ANATOMY OF A SPLIT: WHAT ONE WORKER PER FUNCTION COSTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Orchestrator-worker systems give each tool family its own LLM specialist. We ask, under preregistration, what that split costs in function calling and whether the loss can be repaired. The split we study is the simplest: a planner assigns each function a request needs to a same-model worker, each worker emits that function's calls, and the lists are concatenated, under a rule we set ourselves that keeps only the first item when the planner names the same function more than once; the comparison is one monolithic call (BFCL v4; Gemma-3-12B, Gemma-3-27B, Llama-3.3-70B; a sealed test set). Under the registered wording no split configuration we ran is shown to beat one call; under a rewording that costs the monolithic prompt 11.3 points at 12B, SPLIT+PASS sits 11.9 above it. The registered split loses 13.4, 10.8 and 3.9 points. A control that undoes our first-item rule shows that rule to be the largest single class of loss on the Gemma models at point estimate, and inert on Llama; the registered test of whether the rule is the majority of the harm was inconclusive on the Gemma models. The remaining loss is concentrated where the planner actually splits the request (a post-hoc cut). The one repair confirmed under preregistration, a further call that sees the whole request and the split's calls and corrects them, is confirmed on Gemma-3-27B only; its Gemma-3-12B reading met the registered criterion but hinges on single tasks, so we do not claim it, and on Llama it was not met. It is not shown to need the split: the split's calls add at most two points to it on the Gemma models by unadjusted intervals, and at 12B the same call given nothing to correct is better by an interval wholly below zero (inconclusive on Llama). No repair that keeps the split is confirmed. Post hoc, the loss is planner items our rule drops, wrong function sets and argument errors, and at a large catalogue the loss grows, mostly because a planner shown only names and descriptions picks near-duplicate tools. Except where marked, the rest is descriptive or post hoc; the control-adjusted loss at 27B does not replicate on our earlier test split; the four arms the claims rest on ran under three prompt wordings on all three models, where the control-adjusted loss keeps its sign except at 12B, whose monolithic prompt loses 11.3 points to one rewording. Scope: one-shot text emission, one benchmark family, 1 to 4 functions per task; our registered run inside a native tool-calling API came out refuted and was discarded afterwards for corrupted payloads.

1 INTRODUCTION

Orchestrator-worker systems hand each tool family to a separate specialist: multi-agent frameworks route a request to tool-scoped agents (Wu et al., 2023) and tool protocols group tools into one family per server (Anthropic, 2024), a unit such systems can scope a sub-agent to. Deployed specialists usually run as tool loops with execution feedback; the operator studied here is a one-shot stylization of that, chosen so that every lost call is attributable (§3). Its single-context alternative is native parallel tool calling, in which one model emits every call at once (OpenAI, 2024; Patil et al., 2024a); we emulate both sides with text-prompted JSON emission (§3); our registered run inside that interface came out refuted and was discarded afterwards for corrupted payloads (§6). Whether splitting helps is contested (Cemri et al., 2025; Tran & Kiela, 2026; Jwalapuram et al., 2026). Most of this literature measures *whether* a split hurts. This paper is about what restores the loss when it does and, post hoc, *where* it comes from.

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.

The split we study has a planner assign each needed function to a same-model worker, each worker emit that function's calls from a planner-written subtask, and the lists concatenated; on BFCL v4 multi-function and multi-call tasks it loses 4 to 13 points against one monolithic call, mostly on tasks that need several functions with repeated invocations, the rest where the planner selects a function the request does not need. The obvious explanations pull in opposite directions. *Information loss*: a worker sees one schema and a planner-written subtask, never the request. *Coordination loss*: the workers emit in parallel with no shared count of how many calls the request needs. Each licenses a different fix, so we registered arms that test both, with controls. Figure 1 shows the operator and every arm's accuracy and cost.

Contributions. Triples are Gemma-12B, Gemma-27B and Llama-70B in that order; arm names are in small capitals and defined in §3.

(1) The cost of the split. The registered split loses 13.4, 10.8 and 3.9 points. Against a merge control that undoes our first-item rule it loses 7.1, 3.9 and 3.9. That is a descriptive figure; its 27B value does not replicate on the earlier split; on Llama the merge changes no execution's outcome, so the figure is the raw gap there; under two rewordings of every prompt it is -0.8 and 3.0 at 12B, 6.8 and 6.7 at 27B and 3.6 and 3.8 on Llama (Appendix A). The registered test of whether the rule is the majority of the harm (G0) is inconclusive on the Gemma models. Where the planner actually splits the request the merge-controlled loss is roughly three times the pooled figure on all three models, a post-hoc cut on the planner's own output whose 27B and Llama values are not separated from zero on the earlier split (§3, §4).

(2) The registered answer on repair, and the control that reads it. Eleven paired arms, three models, sealed sets, readings fixed before the single test pass. The one confirmed repair, SPLIT+PASS, a further call that sees the request and the split's calls and corrects them, recovers the majority of the gap at 27B; at 12B the reading met the registered criterion but hinges on single tasks, so we do not claim it; on Llama it was not met. The split's calls add at most $\delta{=}2$ points to it on the Gemma models by unadjusted intervals, and at 12B the same call given nothing to correct, ONE CALL, is better by an interval wholly below zero (inconclusive on Llama), so the split's candidates are not what repairs. That empty-candidate call is the control any claim about an aggregator, verifier or judge placed after workers needs, and here it removes the paper's own positive result. No repair that keeps the split is confirmed: full-context workers and the invocation contract, with and without context, give nine registered readings, four negative and five inconclusive, and the merge control's own reading (G0) is inconclusive on the Gemma models (on Llama the merge is inert) (§4).

(3) Where the hand-off loses, post hoc. A committed classifier over every execution finds planner items our rule drops (the largest class on Gemma at point estimate), a wrong emitted function set, and argument errors in correctly counted calls. Each repair imports a fault of its own: workers given the full context replace under-emission with foreign and byte-identical duplicate calls, and a count contract fails whenever the planner miscounts (§5).

(4) Where the rule stops. Padded to 64 functions the loss grows, mostly because a planner shown only names and descriptions picks a near-duplicate distractor; given the schemas the monolithic call already has, its loss against the merge control is not separated from zero (unregistered) at about the monolithic call's prompt tokens. Our rule, do not partition function calling across workers for accuracy, is therefore scoped to partitioning a request, not to selecting tools before one call (§6).

The operator is a stylization: one-shot emission, same-model workers, no tool loop, 1 to 4 functions per task, text-prompted JSON. We chose it because every lost call is then attributable to one boundary, not because systems emit this way. The anatomy above is post hoc; the large-catalogue arm and the native anchor are registered but read descriptively, and the full-schema planner and its merge control are not registered at all. The registered expectations that went unmet are listed in Appendix B.

An earlier confirmatory pass of this study was retracted for harness defects and rebuilt: every fix and every new arm was pre-declared in a registration update, fresh test sets were sealed, and one pass was run (Appendix B). The defect that shaped the design, our registered first-item-wins merge discarding the planner's repeated items, is measured here as an *arm* rather than assumed away.

