scoring results also feed into observer-log updates: the model's reply

score and the error points in the reply are updated into the `dialogue_reference` in real time (Section B.2), providing quality- feedback signals for the simulated user to generate subsequent follow-ups (especially deep-digging follow-ups that seize on the model's omissions and errors), so that dialogue progression responds dynamically to the model's actual performance and not only to the contextual content.

**Scoring mechanism.** Scoring proceeds turn by turn, anchored on the **expected reply checklist** (`expected_checklist`). The simulated user model produces the question's `expected_checklist` synchronously with the question: it is not a standard answer but a set of key-information and reasoning-step checkpoints with importance-weighted scores, $\kappa = \{(c_i, w_i)\}_{i=1}^{m}$, where $\sum_i w_i = 100$. Anchoring scoring on checkpoints rather than reference answers balances scoring objectivity against reply freedom — a model may cover the checkpoints in any reasonable way, but omitting key information incurs weighted deductions. The `score_reason` field records the judgment for each checkpoint item by item, providing an auditable basis for manual spot checks and case analysis (Section 4). This mechanism constrains the free-form scoring drift common in LLM reviewing (Zheng et al., 2023) into a weighted aggregation of item-by-item binary/partial-satisfaction judgments, improving both the stability and the interpretability of scores. The turn-level scores and `score_reason` fields analyzed above are produced by the evaluator model under the system prompt below, reproduced in its original Chinese form.

**Eval system prompt**

你是一个严格、公正的多轮对话质量评估专家。你的任务是根据对话历史和预先定义的期望回复检查列表，评估当前轮次中模型回复的质量，并给出分数与理由。

## 评估要求

1. 从当前轮对话中提取出模型的回复内容。

2. 首先判断回复内容是否理解了上文对话，若用户问题与上轮对话相关，但模型在回复时没有考虑上文的内容，则该回复不满足要求，直接扣除60分。

3. 逐条对照「期望回复检查列表」，判断模型回复是否完全满足每一项要求。

4. 根据满足程度给出评分（0-100 分，100 分为完美满足所有检查项，0 分为完全不满足）。

5. 若模型完全满足所有的检查项，并且在此基础上还进行了一些扩展，例如提供了额外的信息或建议，则该回复的评分会增加10分。

6. 最后综合以上的扣分点或加分点，给出最终的评分。

7. 在评分原因中，简要说明每项检查点的满足情况，并总结整体表现。若存在不满足的地方，请明确指出。

# E Full Experimental Results

## E.1 Paired Significance Tests for Model Rankings

This section gives the details of the model-ranking significance tests of Section 4.2. The tests use a paired design: for each pair of models, the Overall difference is computed in a paired manner only on the paths shared by the two models, and the stratified cluster bootstrap resamples whole paths at the shared-path level ($B$=2,000) to absorb the within-path correlation of turn-level scores; multiple comparisons are corrected with BH-FDR. All 36 pairs are summarized in Figure 8.

## E.2 Component-Density Bin Statistics

This section supplements the window-level analysis of Figure 3(a) in Section 4.3 (high difficulty tier, pooled over nine models); the per-bin statistics are given in Table 3.