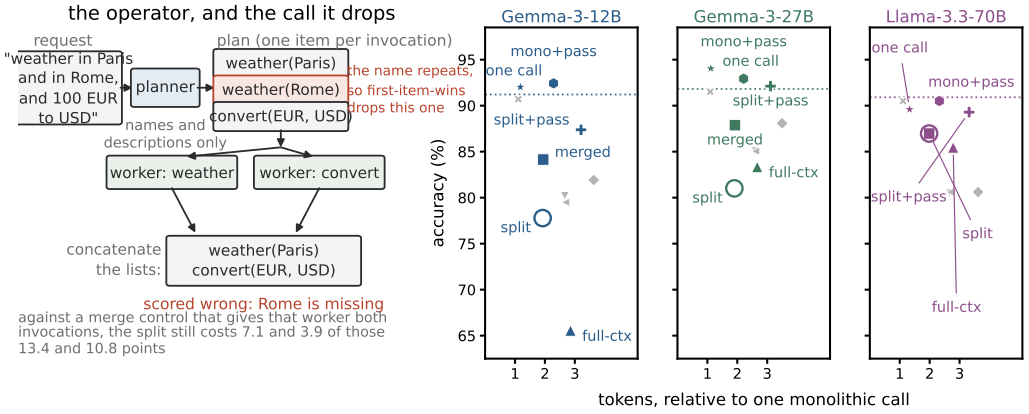

Figure 1: **Left:** the registered operator, schematically, on a request of the shape this benchmark ships (a real execution, with the model's own text, is in Figure 3). The planner is asked to use each function name at most once; when it writes one item per invocation anyway, our first-item-wins rule keeps only the first item per function and the concatenated list is missing a call the request asked for. Workers see one schema and their subtask, except at fan-out 1, where the worker is handed the request. Every other split arm is one edit to this picture; the single-call arms replace it. **Right:** one panel per model, every arm of Table 1 on BFCL fresh as accuracy against tokens relative to that model's monolithic call (uncached totals; Table 24 prices a cached catalogue), whose accuracy is the dotted line; the arms the claims rest on are named, the rest are grey. On every model the split sits below the line; on the Gemma models the merge control sits between, and on Llama it coincides with the split because the merge is a no-op there. The arms that clear the line on more than one model, ONE CALL and MONO+PASS, do so on both Gemma models and are not splits; the one split arm that clears any line is SPLIT+PASS at 27B, at 3.1× the tokens and with an interval covering zero (`og2v2_confirmatory_fresh_eval.json`; `make_og2v2_operator_figure.py`).

2 RELATED WORK

Does splitting help? MAST (Cemri et al., 2025) taxonomizes multi-agent failures; our symptoms fall under its disobeyed task and role specifications (under-emission, cross-worker leakage), but the mechanism, a deterministic merge discarding the planner's per-invocation items under a concurrent fan-out, is not named there. Tran & Kiela (2026) argue from the data-processing inequality that a hand-off can only lose information and predict single-agent parity under equal budgets; our full-context arms test whether restoring the information restores the outcome, which the registered arm does not show and a post-hoc filtered one shows in part on the Gemma models (§4). Jwalapuram et al. (2026) report architectural bloat in auto-generated systems (cost without benefit) but no mechanism. Prajapati & Mohite (2026) find that two calls beat a five-agent pipeline for small local models on math and code; Zhang (2026) that a holistic judge beats decompose-then-verify on completeness; Le Bronnec & Yokota (2026) measure the validated coverage of candidate sets; sequential splits gain on long inputs (Zhang et al., 2024). Kaliyev & Maryanskyy (2026) put the paired noise floor of coordination gains near 15 points.

Parallel function calling. LLMCompiler (Kim et al., 2024) and ReWOO (Xu et al., 2023) plan the whole call graph in *one* call before executing, so the failure we study cannot arise, and it is the confound a planner-side contract must be designed against: a planner that names every invocation has nearly written the answer. Skeleton-of-Thought (Ning et al., 2024) and planner-assigned semantic regions for parallel ideation (Ibrahim et al., 2026) coordinate parallel generations through one planning call. Decomposed prompting (Khot et al., 2023) and skill-level decompose-retrieve-compose routing (Gao, 2026) are related decompositions at the level of skills rather than invocations. Planner-plus-executor tool systems (Shen et al., 2023), tool retrieval ahead of one call (Qin et al., 2024; Patil et al., 2024b; Du et al., 2024) and role-decomposed agents that help small open models *after fine-tuning each role* (Shen et al., 2024) select or train where we partition a prompted model; our fan-out-1 case is retrieve-then-call (§6). Two of our channels are known single-call failures in-

side a multi-agent system: over-selection is irrelevance detection, which Patil et al. (2024a) scores as its own category, and whether a final call should see the workers' candidates is the anchoring question asked of judges and aggregators. Monitor-and-rerun repair (Dubey, 2026) detects drift and re-runs a step; we show that deterministic merges alone cannot repair (§5) and quantify what the extra call needs. Cardinality instructions are a known instruction-following target (Zhou et al., 2023); constrained decoding can enforce them but may suppress tool calls in open-weight models (Li et al., 2026), so our contract is prompt-level and reports count compliance separately.

Self-correction and controllers. Self-Refine (Madaan et al., 2023) and CRITIC (Gou et al., 2024) improve outputs with feedback, and intrinsic self-correction is often weak (Huang et al., 2024); our full-context pass is not self-correction, since its candidate list can be empty. Per-query orchestration controllers (Yue et al., 2025; Zhang et al., 2025; Su et al., 2026; Cui et al., 2026) are out of scope; a companion study (anonymized) reports a preregistered null for per-instance gating.

3 REGISTERED DESIGN

Reading key. Every triple in this paper is Gemma-12B, Gemma-27B, Llama-70B. The arms are MONO (one monolithic call), SPLIT (the operator), MERGED (the split under a merge control that keeps every repeated item), SPLIT+PASS and MONO+PASS (a correction pass appended to either), ONE CALL (that pass given no calls to correct), FULL-CTX (workers given the request, every schema and their assignment instead of a subtask), CONTRACT, CONTRACT+RC and CONTRACT+CTX (workers told how many calls to emit; +rc adds a count re-ask, +ctx replaces the subtask with the request and every schema and filters calls to non-assigned functions), CHECKLIST (MONO under the pass's rules); Table 7 defines each.

Task and operator. A unit is one BFCL v4 (Patil et al., 2024a) single-turn request with its tool schemas, scored by a pinned schema-aware exact-match checker (our re-implementation of the official AST checker, unit-tested against the port it was taken from; absolute scores are not leaderboard-comparable, paired differences are). Calls are emitted as JSON between tags in the model's text, not through a native tool-calling API, which §6 anchors on two models. The registered split (SPLIT) is: a planner sees the function catalogue and the request and emits a JSON list of $\{$function, subtask$\}$; one worker per selected function sees that function's schema and its subtask (a single-function plan hands the worker the full request) and emits that function's calls; the lists are concatenated. That last rule matters for scope: 64 to 67% of BFCL-fresh executions have fan-out 1 (the planner launches one worker), including every *parallel* execution, so the operator is a genuine split mainly on parallel_multiple and wherever the planner selects wrongly. Over every execution with fan-out ≥ 2 the loss is 37.0, 32.2 and 11.5 points, and 19.7, 13.3 and 11.5 against the merge control, both conditioning on the split planner's own output (paired over tasks, the unit every interval here uses; Appendix E). The aggregation rule is first-item-wins: a repeated function name keeps its first item. The harness executes workers sequentially; latency is reported as a counterfactual (planner plus the slowest worker) from per-call timings.

Arms (eleven, Table 7; paired on identical (task, sample) pairs; $k{=}5$ samples per arm at temperature 0.7; planner calls shared within a pair). MONO: one call with every schema and the request. SPLIT: the registered split. MERGED: the same planner call and worker prompt with repeated-name items merged into one subtask, one line per item. This is the control that isolates the aggregation rule. FULL-CTX: same plan, every worker sees the request and every schema and must emit only its function's calls. CONTRACT: a manifest planner emits, per function, n_{calls} and a numbered roster of the invocation cases in the request's words; each worker gets its schema, subtask and roster and must emit exactly n_{calls} calls; CONTRACT+RC adds one count re-ask (descriptive); CONTRACT+CTX gives the contract workers the full request, every schema and the assignment, and drops calls to non-assigned functions. SPLIT+PASS: SPLIT followed by one full-context pass that sees the schemas, the request and the candidate calls and outputs a corrected list (add missing invocations, remove duplicates; fail-open to the candidates); MONO+PASS: the same pass on the monolithic output; ONE CALL: the pass prompt with an *empty* candidate list, the single-call comparator matched to the pass call's compute; CHECKLIST: the monolithic prompt plus the pass's three coverage rules. Prompts are in Appendix A.

Data. Primary: **BFCL fresh**, the 66 curated tasks per category (*multiple*, *parallel*, *parallel_multiple*) at sha-prefix positions 135 to 200, 198 tasks never executed under any arm of this or the companion

Table 1: Expected accuracy per arm on BFCL fresh (n=198, k=5, one pass); gap in points.

Model	gap	gap vs merged	mono	split	merged	full-ctx	contract	contract +rc	contract +ctx	split +pass	one call	checklist	mono +pass
Gemma-12B	13.4	7.1	0.912	0.778	0.841	0.656	0.802	0.795	0.819	0.874	0.920	0.907	0.924
Gemma-27B	10.8	3.9	0.918	0.810	0.879	0.833	0.851	0.851	0.881	0.921	0.940	0.915	0.929
Llama-70B	3.9	3.9	0.909	0.870	0.870	0.855	0.805	0.806	0.806	0.893	0.896	0.905	0.905

study, sealed by committed manifest before the update. Replications: the earlier sealed BFCL test split (166) and BFCL-Live test split (78 real-user requests), re-executed under the rebuilt harness. Both are a *second contact*: the first pass observed all four of its arms on them and the parser audit used their cached planner outputs, so they are not clean (Appendix B). **Live fresh**, 120 further *live_multiple* tasks (single-call routing; the parallel live categories are exhausted), is clean. Train splits (162 curated, 55 live) served every pilot; calibration splits were not used.

Harness and provenance. Planner ceiling 512 tokens (768 for the manifest), lenient JSON parsing with one corrective retry, fail-closed after (no empty split plan in any test cell; the manifest planner failed closed on one 12B task, $5/990$ executions); transport errors re-asked; the serving provider recorded per call and pinned for the test pass (deepinfra for the Gemma models; Together's FP8 build for Llama, which is what "Llama-3.3-70B" means here); no transport error in any test artifact. The pass's fail-open rate is 0 to 1.9% per arm and cell. Arms that build a byte-identical prompt share a cached completion (common random numbers; Appendix D).

Hypotheses. All readouts are paired t-intervals (95%) on per-task expected accuracy, sign-flip p as sensitivity; gap $=$ MONO $-$ SPLIT. Family A: **G1** (SPLIT+PASS $-$ SPLIT) $-\frac{1}{2}$ gap > 0 (the pass recovers the majority); **G2** its gain on split output exceeds its gain on monolithic output (DiD > 0); **G5** (FULL-CTX $-$ SPLIT) $-\frac{1}{2}$ gap, two-sided (lower bound > 0: the arm recovers the majority; upper bound < 0: it does not; otherwise inconclusive); **G6** SPLIT+PASS $-$ ONE CALL, two-sided: candidates add value if the lower bound is > 0, add at most δ=2 points if the upper bound is < 2, otherwise inconclusive. Family B (added with the constructive arms): **G0** merged, **G7** contract, **G8** contract+ctx, each (arm $-$ SPLIT) $-\frac{1}{2}$ gap with the G5 ladder; **G9** CHECKLIST $-$ MONO, no material effect if the CI lies inside $(-2, 2)$. **G3** (audited-space gating) went to the companion study and **G4**, what the pass returns per extra token, is descriptive. A later update adds three hypotheses, read outside the families: **H10**, a fourth model's planner misses fewer functions shown schemas than names; **H11**, that model's arms under the registered scorer; **H12**, the split's loss survives inside the provider's tool-calling API, on the G6 ladder. Holm within model and family on BFCL fresh (24 registered tests; no correction across models); replications reported with CIs outside the families. *Deviation disclosed:* the registration adjusted G1 and G2 jointly across models; a later update, written after the first pass but before any sealed v2 execution, made per-model families of four, and we report 12B's G1 under both (Appendix B). Cells with a gap below 3 points are descriptive; post hoc that threshold is too permissive (power 0.17 to 1.00 on the registered cells, Appendix E). The contract's attribution rule and its proxy are a disclosed deviation (Appendix B).

Timeline and disclosure. Registration, seven updates and pilots (train only) are anonymized for review (view-only links in the Reproducibility statement); the earlier pass, its defects and every number it produced remain in the record (Appendix B). Each new arm was piloted on train, its numbers and its one prompt revision disclosed in the update, and its prompt frozen before posting. Eleven deviations (among them a dropped hypothesis, G3) and ten expectations that went partly unmet are listed in Appendix B. Updates 3 to 7 were designed after the sealed set had been analysed: for them "registered" means the design was frozen before the run, not that the data were unseen. Update 7 fixes the two cells of Appendix A and Table 28, their reading rules, the equivalence margin and a native probe's payload criterion; its text was committed 86 seconds before the first call of those cells and posted afterwards, so for it "registered" means a timestamped commit, not a prior posting. The pass was interrupted once and resumed from the cache (Appendix B).

4 RESULTS ON THE FRESH SEALED SET

Table 2 carries every registered contrast with its ladder verdict. Established: the size of the loss; that a pass appended to the split recovers the majority of the gap at 27B; and, by unadjusted intervals,

Table 2: Registered contrasts on BFCL fresh (points; paired t 95% CI; raw and Holm p within model and family; reading by the registered ladder). Recovery contrasts $(\text{arm} - \text{SPLIT}) - \frac{1}{2}(\text{MONO} - \text{SPLIT})$ are positive when the arm recovers more than half of the gap. ‡not met after Holm (unadjusted CI excludes 0); §interval wholly below zero, an unregistered reading. 12B's G1 row reads met by the ladder; the text declines the claim because 17 single tasks are pivotal. Llama's G0 row is $-\frac{1}{2}$gap by construction (the merge returns the split's outcome on every execution).

Contrast	Model	Estimate (95% CI)	p	Holm p	Registered reading
G1 pass > half-gap	Gemma-12B	$+2.9$ ($+0.2$, $+5.6$)	0.036	0.036	met
	Gemma-27B	$+5.7$ ($+3.0$, $+8.5$)	6.1×10^{-5}	1.8×10^{-4}	met
	Llama-70B	$+0.4$ (-1.4, $+2.2$)	0.699	1.0	not met
G2 DiD (split-specific gain)	Gemma-12B	$+8.4$ ($+4.4$, $+12.3$)	4.2×10^{-5}	1.3×10^{-4}	met
	Gemma-27B	$+10.0$ ($+5.5$, $+14.5$)	2.0×10^{-5}	8.2×10^{-5}	met
	Llama-70B	$+2.7$ ($+0.1$, $+5.3$)	0.041	0.123	not met after Holm‡
G5 full-ctx > half-gap	Gemma-12B	-18.9 (-24.7, -13.2)	8.2×10^{-10}	3.3×10^{-9}	does not recover
	Gemma-27B	-3.1 (-7.2, $+1.0$)	0.141	0.282	inconclusive
	Llama-70B	-3.5 (-6.0, -1.0)	0.007	0.027	does not recover
G6 split+pass − one call	Gemma-12B	-4.6 (-8.3, -1.0)	0.012	0.024	adds at most δ=2§
	Gemma-27B	-1.9 (-4.8, $+1.0$)	0.189	0.282	adds at most δ=2
	Llama-70B	-0.3 (-2.8, $+2.2$)	0.814	1.0	inconclusive
G0 merged > half-gap	Gemma-12B	-0.4 (-2.9, $+2.2$)	0.787	0.787	inconclusive
	Gemma-27B	$+1.5$ (-1.2, $+4.2$)	0.284	1.0	inconclusive
	Llama-70B	-2.0 (-3.5, -0.5)	0.011	0.022	does not recover
G7 contract > half-gap	Gemma-12B	-4.3 (-8.9, $+0.3$)	0.065	0.260	inconclusive
	Gemma-27B	-1.4 (-5.3, $+2.5$)	0.491	1.0	inconclusive
	Llama-70B	-8.4 (-12.2, -4.7)	1.3×10^{-5}	5.4×10^{-5}	does not recover
G8 contract+ctx > half-gap	Gemma-12B	-2.6 (-7.3, $+2.2$)	0.284	0.678	inconclusive
	Gemma-27B	$+1.7$ (-1.9, $+5.2$)	0.356	1.0	inconclusive
	Llama-70B	-8.3 (-12.2, -4.5)	3.1×10^{-5}	9.4×10^{-5}	does not recover
G9 checklist − mono	Gemma-12B	-0.5 (-1.3, $+0.3$)	0.226	0.678	no material effect
	Gemma-27B	-0.3 (-1.7, $+1.1$)	0.663	1.0	no material effect
	Llama-70B	-0.4 (-1.4, $+0.6$)	0.416	0.416	no material effect

that the pass's candidates add at most δ on the Gemma models (inconclusive on Llama), at 12B with an interval wholly below zero. Not established: that any repair inside the split works (four readings negative, five inconclusive), and that our own aggregation rule is the majority of the harm (G0 inconclusive on the Gemma models), though the merge control that undoes the rule is the figure we cost the operator at throughout.