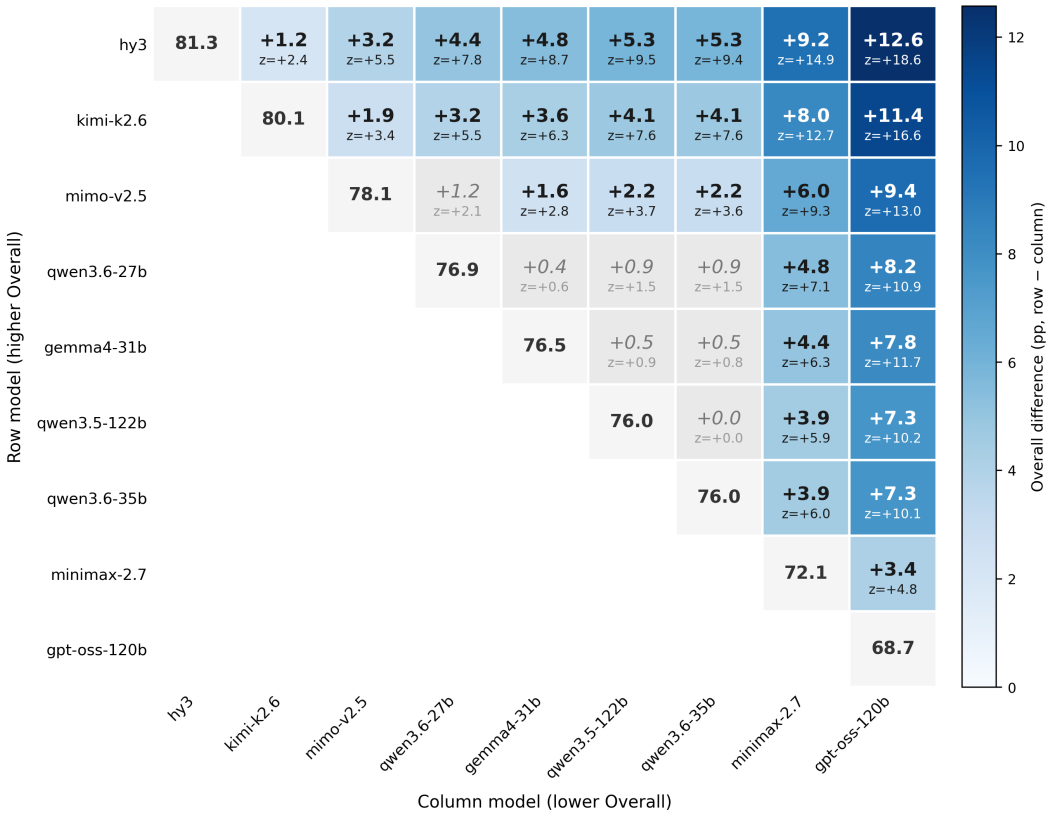



Figure 8: Model-by-model Overall difference heatmap. Models are ordered by Overall in descending order; upper-triangle cells show the difference (in pp) of the row model (the higher-ranked one) relative to the column model, with the $z$ statistic at the bottom of the cell; color depth encodes the difference magnitude; diagonal cells show each model's Overall score; gray italic cells are pairs that are not significant after BH-FDR correction ($q \geq 0.05$): `mimo-v2.5`–`qwen3.6-27b`, and all six pairs within {`qwen3.6-27b`, `gemma4-31b`, `qwen3.5-122b`, `qwen3.6-35b`}.

Table 3: Component-density bin statistics. L1–L3 use the window-level convention; L0 uses the turn-level convention, pooling `conn` turns across environmental densities (whose internal differences are shown in Table 5), and serves only as a horizontal reference since it differs from the window-level convention. The confidence intervals of adjacent density bins do not overlap.

| Density bin | Window composition | # samples | Mean | 95% CI |
|---|---|---|---|---|
| L0 (sequential baseline) | `conn` turns | 22,900 | 77.21 | [76.86, 77.61] |
| L1 ($k$=1) | 1 construct + 5 `conn` | 1,944 | 79.90 | [79.26, 80.57] |
| L2 ($k$=2) | 2 constructs + 4 `conn` | 5,805 | 76.71 | [76.30, 77.12] |
| L3 ($k\geq$3) | $\geq$ 3 constructs | 23,725 | 74.53 | [74.18, 74.92] |

Three aspects of these statistics merit elaboration. First, the L1 excess over the baseline: each of the three control constructs in isolation scores no lower than the baseline, so model performance under single interference is indistinguishable from the interference-free case; the medium tier happens to require only that each of the three construct types be covered in isolation, which is why an evaluation stopping at branch coverage would observe only this benign regime. Second, the degradation is driven by density rather than composition or arrangement: within the same density bin, score differences across construct combinations and arrangements are limited, so the risk cannot be foreseen by observing any particular interference combination at the lower coverage tiers. Third, the

Table 4: Per-model placebo slopes and drift-corrected L1–L3 gaps (high tier; 95% cluster-bootstrap CIs; $q$ from BH-FDR over the nine models). Corrected gaps use each model's within-L1 drift slope. With 216 L1 windows per model, individual models are underpowered relative to the pooled analysis; the observed per-model gaps span $-2.80$ to $-6.76$pp.

| Model | Slope (pp/turn) | Observed gap | Corrected gap [95% CI] | $q$ |
|---|---|---|---|---|
| qwen3.6-27b | $-1.21$ | $-6.46$ | $-0.54$ $[-3.46, +1.97]$ | 0.75 |
| qwen3.6-35b | $-0.57$ | $-5.72$ | $-3.03$ $[-7.03, +0.67]$ | 0.52 |
| gemma4-31b | $-0.85$ | $-4.60$ | $-0.60$ $[-4.08, +3.09]$ | 0.76 |
| qwen3.5-122b | $-1.12$ | $-6.76$ | $-1.36$ $[-5.75, +3.35]$ | 0.75 |
| gpt-oss-120b | $-0.78$ | $-5.35$ | $-1.92$ $[-7.01, +5.65]$ | 0.75 |
| minimax-2.7 | $-0.51$ | $-4.93$ | $-2.51$ $[-6.24, +1.26]$ | 0.54 |
| hy3 | $-0.72$ | $-5.03$ | $-1.61$ $[-4.60, +2.12]$ | 0.69 |
| mimo-v2.5 | $-0.13$ | $-2.80$ | $-2.38$ $[-6.25, +1.97]$ | 0.66 |
| kimi-k2.6 | $-0.19$ | $-6.64$ | $-5.77$ $[-7.98, -2.82]$ | 0.02* |

per-model curves in Figure 3(b) decline by about 3pp to nearly 7pp from each model's L1 peak, and the divergence in drop magnitude profiles each model's density sensitivity.

### E.3 Position Control for the Density Gradient

This section separates the window-density gradient of Figure 3(a) from natural drift — the decline of scores with dialogue position that occurs even without interference. All confidence intervals are 95% percentile cluster bootstraps ($B$=2,000; cluster = model $\times$ path), and the drift correction re-estimates its slopes inside every bootstrap replicate.

**Density is entangled with position.** Single-construct windows can occur only at path boundaries, so denser windows sit later in a path: the density–position correlation is $r$=0.367, and L3 windows are on average 4.87 turns later than L1 windows (mean center positions 3.79, 6.68, and 8.67 for L1, L2, and L3). Structurally, L1 windows exist only at center positions 3, 4, and 13 (1,134, 729, and 81 windows pooled over nine models); no single-construct window can occur at middle positions, so any position control rests on these three support points.

**Placebo slope.** Natural drift is estimated within L1 windows only, where density is held constant at the reference level. The per-model slopes range from $-0.13$pp per turn (`mimo-v2.5`) to $-1.21$pp (`qwen3.6-27b`): drift is real, model-dependent, and non-negligible.

**Three position-controlled estimates.** First, correcting each window to its model's mean L1 position with the within-L1 slope yields corrected gaps of $-1.26$pp ($[-2.17, -0.37]$) for L2 and $-2.10$pp ($[-3.43, -0.80]$) for L3. Second, comparing bins at matched positions: the early stratum (center positions 3–4; 1,863 L1 and 855 L3 windows) yields $-2.31$pp ($[-3.47, -1.15]$), while at the final position (13; 81 L1 and 1,393 L3 windows) the gap vanishes ($-0.05$pp, $[-2.58, +2.63]$; equal-weight combination $-1.19$pp, $[-2.56, +0.23]$). Third, an OLS regression of window score on position and bin dummies yields a position coefficient of $-0.44$pp per turn ($[-0.49, -0.38]$), with L2 at $-1.92$pp ($[-2.62, -1.26]$) and L3 at $-3.24$pp ($[-3.95, -2.55]$); the variance inflation factor of position and density is 1.16, so the regressors are only mildly collinear at the window level, although L1's absence from middle positions means the position control extrapolates from three support points.

**Interpretation as a bracket.** The three estimates converge on a density-attributable L1–L3 gap between $-2.1$ and $-3.2$pp. The raw gap ($-5.36$pp) is an upper bound that attributes all positional decline to density; the drift-corrected gap ($-2.10$pp) is a conservative lower bound, because the drift baseline is itself contaminated: position-13 L1 windows follow paths that exhausted their construct caps early, so their decline partly reflects carryover from earlier interference rather than pure position. The vanishing gap at the final position supports this cumulative reading — once a path has accumulated enough interference, its late turns are depressed regardless of the current window's density. Table 4 reports per-model statistics: after BH-FDR correction over the family of nine tests, only `kimi-k2.6` retains an individually significant corrected gap ($q$=0.018); with 216 L1 windows per model, per-model power is limited, and the pooled estimates above remain the appropriate level of analysis. Figure 9 visualizes the three aspects.