The gap, and what is left of it once our own rule is undone. Against the merge control that keeps the planner's repeated items, the split costs 7.1, 3.9 and 3.9 points (Gemma-12B, Gemma-27B, Llama-70B; Table 1), which is the figure we cost the operator at throughout; the raw gap, which also carries what our own first-item-wins rule discards, is 13.4, 10.8 and 3.9. The loss sits in *parallel_multiple* (0.909 → 0.591, 0.888 → 0.618, 0.848 → 0.752) and *multiple* (0.912 → 0.836, 0.918 → 0.867, 0.933 → 0.894); there is none on *parallel* (Appendix C). A third of the curated set ships one function and two thirds of executions have fan-out 1, so both figures are diluted by tasks the operator never splits: on the 62 tasks whose ground truth needs more than one distinct function the merge-controlled loss is 11.9 points (CI 3.2 to 20.7), 4.5 (not separated from zero, CI -4.8 to $+13.9$) and 7.4 (CI 0.9 to 13.9), the last of these a raw gap because Llama's merge is inert (`og2v2_v26_addenda.py`).

The pass (G1, G2, G6). SPLIT+PASS recovers 71%, 103% and 59% of the gap. G1 is met for 27B under every multiplicity scheme we ran ($+5.7$, CI $+3.0$ to $+8.5$; Holm 2×10^{-4} within the registered family, 9×10^{-4} correcting across all four sets), and no single task is pivotal there (0 of the 42 tasks whose outcome differs between the arms). At 12B it is met only in the registered family ($+2.9$, CI $+0.2$ to $+5.6$; Holm 0.036, against 0.109 under the original six-test family and 0.217 in the released twelve-cell report), and removing any one of 17 of those 42 tasks makes the CI include 0, so we do not claim it. It is not met for Llama ($+0.4$, CI -1.4 to $+2.2$). The pass's gain on split output exceeds its gain on monolithic output by 8.4 and 10.0 points (G2, Gemma) and 2.7 on Llama (CI excludes 0, Holm 0.12); G2 is ceiling-confounded (22 points of headroom on split output, 9 on monolithic) and, with G6, shows only that the pass edits split output. SPLIT+PASS

Table 3: The empty-candidate control on nine readings: the pass over three candidate sources minus the same call with no candidates (points, paired 95% CI over tasks; *interval excludes zero). The merge-control row is a cache bound, scored where its prompt is already cached and credited with a success everywhere else, so the true value is at or below the figure shown, against ONE CALL's level. Only the first row is registered (G6; §interval wholly below zero, our reading) (`og2v2_controls_table.py`).

Candidates handed to the pass	Gemma-12B	Gemma-27B	Llama-70B
from the split (G6, registered)	-4.6 (-8.3, -1.0)*	-1.9 (-4.8, $+1.0$)	-0.3 (-2.8, $+2.2$)
registered reading	at most δ§	at most δ	inconclusive
from the monolithic call (unregistered)	$+0.4$ (-2.1, $+2.9$)	-1.1 (-3.1, $+0.8$)	$+0.9$ (-1.4, $+3.2$)
from the merge control (cache bound)	$\leq$ 0.890 vs 0.920 (924/990)	$\leq$ 0.924 vs 0.940 (939/990)	$\leq$ 0.888 vs 0.896 (987/990)
ONE CALL − MONO (descriptive)	$+0.8$ (-1.8, $+3.4$)	$+2.2$ ($+0.1$, $+4.4$)*	-1.3 (-3.8, $+1.2$)

does not reach MONO at 12B: the difference is -3.8 points (CI -7.2 to -0.5), $+0.3$ at 27B, -1.6 on Llama. The candidate calls never separate above the empty call: SPLIT+PASS minus ONE CALL is -4.6 points at 12B (G6, CI -8.3 to -1.0; the registered reading is "adds at most δ", and harm is our reading of an interval below zero); at 27B the candidates add at most δ (-1.9, upper bound $+1.0 < \delta$), and on Llama the reading is inconclusive (-0.3, upper bound $+2.2 > \delta$; at most δ under a delimiter-aware parser, Table 30). The same single call sits at $+0.8$, $+2.2$ and -1.3 points relative to MONO, only the 27B interval excluding zero. CHECKLIST itself is within a point of MONO on all three models (G9). Table 3 asks G6's question of three candidate sources: across the nine readings the pass never separates above the call that saw no candidates; seven are at or below it, the two above it (the monolithic call's candidates at 12B and on Llama) have intervals covering zero, and the merge control's candidates, a cache upper bound, sit below it on every model (Appendix E.1). Rerun under two further wordings on all three models, the same contrast is at or below the empty-candidate call in 8 of 9 model-wording cells; the exception, Llama under wording A, is the fail-open convention scoring the 97 of 990 pass outputs the parser cannot read as the unchanged empty candidate list, and over the executions where both arms' output parses it reads -2.1 (CI -4.2 to $+0.1$), so under that reading all nine are at or below the empty-candidate call (Appendix A).

Aggregation rule (G0). Merging the planner's repeated-name items recovers (Table 12) 47% (bootstrap 28 to 67%) and 64% (40 to 98%) of the Gemma gaps and 0% of Llama's: $+6.4$ and $+6.9$ points over the split. Under the registered ladder the Gemma contrasts are *inconclusive* (-0.4 and $+1.5$ points, CIs cover 0), so the pre-declared expectation of a positive lower bound, "most of the harm was the rule", was not met; Llama's reads *does not recover* (-2.0, CI -3.5 to -0.5). Repeated items occur in 41%/40%/18% of plans. Llama's sit almost entirely in the *parallel* category, where the plan names one function and that worker is handed the full request; on Llama the merge changes no execution's outcome, so on its fresh cell and every live cell MERGED equals SPLIT and G0 is $-\frac{1}{2}$gap by construction. Removing the rule does not close the gap: the merge-controlled figures above are descriptive contrasts with CIs -10.8 to -3.4, -7.7 to -0.2 and -7.0 to -0.9, and on the earlier test split they are -5.2, -1.2 (CI -5.2 to $+2.8$) and -2.5, so the 27B figure does not replicate. They also locate the model difference: Llama's split arm already is a merge arm, and against the merge control it is not separated from Gemma-27B (CI -3.9 to $+3.9$).

Information (G5). Full-context workers recover -91%, 21% and -38% of the gap. The registered contrast is conclusively below the half-gap for Gemma-12B (-18.9, CI -24.7 to -13.2) and Llama (-3.5, CI -6.0 to -1.0) and *inconclusive* for 27B (-3.1, CI -7.2 to $+1.0$). Post hoc, two zero-LLM filters change the reading: dropping calls to non-assigned functions and then exact duplicate calls raises the arm to 0.884/0.880/0.864, which on the registered scale is $+3.9$ at 12B (CI $+0.6$ to $+7.2$; 79%), $+1.6$ at 27B (CI -1.7 to $+4.9$; 64%) and -2.6 on Llama (CI -4.9 to -0.3). Both filters were built after the failure mode was seen, so the filtered arm is exploratory (Appendix E). The registered negative stands for the arm as registered, which lacked its sibling's hygiene; the reading that *information* does not repair the split does not. Even filtered, the arm stays below MONO at 2.7 to 2.9× the tokens, and the 12B reading does not replicate on the other three sets (Appendix E.1).