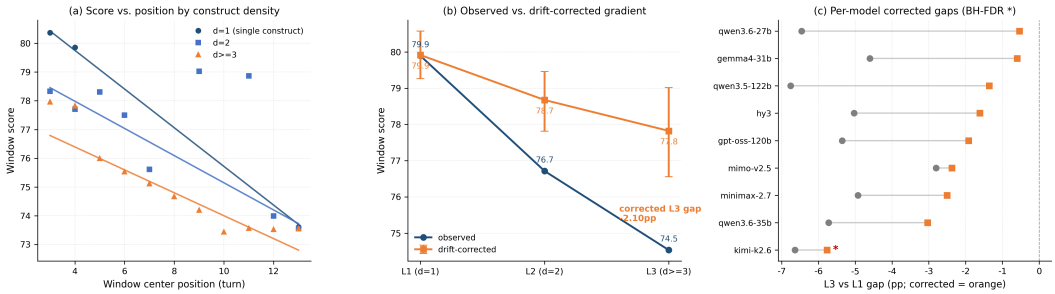



Figure 9: Position control for the density gradient (high tier, pooled over nine models). **(a)** Window score versus window center position for single-construct ($d$=1), construct-pair ($d$=2), and composite ($d\geq$3) windows. Markers are per-position pooled means (shown only where at least 30 windows exist) and are deliberately not connected, so unsupported positions do not interpolate a local trend; solid lines are group-level OLS trend fits over all windows. All three fitted trends decline with position, and the denser groups lie below the single-construct group; single-construct windows exist only at positions 3, 4, and 13. **(b)** Observed versus drift-corrected L1/L2/L3 means (95% clustered bootstrap CIs): the linear correction removes the positional component but leaves the L2 and L3 gaps significantly negative. **(c)** Per-model observed (gray circles) versus drift-corrected (orange squares) L1–L3 gaps; the star marks the single model surviving BH-FDR (`kimi-k2.6`, $q$=0.018).

Table 5: Turn-level scores versus environmental density (turn counts in parentheses). The sparse ($d\leq$1) versus dense ($d\geq$2) contrast for `conn` turns is $-4.07$pp (95% CI $[-4.95, -3.22]$), and the decline is monotonic on all nine models (`kimi-k2.6` from 85.73 to 78.91; `hy3` from 85.16 to 81.22). There are no $d$=0 `conn` turns in the high tier, so the count starts at $d$=1.

| Turn type | $d$=1 | $d$=2 | $d\geq$3 | $d$=1→3+ change |
|---|---|---|---|---|
| `conn` (cooperative processing) | 80.67 (3,465) | 77.64 (8,865) | 75.72 (10,570) | −4.95 |
| `bak_hr` (long-range recall) | 81.94 (198) | 76.03 (738) | 74.78 (4,787) | −7.16 |
| `shift_nt` (topic shift) | 75.08 (504) | 73.37 (1,314) | 73.04 (6,758) | −2.04 |
| `itp_ch` (assertion) | 74.53 (207) | 70.08 (666) | 72.21 (4,850) | −2.32 |

### E.4 Turn-Level Scores versus Environmental Density

This section details the turn-level attribution analysis of Section 4.3 (high difficulty tier, pooled over nine models); the values are given in Table 5.

Two candidate causes underlie the emergent degradation: the construct turns themselves become harder when stacked, and context-dependent turns are polluted by surrounding constructs. To disentangle their relative contributions, the environmental density $d$ is defined as the number of control constructs in the 6-gram window centered on the turn (clamped at path boundaries; $d \geq 3$ merged). The results support the pollution mechanism as the dominant cause: were the hardening of construct turns themselves the main cause, all four turn types should show similar declines, whereas in fact `shift_nt` and `itp_ch` turns fluctuate only slightly with density while `conn` turns decline monotonically by nearly 5pp and `bak_hr` turns decline even more deeply. The mechanism also quantitatively reproduces the window-level gradient of Figure 3(a): the $k$=1 window mean coincides with the weighted score of sparse `conn` turns and single-construct turns, while the $k\geq$3 window mean matches the scores of the component turns in dense environments.