Invocation contract (G7, G8). Declared counts match the ground truth per function in 91 to 94% of executions and workers emit exactly the declared number in 98 to 100%, yet the arm recovers 18%, 37% and -164% of the gap; with full context added, 31%, 65% and -162%. Under the ladder, Llama's contrasts are conclusively negative (-8.4 and -8.3; the contract is 6.5 points below the split

and 10.4 below MONO); all four Gemma contrasts are *inconclusive*, and at 27B CONTRACT+CTX is within 0.2 points of the merge control (0.881 vs. 0.879; on parallel_multiple 0.852, the best split arm without a pass). Conditioning on whether the planner counted right (post hoc, a cut selected by the treatment arm's own output) locates much of it: on the miscounted executions the arm scores at or near zero; on the rest it is inconclusive against MONO rather than at parity on the Gemma models and below it on Llama, and a further cut by a proxy for our roster rule goes against us; the count re-ask changes nothing (Appendix E.1).

Replications. The earlier test split reproduces the direction of every non-null fresh contrast (Appendix D): gaps 11.0/8.6/3.5, pass 81%/113%/76%, contract −9%/41%/−114%. The real-user sets differ in kind. On Live test the pass recovers 100%/91% of the Gemma gaps (6.9, 8.2 points; CIs cover the half-gap) and Llama has no gap (−1.3, descriptive). On Live fresh (single-call routing tasks) the loss is 15.8, 15.0 and 3.2 points and is *planner over-selection*: the fan-out-≥ 2 executions score at or near zero because the extra worker emits a call rather than the empty list it was told to return (Appendix E.1).

5 ANATOMY OF THE LOSS

All statements here are post hoc and carry no registered reading; they are computed on the fresh set by committed scripts, and the train pilots agree.

Three kinds of loss event. Classifying every execution where the monolithic call is right and the split wrong (Table 35; sub-classes in Appendix E) gives three classes. *Dropped items*: 43%/49%/0% of loss events (6.4/6.4/0 points) by the conservative rule (the merge control is right on the same execution); 52%/53%/0% carry the signature (a repeated-name plan and an under-emitting split), the difference being merged re-samples that failed on an argument, so the rule is a lower bound. *Wrong emitted function set*: 22%/22%/43% (3.2/2.8/2.3 points), mostly a planner superset whose extra worker emitted a call. *Residual*: 34%/29%/57% (5.1/3.7/3.0 points), mostly argument errors in correctly counted calls. These are gross shares: netted against MONO's own failures, only the count channel on the Gemma models and the wrong function set at 12B and on Llama exclude zero, and the argument channel does on no model, though against the merge control it is the largest part of the 27B residual at point estimate (sub-class counts and the netting in Appendix E.2). Figure 3 shows the first kind.

One call per handed case. Among function-slots that need ≥ 2 invocations on parallel_multiple, split workers emit exactly one call in 82%/78%/5% of slots; after merging the dropped items, in 3%/3%/3%. Where the planner repeated a function's name the workers emit exactly one call in every genuinely split slot, and where it did not, in 19%, 15% and 3% of them, one slot in 32 on Llama (Appendix E): there is no one-call prior of the worker, it emits one call for each case it is handed, and the aggregation rule handed it one. No under-emission survives the merge; what remains is wrong function sets, argument errors in correctly counted calls and over-emission (Appendix E.2).

Over-selection. The planner's function set is exactly right in 89%/93%/97% of *multiple* executions, and when it is wrong it is, on the Gemma models, almost always a superset. A superset is fatal because the extra worker emits rather than abstains: those 31/20/5 fan-out-≥ 2 executions (9, 7 and 1 tasks) score 0.00/0.10/0.00 against MONO's 0.65/0.90/1.00. On Live fresh any fan-out is over-selection by construction and is the whole net loss (§4). The planner prompt induces it (Appendix E.2).

Information restores the cases and adds two degenerate emissions. Full-context workers get the repeat structure right and over-emit: at 12B most of their surplus slots are byte-identical copies of another worker's call and most of their failures leak a foreign function (Figure 3, lower panel; Appendix E.2). Neither needs an LLM to fix: the leak filter and exact-duplicate removal lift the arm to 0.884/0.880/0.864 (§4).

6 COST, LATENCY AND THE PRACTICAL RULE

Tokens and latency. On BFCL fresh the split costs 1.9 to 2.0× the monolithic call in tokens (1.0 to 1.1× on the real-user sets, whose tasks ship larger catalogues), SPLIT+PASS 3.1 to 3.3× and ONE CALL 1.1 to 1.3× (Appendix D). Counterfactual latency, the planner plus the slowest worker under

perfect parallelism and no overhead, is 1.6 to 2.2× the single call's at *every* fan-out and 1.7× at 64 functions (Appendix E). The rule that survives every curated cell is about accuracy: do not partition function calling across workers for it.

A larger catalogue. Every curated task ships 1 to 4 functions, and a practitioner splits because the catalogue is large. Padding every catalogue with distractors both arms see (one pass at 27B, own functions listed first; pre-declared in Update 5 but read descriptively, the test set having been opened, Appendix B), the loss grows from 10.8 points unpadded to 12.9 at 8 functions and 18.7 at 64, no interval covering zero, while MONO holds at 0.917 to 0.924 and the split's measured prompt-token ratio falls from 1.9× to 0.33× (Table 25). The merge control does not remove the growth (6.1 to 12.8). It is one channel, the planner missing a needed function (0.010 to 0.073 of executions), and 65 of the 72 misses at 64 name a distractor instead (17 tasks), nearly always a near-duplicate the scorer cannot credit: without those tasks the growth is +1.8 (CI −1.8 to +5.4), so what a deployment would see lies between (Table 26). That is our planner, not decomposition: it routes on names and descriptions. Given the same schemas it misses 0.011 and the raw loss falls to 10.2 at 1.07× the prompt tokens, with no detectable growth against the merge control, which was run for that planner afterwards (unregistered, Appendix B; Table 27). Under the same fan-out conditional we use above, where the operator is a split, that planner still loses 9.9 points (CI 2.5 to 17.2; `og2v2_v26_addenda.py`). The same cell was run on 12B and Llama (Update 7, whose text fixes the equivalence margin δ for the 64-function reading and was committed before the first call), and the 27B cell of Update 5 is read under the same rule afterwards: at 64 functions MERGED minus MONO is +1.1 (CI −2.6 to +4.8), −2.1 (CI −4.4 to +0.1) and −0.3 (CI −3.5 to +2.9), all inconclusive (Table 28, whose 8-function column applies the same margin after the fact). In these cells no split configuration beats the monolithic call, and the only one within δ of it is the full-schema planner with the merge control at 8 functions on Llama, the post-hoc reading. The selection-then-one-call subset is in Appendix E and cached-catalogue pricing in Table 24.

If you split anyway. (a) Keep one line per invocation through aggregation, free and worth 47% and 64% of the Gemma loss, nothing on Llama or the real-user sets; (b) with full context, drop non-assigned and duplicate calls, free but negative in four of twelve cells at 1.7 to 2.9× the tokens; (c) do not split: make one call that sees the request and every schema, the only one of the three positive in every cell against the split and separated from it in seven of twelve. Against calling the model once, its interval excludes zero in three of twelve and one of those is negative (+5.4 on Llama's Live test to −5.3 at 12B on Live fresh). That last figure is our audit prompt's fail-open convention, not the model (Appendix A), so end on the plain monolithic prompt.