### E.5 Correction Stability and the Matched-Control Analysis

As the assertion construct in the control flow, `itp_ch` participates in the coverage targets of the higher tiers with a higher occurrence cap, probing the stability of the model's self-correction when interferences stack. All adjacent `itp_ch`→`conn` turn pairs in the high tier are tabulated: the assertion-turn score measures immediate correction quality, while the score of the immediately following `conn` turn measures whether the correction is retained; environmental density is binned by

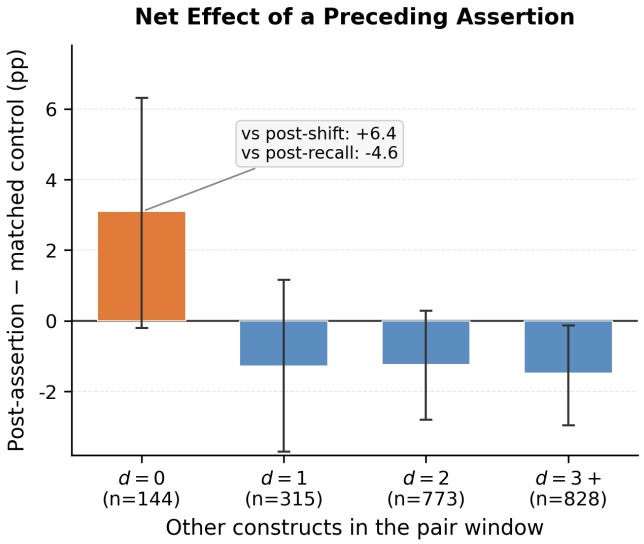



Figure 10: Post-assertion turn score minus the mean of matched controls (same model, same dialogue position, same environmental density; 95% clustered bootstrap confidence intervals), by pair environmental density. The $d$=0 excess is the mixture of a $+6.40$pp advantage over post-shift controls and a $-4.64$pp deficit against post-recall controls.

the number of other control constructs in the 6-gram window centered on the pair (the focal assertion is not counted; $d\geq 3$ merged).

**Dissociation between correction and retention.** The immediate-correction score of assertion turns shows no systematic decline as density grows (fluctuating between 71 and 75 across bins), whereas the post-correction turn drops markedly with density. This dissociation corresponds one-to-one with the turn-level attribution in Figure 4: assertion response requires only local information about the current assertion and is a density-insensitive turn, whereas correction retention unfolds on the following `conn` turn, a behavior dependent on complete context and precisely one of the main bearers of state pollution.

**Reversal relative to the sequential baseline.** Relative to the sequential baseline, the post-assertion turn sits $+3.08$pp above it when the assertion appears in isolation ($d$=0) and $-2.76$pp below it at $d$=2; the pooled low-density ($d\leq 1$) versus dense ($d\geq 2$) contrast is $-2.65$pp, with a 95% confidence interval of $[-4.96, -0.28]$.[4] These baseline-relative values mix the assertion's own effect with where assertions of each density live; the two analyses below separate the two.

**Matched-control analysis.** For every post-assertion turn we use as controls the `conn` turns of the same model, at the same dialogue position, with the same environmental density $d$, that are not preceded by an assertion; all 2,060 pairs admit such controls (Figure 10). The net effect of a preceding assertion is $-1.04$pp (95% CI $[-1.94, -0.13]$) pooled over densities and $-1.36$pp ($[-2.45, -0.25]$) in dense contexts. The $d$=0 bin requires care: isolated assertions sit at turn 4 in 135 of 144 pairs, and in these early single-construct windows the generator places the sole construct immediately before the focal turn, so the available controls are precisely post-shift and post-recall turns. Against them the post-assertion turn averages $+3.09$pp ($[-0.20, +6.31]$) — itself the mixture of a $+6.40$pp advantage over post-shift controls ($[+3.14, +9.64]$) and a $-4.64$pp deficit against post-recall controls ($[-8.09, -1.32]$) — so the baseline-relative excess at $d$=0 reflects the position and composition of isolated assertions rather than a refocusing effect of the challenge. A cluster-robust regression of `conn`-turn scores on preceding-node indicators with position, density, and model fixed effects confirms the ordering: $+1.13$pp after a topic shift ($[+0.46, +1.81]$), $-0.52$pp after a recall ($[-1.47, +0.43]$), and $-1.14$pp after an assertion ($[-2.21, -0.07]$), each relative to routine continuation.