Does the interface explain it? The anchors cannot say. Behind the provider's tool-calling API the 27B split loses 9.9 points (CI +5.1 to +14.7) against 10.8 in text, but neither 27B variant measures the operator: our registered variant (`required`) read *refuted* (+16.6) on a run we set aside by a payload criterion written after the reading, which does not look at accuracy: half the monolithic call's tool calls come back with an empty function name. The `auto` variant, which gives the 9.9, has a defect of its own (60 monolithic and 48 split executions of 990 returning no call) and was never registered, so H12 has no usable registered reading. In the one clean cell, Llama under `required`, split minus MONO is −1.4 (CI −4.7 to +1.8) against −3.9 in text, so the rule is neither supported nor contradicted there (Tables 31, 32). Our practical rule is therefore scoped to text emission.

7 DISCUSSION

Mechanism. The split loses at the context's boundaries: *what* is handed over, *to whom*, and *how a worker executes its slice* (§5). Each arm that moves one boundary recovers part of the Gemma loss and imports a fault of its own; on Llama none helps.

What is established. Under preregistration: the size and location of the loss; the pass recovering the majority at 27B under every multiplicity scheme we ran, with no pivotal task; neither registered repair confirmed (four of nine readings negative, five inconclusive); the candidates adding at most δ on both Gemma models; and, of the later hypotheses, H10. H11 is descriptive, H12 read refuted on a cell discarded afterwards, the 12B pass result is met but not claimed, and the anatomy is post hoc.

What is not. On the Gemma models the largest loss class is our own merge rule's, at point estimate, and on Llama it is inert. The gap is metric-dependent: under per-call F_1 the real-user gaps shrink

(Table 38) and the class ordering of §5 does not survive on Gemma (Appendix E). Small open planners do not reliably group repeated invocations when told to. Our scorer reads calls by bracket span, which costs Llama alone (Table 30); Llama is also the only other family; a fourth model's merge-controlled loss is null at one end (Table 29); and the claim-bearing arms ran under three wordings, the rest under one (Appendix A).

AI USE STATEMENT

We used generative AI tools (LLM-based assistants) extensively: to write experiment and analysis code, to draft and revise the manuscript, to run the forensic replays, and to generate simulated pre-submission reviews, whose requested corrections were verified against the artifacts before being applied. We did not use generative AI to fabricate, select or embellish results: every reported number is produced by the committed deterministic pipeline and regenerates from the released artifacts (one historical run-log count excepted, flagged in Appendix B), and the registered analyses and interpretation rules were fixed before the test pass. AI-assisted code is guarded by executable tests: canned-cache tests of every arm's execution path, which run from the supplement as it ships, and a scorer test that reproduces the published outcome counts of the earlier study this checker was ported from and, when that study's repository is present, agrees with it call by call. Those scorer tests read the earlier study's own seed record and repository, which the anonymous package cannot carry, so they report themselves skipped there with that reason, as do the five byte-identity comparisons of the generated tables, whose .tex files the package does not carry (eight skips; every other test passes offline). The LLMs under study are the experimental subjects, not authors. We take responsibility for the final content.

ETHICS STATEMENT

The work measures a failure mode of multi-agent LLM systems and whether it can be repaired, and quantifies when the split is unnecessary. We foresee no direct negative societal impact; honest negative results may reduce wasted compute.

REPRODUCIBILITY STATEMENT

Registration texts (with disclosed pilots and pre-declared criteria), sealed split manifests with SHA-256 digests, the harness (v1 and v2), every collection, forensic and analysis script, cached raw responses enabling offline re-scoring of every execution, provider provenance per call, and the table generators are released. The registration and its seven updates are open to reviewers as anonymized view-only links:

Registration, 20 Aug 2026:
`https://osf.io/tyq5m/overview?view_only=63afb5a21833402684d7366014aac59d`

Update 1, the within-family scale point:
`https://osf.io/9f5sw/overview?view_only=d679d5cc3b8c40b296276b48a6f9c2f0`

Update 2, harness v2, the fresh sealed sets and the new arms:
`https://osf.io/rqve8/overview?view_only=c2dc2a94173346c7a1246f3d72e0e0ba`

Update 3, the native tool-calling anchor, with its train pilot:
`https://osf.io/w32un/overview?view_only=b52d69944a8740799857deab8c3a5ef9`

Update 4, the split under that interface, with its train pilot:
`https://osf.io/h32ab/overview?view_only=86c208b748014abc9de67d90b7b09147`

Update 5, accuracy at a large function catalogue, with its train pilot:
`https://osf.io/br2u8/overview?view_only=1838d1f00f8a4aca9f679739f849af5b`

Update 6, a native anchor with resolution, a fourth model, and a correction:
`https://osf.io/kgqw7/overview?view_only=236fdde6b064424692aa9cacf94ca0f5`

Update 7, the wording control and the catalogue cell on the remaining models, posted after their runs (see below):
`https://osf.io/7mjqs/overview?view_only=e464b587efdf464a93b547ffdee2929f`

Every number in this paper regenerates from committed artifacts without network access, except one run-log count flagged in Appendix B, which also lists the v1 defects, the update chain and the interruption of the test pass.

REFERENCES

Anthropic. Model context protocol. https://modelcontextprotocol.io, 2024.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

Zhiqing Cui, Haotong Xie, Jiahao Yuan, Cheng Yang, Hanqing Wang, Yuxin Wu, Yifan Wu, Siru Zhong, Tao Yu, Yifu Guo, Siyu Zhang, Xinlei Yu, Qibing Ren, and Usman Naseem. Uno-orchestra: Parsimonious agent routing via selective delegation. *arXiv preprint arXiv:2605.05007*, 2026.

Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. In *International Conference on Machine Learning (ICML)*, 2024.

Sunny Dubey. Real-time detection and repair of LLM agent failures. *arXiv preprint arXiv:2608.02464*, 2026.

Xueping Gao. Compositional skill routing for LLM agents: Decompose, retrieve, and compose. *arXiv preprint arXiv:2606.18051*, 2026.

Zhibin Gou et al. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *Proceedings of ICLR*, 2024.

Jie Huang et al. Large language models cannot self-correct reasoning yet. In *Proceedings of ICLR*, 2024.

Fares Nabil Ibrahim, Nafis Saami Azad, and Raiyan Abdul Baten. Anchorless diversification for parallel LLM ideation. *arXiv preprint arXiv:2605.30150*, 2026.

Prathyusha Jwalapuram, Hehai Lin, Chuyuan Li, Fangkai Jiao, Sudong Wang, Yifei Ming, Zixuan Ke, Chengwei Qin, Giuseppe Carenini, and Shafiq Joty. The illusion of multi-agent advantage. *arXiv preprint arXiv:2606.13003*, 2026.

Alibek T. Kaliyev and Artem Maryanskyy. How much coordination gain is real? a paired noise-floor protocol for multi-agent LLM benchmarks. *arXiv preprint arXiv:2606.20695*, 2026.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations (ICLR)*, 2023.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *International Conference on Machine Learning (ICML)*, 2024.

Florian Le Bronnec and Rio Yokota. Evaluating multiple LLM generations with validated task coverage. *arXiv preprint arXiv:2608.24228*, 2026.

Fangzheng Li, Aimin Zhang, and Chen Lv. Constraint tax in open-weight LLMs: An empirical study of tool calling suppression under structured output constraints. *arXiv preprint arXiv:2606.25605*, 2026.

Aman Madaan et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting LLMs for efficient parallel generation. In *International Conference on Learning Representations (ICLR)*, 2024.

OpenAI. Function calling guide: parallel function calling. https://platform.openai.com/docs/guides/function-calling, 2024.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Berkeley function calling leaderboard. https://gorilla.cs.berkeley.edu/leaderboard.html, 2024a.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Ashish Prajapati and Om Mohite. Two calls beat five agents: Evaluating multi-agent pipelines against self-refinement for local language models. *arXiv preprint arXiv:2607.26922*, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small LLMs are weak tool learners: A multi-LLM agent. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Xinyuan Song, Lewei He, and Jingsong Yang. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference*, 2026. arXiv:2509.11079.