[4] $d$=0 pairs are structurally rare, so the dichotomized contrast serves as the primary reported quantity.

Table 6: Type compliance of question generation. *Match* is the type-match rate, corresponding to judgment item A1, i.e., the share of generated questions that experts judge to instantiate their intended node type. *Grounded* is the grounding rate, corresponding to item A2; grounding takes an explicit or an implicit form, and Grounded aggregates both. *Natural* is the naturalness rate, corresponding to item A3, i.e., the share of questions that experts rate as natural. `shift_nt` introduces a new topic whose questions by definition do not rely on prior context, so Grounded is not applicable.

| Node type | Match | Grounded | Natural |
|---|---|---|---|
| `conn_fu` | 100% | 98.6% | 90.3% |
| `conn_rw` | 100% | 100.0% | 87.5% |
| `conn_gr` | 100% | 97.2% | 86.1% |
| `itp_ch` | 100% | 96.7% | 85.6% |
| `shift_nt` | 100% | — | 90.7% |
| `bak_hr` | 100% | 100.0% | 83.3% |

**Within-path trajectory.** Restricting to conn–assertion–conn triples and correcting for the pooled positional drift ($-0.56$pp per turn, $[-0.61, -0.50]$), the post-assertion turn does not exceed the pre-assertion turn in sparse contexts ($-0.05$pp, $[-2.60, +2.51]$) and falls $-2.38$pp short in dense ones ($[-4.82, +0.02]$); at $d$=0 the drift-corrected difference is $-0.73$pp ($[-5.26, +4.00]$).

**Per-model consistency.** Figure 5(b) shows that this effect is not a pooling artifact: its direction is consistent across all nine models — post-assertion turns in dense contexts are both below each model's own low-density performance and, without exception, below that model's own sequential baseline (gaps of 0.43–4.27pp, deepest for `qwen3.5-122b` and shallowest for `gpt-oss-120b`). Against the broadly reliable immediate correction, correction retention is a weakness shared by all models, and the divergence in gap magnitude provides yet another dimension for distinguishing models by interference sensitivity.

# F Evaluation Reliability Analysis

Trace-MT's evaluation pipeline is LLM-driven — a simulated user generates questions and checklists and an evaluator model assigns the scores — and question content is produced by dynamic instantiation, so two questions must be answered head-on. First, *scoring accuracy*: do the judge scores, assigned against model-generated checklists, agree with human expert judgment? Second, *generation stability*: when the same model faces identical type sequences in two independent evaluations, how much do its scores fluctuate, and do the fluctuations alter the comparative conclusions across models? This appendix answers the two questions with two studies: Section F.1 validates scoring accuracy through an expert audit, and Section F.2 quantifies the end-to-end fluctuation of dynamic generation through repeated evaluations.

## F.1 Expert Audit

**Audit design.** We draw a subset of 50 instances from the main benchmark for expert auditing. The audit covers three stages of the evaluation pipeline under three judgment protocols. *Protocol A* (question generation) judges whether each generated question realizes its intended node type, whether it relies on prior context, and how natural it is; *Protocol B* (scoring criteria) judges the consistency of the expected checklist with the question, its decidability, and its overall usability; *Protocol C* (scoring results) has experts evaluate, under blind review, the fairness of scores and the type of bias. Protocol C uses four rating bands (poor $<$40, fair 40–60, good 60–80, excellent 80–100). In addition, about 15% of the positions are independently re-annotated by a second annotator to estimate inter-annotator agreement.

**Type fidelity of question generation.** Table 6 shows that question generation meets the fidelity target at the type level. Generated questions are consistent with the design intent: every node is instantiated faithfully to its definition in the type taxonomy, so the type sequences presupposed by the N-gram coverage matrix hold at the question level. Naturalness, however, leaves substantial room for improvement. The first issue is semantic adjacency between node types: `bak_hr` and

Table 7: Checklist quality audit. *Consistent* is the consistency pass rate, corresponding to judgment item B1; a checklist passes when its items match the question and its context, with neither key requirements missing nor unwarranted ones imposed. *Decidable* is the decidability rate, corresponding to item B2; an item is decidable when it can be judged directly from the reply text, allowing the judge to score objectively. *Usable* is the overall usability rate, corresponding to item B4, counting instances judged either usable as-is or usable after repair.

| Node type | Consistent | Decidable | Usable |
|---|---|---|---|
| `conn_fu` | 95.8% | 91.7% | 91.7% |
| `conn_rw` | 97.2% | 94.4% | 94.4% |
| `conn_gr` | 91.7% | 88.9% | 90.3% |
| `itp_ch` | 94.4% | 88.9% | 90.0% |
| `shift_nt` | 94.4% | 89.8% | 91.7% |
| `bak_hr` | 93.3% | 88.9% | 91.1% |
| **Overall** | **94.4%** | **90.3%** | **91.5%** |

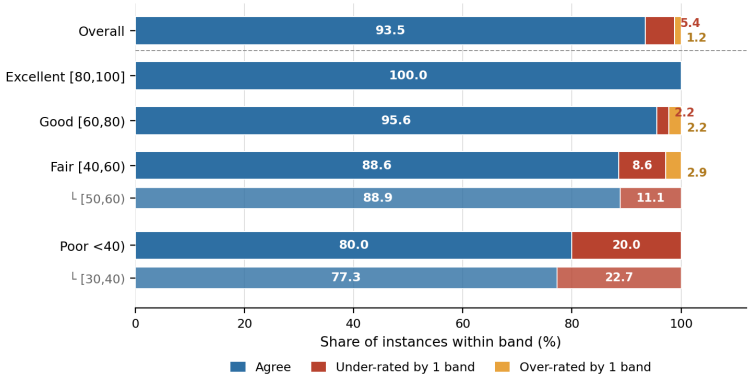



Figure 11: Band agreement between judge scores and blind expert ratings.

`itp_ch` overlap in one direction, and 37.8% of history-recall questions carry a challenging component beyond the recall. This is not a generation error: pressing on an earlier reply while recalling historical information is a natural composite behavior. Second, `bak_hr` has the lowest naturalness rate: such questions must restate or refer to content discussed earlier, and the simulated user tends to re-introduce that information in a stiff, explicit manner; real users rarely revisit a conversation they have just had in that tone.

**Checklist quality.** Table 7 shows that the dynamically generated checklists are reliable overall, with an overall usability rate of 91.5%; the structured process scores effectively prevent the judge from assigning coarse 0-or-100 scores based solely on answer correctness. This design is a double-edged sword, however: when the tested model's solution path does not match the process decomposition of the checklist, partial process points are also deducted. This is the negative edge of the design, whose reflection at the score level is examined next.

**Band consistency of scores.** Figure 11 contrasts judge scores with the bands assigned by blind experts. The overall agreement is good and contains no cross-band errors: the overall agreement rate is 93.5%, and every disagreement falls within a single band, with no case spanning two bands — the judge makes no directional or order-of-magnitude errors. Disagreements occur only near band boundaries, and the dominant bias type is over-strictness: 81% of the disagreements are judge scores lower than the experts', corroborating the checklist weakness noted above. The bias is global rather than directed at particular models: per-model band agreement ranges from 88.9% to 100% across the nine models, and Fisher's exact test gives $p = 0.87$ — strictness is a global tendency of the judge, with no systematic preference against any specific model.

Replacing every disagreeing score with the midpoint of the expert band and recomputing the nine models' scores on the subset changes each per-turn mean by at most 0.6pp, with the rankings un-