Dat Tran and Douwe Kiela. Single-agent LLMs outperform multi-agent systems on multi-hop reasoning under equal thinking token budgets. *arXiv preprint arXiv:2604.02460*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Yanwei Yue et al. Masrouter: Learning to route LLMs for multi-agent systems. In *Proceedings of ACL*, 2025. arXiv:2502.11133.

Guibin Zhang et al. Multi-agent architecture search via agentic supernet. In *Proceedings of ICML*, 2025. arXiv:2502.04180.

Xinran Zhang. Rethinking atomic decomposition for LLM judges: A prompt-controlled study of reference-grounded QA evaluation. *arXiv preprint arXiv:2603.28005*, 2026.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan O. Arik. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

HOW TO READ THE APPENDIX

The appendix is organised by the claims of the Introduction. Claim (1), the cost of the split: Table 1 and Table 2 for the registered figures, Table 36 for the fan-out conditional, Table 37 for what the registered ladder could resolve, Appendix C by category and Appendix D on the replication sets. Claim (2), repair and the empty-candidate control: Table 2, Table 12 for the recovery shares, and Appendix E.1 for the comparisons moved out of Section 4. Claim (3), the anatomy: Table 35, Figure 2, Figure 3, Table 33, and Appendix E.2 for the sub-class counts moved out of Section 5. Claim (4), the catalogue: Table 25, Table 26, Table 27 and Table 24. Scope: the native anchors in

Table 31 and Table 32, the fourth model in Table 29, the parser in Table 30 and the ONE CALL arm's empty parses in Table 34, the metric in Table 38 and Table 40, the prompts and their rewording in Appendix A. Provenance, the earlier pass and every deviation from the registration: Appendix B. Whether the aggregation assumption exists in public frameworks: Appendix G. The arm definitions are Table 7; protocol details are Appendix H.

A PROMPTS

The claim-bearing arms ran under three wordings and the rest under one, and wording alone moves single-call accuracy by up to 11.3 points on Gemma-3-12B, past the order of several contrasts in this paper: ONE CALL and CHECKLIST differ only in wording and by 1.3 and 2.5 points on the Gemma models (§4).

Prompt sensitivity (unregistered, run after the last review round). Two further wordings, A and B, carry every instruction element of the registered planner, worker and monolithic prompts and change only prose and, in B, the headings, with three wording-level deviations found on re-reading and listed in the module: both drop the worker's specialist persona, both narrow "do not answer the request" to a ban on prose, and B's worker adds "and nothing else" (`og/arms/paraphrase.py`, frozen before any call was made with them; `tests/test_paraphrase_prompts.py` pins each element against the phrasings a paraphrase may use for it and refuses a variant that keeps the registered wording). Gemma-3-27B then ran MONO and the merge control on BFCL-fresh under all three wordings, one draw each at $T = 0.7$: 1,394 new calls, no transport error. The merge-controlled loss is 4.0 points under the registered wording ($p = 0.074$), 8.1 under A ($p = 0.0001$) and 5.1 under B ($p = 0.033$), and the levels move less than the contrast does (1.5 and 2.5 points against 4.0). Five draws of the registered wording alone span 1.5 points of contrast (mean -3.9, the registered reading), so the spread across wordings is 2.7 times the spread one wording gives; paired over tasks, however, neither paraphrase's contrast separates from the registered one (-4.0, CI -8.9 to $+0.8$; -1.0, CI -5.5 to $+3.5$). Fan-out is unchanged (1.53 under each wording), no plan came back empty, and the paraphrases needed fewer lenient parses than the registered wording (1 under A and 0 under B, against 3), so the larger losses are not malformed output. We report both readings. The sign of the operator's cost survives rewording, and of the three wordings the registered one is the most favourable to the operator, so the headline reading is the conservative one; but the magnitude is wording-sensitive at the scale of the effect itself, and nothing here separates a different wording from a worse one. Variant A, which raises MONO by 1.5 points while lowering the merge control by 2.5, is consistent with either. Generated by `scripts/og2v2_prompt_sensitivity.py` into `results/og2v2_prompt_sensitivity.json`.

The same check at k=5, on the four arms the claims rest on (unregistered, run on 24 September). The pass prompt was reworded too, element for element (the module lists its elements), and every wording ran five draws per task on MONO, the merge control, SPLIT+PASS and ONE CALL: 9,857 new calls, no transport error, the registered wording replaying from the committed cache. Under the registered wording, A and B the merge-controlled loss is 3.9 (CI 0.2 to 7.7), 6.8 (3.2 to 10.4) and 6.7 (3.1 to 10.2) points, separated from zero under all three; SPLIT+PASS sits at $+0.3$, -0.6 and -0.3 from MONO, within a point of it throughout; and ONE CALL is $+2.2$ (CI $+0.1$ to $+4.4$), $+0.5$ and $+0.2$ above MONO, so its edge over the plain monolithic call at 27B belongs to the registered wording. The merge-controlled loss moves by 2.8 points across the three wordings, about as much as five draws of one wording move it (1.5 under the registered wording, 3.5 under A and B), and paired over tasks A does not separate from the registered wording (-2.8, CI -6.1 to $+0.4$) while B stops at an upper bound of $+0.0$ (-2.7, CI -5.5 to $+0.0$), a near-crossing rather than a clean non-separation; the other three contrasts move by 0.9 to 2.0 points. The G6 reading, at most δ, holds under the registered wording and A (upper bounds $+1.0$ and $+1.7$) and is inconclusive under B ($+2.1$). The registered wording is again the most favourable of the three to the operator. The raw split, with our first-item-wins rule, loses 10.8, 13.6 and 14.1 under the three wordings, so the merge recovers 64%, 50% and 53% of it rather than the registered wording's share alone; the raw figure is the one contrast whose paraphrase separates from the registered wording paired over tasks (B: -3.3, CI -6.2 to -0.5; A: -2.8, CI -6.0 to $+0.4$), which is a statement about our rule's interaction with the wording, not about the merge-controlled operator. Reading: the sign of the operator's cost does not depend on the wording, its size moves by about as much as five draws of one wording move it, and the one number that was a wording effect is the empty-candidate call's

edge over MONO (Table 5). Generated by `scripts/og2v2_prompt_sensitivity_k5.py` into `results/og2v2_prompt_sensitivity_k5.json`.

The same check on Gemma-3-12B (Update 7, whose text was committed before the first call and posted afterwards; run on 24 and 25 September; unregistered, sealed set long open). 11,202 new calls, no transport error. Under the registered wording, A and B the merge-controlled loss is 7.1, −0.8 and 3.0 points (CIs +3.4 to +10.8, −5.6 to +4.0, −0.6 to +6.7), so the sign does not survive rewording on this model: under A the merge control is not below the monolithic call. Paired over tasks, A separates from the registered wording (+7.9 (CI +2.4 to +13.4)) and B separates (+4.0 (CI +0.7 to +7.3)). The raw split loses 13.4, 5.2 and 10.4. SPLIT+PASS sits at −3.8, +11.9 and −2.1 from MONO, ONE CALL at +0.8, +12.2 and +1.0, and the G6 contrast at −4.6, −0.3 and −3.1 (Table 4). Generated by `scripts/og2v2_prompt_sensitivity_k5.py --model gemma12b`.