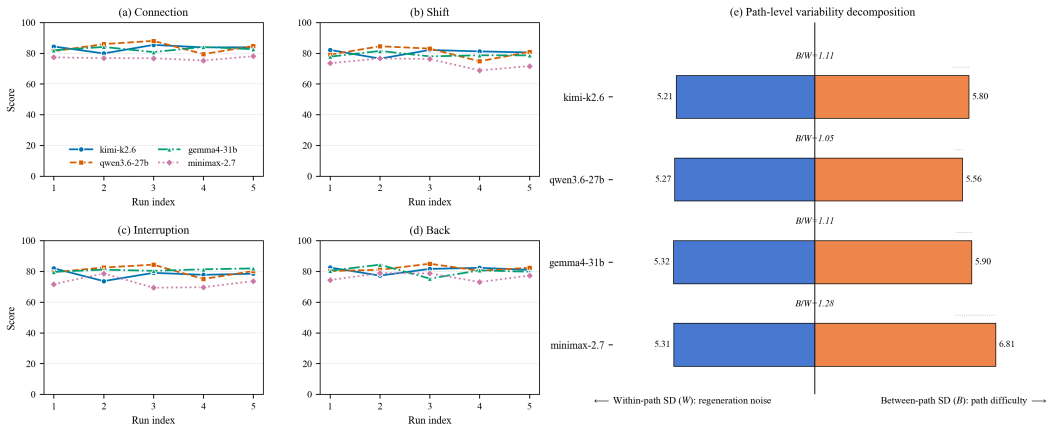



Figure 12: Overview of run-to-run stability. Left (a)–(d): per-run scores of the four evaluated models on the four advanced-level task types (connection, shift, interruption, back) across five repeated runs. Right (e): path-level end-to-end variability decomposition — each row is one model; the blue bar extending left is the within-path SD ($W$; regeneration noise), and the orange bar extending right is the between-path SD ($B$; intrinsic path difficulty); values are annotated at the bar tips, and the $B/W$ ratio is reported in the central gap.

changed. Judge error is thus a small, strict-leaning, non-systematic boundary effect that does not alter the ordering conclusions. Its source — overly fine-grained process decomposition in the checklist — also indicates the direction for improving the evaluation: while retaining the protection against polarized scoring, the weight of process points should be relaxed for replies whose conclusions are correct but whose processes are heterogeneous. Inter-annotator agreement is also good: across the three audited components (type fidelity of question generation, checklists, and resulting scores), the minimum agreement on re-annotated positions is 92.3%, confirming the validity of the annotations.

### F.2 Stability of Dynamic Generation

The question content in Trace-MT is generated turn by turn by the simulated user, conditioned on the tested model's actual replies and the observer log: in two independent evaluations of the same model, even with identical seed questions and identical node-type sequences, the concrete question instances and the model's own replies differ. This dynamism is the source of the benchmark's realism, but it also introduces a fluctuation component into evaluation scores that must be quantified. This section measures the end-to-end score fluctuation introduced by dynamic generation through repeated evaluations and examines its effect on the comparative conclusions across models.

The repeated-evaluation study selects four models spanning different rank segments in the main experiment (`kimi-k2.6`, `qwen3.6-27b`, `gemma4-31b`, and `minimax-2.7`), extends the subset used for expert validation to 100 seed questions, and independently evaluates each model five times on this larger subset. In every run, the seed questions and the node-type sequences given by the dialogue trees are held fixed, and the model configurations and parameters match the main experiment, so that between-run differences are attributable entirely to the end-to-end nondeterminism of the dynamic generation stage rather than to structural differences between evaluation paths. The score convention matches the main experiment, and the analysis counts only paths that fully unfold in all five runs.

Fluctuation is quantified at two levels. *Path level* (Figure 12, right): the within-path standard deviation $W$ — the standard deviation of a path's per-turn mean across five runs, averaged over paths — measures the end-to-end fluctuation of dynamic generation under a fixed evaluation path, while the between-path standard deviation $B$ measures the intrinsic difficulty difference across paths. Across the four models, $W$ is highly consistent and reaches 78%–95% of $B$: re-evaluating the same path changes scores by the same order of magnitude as switching to another path, so the score of any single path is not a stable quantity, and model comparison must rely on multi-path aggregation. *Model*

*level* (left): the per-run score trajectories of the four models on the four task types are steady, and the relative strengths across dimensions never reverse across runs; the standard deviation of Overall scores over the five runs is 1.1–3.1pp. In sum, the fluctuation from dynamic generation is bounded: it determines the smallest gap the benchmark can stably resolve, but does not change comparative conclusions within the resolvable range. After multi-path aggregation, gaps smaller than about 3pp between models cannot be stably resolved from a single evaluation, whereas gaps above this magnitude and the qualitative, dimension-level conclusions are unaffected.