The same check on Llama-3.3-70B (Update 7, whose text was committed before the first call and posted afterwards; run on 24 and 25 September; unregistered, sealed set long open). 10,949 new calls, no transport error. Under the registered wording, A and B the merge-controlled loss is 3.9, 3.6 and 3.8 points (CIs +0.9 to +7.0, +0.7 to +6.6, +0.3 to +7.3), so the sign survives rewording; the move across wordings, 0.3 points, is at most the largest five-draw spread within one wording (5.1). Paired over tasks, A does not separate from the registered wording (+0.3 (CI −1.7 to +2.3)) and B does not separate (+0.1 (CI −3.6 to +3.8)). The raw split loses 3.9, 5.6 and 3.5. SPLIT+PASS sits at −1.6, −0.7 and −1.6 from MONO, ONE CALL at −1.3, −6.4 and +2.9, and the G6 contrast at −0.3, +5.7 and −4.5, reading inconclusive, candidates add value and at most δ, that interval wholly below zero under the registered ladder (Table 4). Generated by `scripts/og2v2_prompt_sensitivity_k5.py --model llama70b`.

Is the 12B collapse a parse artifact? No. Wording A costs Gemma-3-12B's monolithic prompt −11.3 (CI −15.5 to −7.1), and the parser reads 11 of its 990 responses as no call against 6 under the registered wording; excluding every unreadable execution the drop is −10.9 (CI −15.0 to −6.7), so it is the model's answer that changes and not its format. The same rewording moves 27B by +0.9 (CI −1.1 to +2.9) and Llama by −1.0 (CI −2.3 to +0.3). A single call's accuracy at 12B therefore moves further under a rewording of its own prompt than the operator moves it, which is the wording control's point and the reason the 12B row of Table 4 has no stable sign (`scripts/og2v2_wording_parse_rates.py`).

The fail-open convention is asymmetric, and what it costs. An unreadable pass output leaves the candidate list unchanged. For SPLIT+PASS those candidates are the split's calls; for ONE CALL they are the empty list the scorer marks wrong, so the convention taxes only the empty-candidate arm. Of 990 executions per cell the parser reads no call in: 12B 5, 5 and 10 against 5, 5 and 10; 27B 6, 5 and 6 against 0, 0 and 3; Llama 42, 97 and 4 against 16, 27 and 8 (ONE CALL against SPLIT+PASS). On Llama this is what moves the G6 contrast across wordings: as scored it is −0.3 (CI −2.8 to +2.2), +5.7 (CI +2.2 to +9.1) and −4.5 (CI −7.3 to −1.8); over the executions where both arms' output parses, −2.9 (CI −4.8 to −1.0), −2.1 (CI −4.2 to +0.1) and −4.9 (CI −7.6 to −2.2); and crediting the empty-candidate arm's unreadable executions with the monolithic outcome, −3.3 (CI −5.3 to −1.4), −2.5 (CI −4.8 to −0.2) and −4.8 (CI −7.5 to −2.2). The registered readings of Table 2 are unchanged and stay as registered; this is a sensitivity analysis, and it says that where the reworded pass prompt is less readable the registered convention flatters the split's candidates (`scripts/og2v2_one_call_failopen.py`).

Model	Contrast (points)	registered	A	B	spread over draws
Gemma-12B	MERGED − MONO	$-7.1\,(-10.8, -3.4)^*$	$+0.8\,(-4.0, +5.6)$	$-3.0\,(-6.7, +0.6)$	3.0 to 4.5
	SPLIT+PASS − ONE CALL	$-4.6\,(-8.3, -1.0)^*$	$-0.3\,(-2.9, +2.3)$	$-3.1\,(-6.1, -0.2)^*$	2.5 to 3.5
	ONE CALL − MONO	$+0.8\,(-1.8, +3.4)$	$+12.2\,(+7.7, +16.7)^*$	$+1.0\,(-1.0, +3.0)$	2.5 to 3.5
Gemma-27B	MERGED − MONO	$-3.9\,(-7.7, -0.2)^*$	$-6.8\,(-10.4, -3.2)^*$	$-6.7\,(-10.2, -3.1)^*$	1.5 to 3.5
	SPLIT+PASS − ONE CALL	$-1.9\,(-4.8, +1.0)$	$-1.1\,(-3.9, +1.7)$	$-0.5\,(-3.1, +2.1)$	1.5 to 2.5
	ONE CALL − MONO	$+2.2\,(+0.1, +4.4)^*$	$+0.5\,(-1.6, +2.6)$	$+0.2\,(-1.5, +1.9)$	2.0 to 2.5
Llama-70B	MERGED − MONO	$-3.9\,(-7.0, -0.9)^*$	$-3.6\,(-6.6, -0.7)^*$	$-3.8\,(-7.3, -0.3)^*$	3.5 to 5.1
	SPLIT+PASS − ONE CALL	$-0.3\,(-2.8, +2.2)$	$+5.7\,(+2.2, +9.1)^*$	$-4.5\,(-7.3, -1.8)^*$	3.0 to 6.6
	ONE CALL − MONO	$-1.3\,(-3.8, +1.2)$	$-6.4\,(-9.7, -3.0)^*$	$+2.9\,(+0.4, +5.5)^*$	2.0 to 3.0

Table 4: The wording check at k=5 on all three models (27B run on 24 September, the other two under Update 7 on 24 and 25 September, its text committed before their first call): the merge-controlled loss, the registered G6 contrast and the empty-candidate call against the monolithic one, under the registered wording and two rewordings, paired over tasks (*interval excludes zero). The last column is the range, over the three wordings, of that contrast's spread across the five draws within one wording (`og2v2_update7_tables.py`).

Table 5: The wording check at k=5 on Gemma-3-27B, BFCL fresh: five contrasts under the registered wording and two rewordings (points, paired 95% CI over tasks; *interval excludes zero), and each rewording paired against the registered wording task by task. The last row is the range of the merge-controlled contrast over the five draws within one wording (`og2v2_controls_table.py`).

Contrast (points)	registered	A	B	A − registered	B − registered
SPLIT − MONO	$-10.8\,(-15.6, -6.0)^*$	$-13.6\,(-18.4, -8.8)^*$	$-14.1\,(-18.9, -9.3)^*$	$-2.8\,(-6.0, +0.4)$	$-3.3\,(-6.2, -0.5)^*$
MERGED − MONO	$-3.9\,(-7.7, -0.2)^*$	$-6.8\,(-10.4, -3.2)^*$	$-6.7\,(-10.2, -3.1)^*$	$-2.8\,(-6.1, +0.4)$	$-2.7\,(-5.5, +0.0)$
SPLIT+PASS − MONO	$+0.3\,(-2.6, +3.2)$	$-0.6\,(-3.2, +2.0)$	$-0.3\,(-2.8, +2.2)$	$-0.9\,(-3.5, +1.7)$	$-0.6\,(-3.2, +2.0)$
ONE CALL − MONO	$+2.2\,(+0.1, +4.4)^*$	$+0.5\,(-1.6, +2.6)$	$+0.2\,(-1.5, +1.9)$	$-1.7\,(-4.1, +0.6)$	$-2.0\,(-4.2, +0.2)$
SPLIT+PASS − ONE CALL	$-1.9\,(-4.8, +1.0)$	$-1.1\,(-3.9, +1.7)$	$-0.5\,(-3.1, +2.1)$	$+0.8\,(-1.3, +3.0)$	$+1.4\,(-1.4, +4.2)$
spread of MERGED − MONO over the five draws	1.5	3.5	3.5		

Split planner (registered): "Decompose the user request by function capability. Respond between <selection> tags with a JSON array whose elements are {name, subtask}. Use each function name at most once and combine all repeated invocations of that function into its subtask [...] never summarize or drop repeated cases [...]." **Worker:** the call-format instruction, "You are the specialist for exactly one function. Emit all invocations of this function required by the request; if none are required, return []," the function schema, the subtask. **Full-context worker:** the same with every schema, the full request, and "emit ONLY invocations of your assigned function." **Manifest planner:** as the split planner, but each element is {name, n_calls, cases, subtask} with "one case is the complete set of values for one call; never list argument names, descriptions, or steps as separate cases" and a worked example. **Contract worker:** "The planner determined that the request requires exactly n invocation(s) of this function, listed under INVOCATION CASES. Emit exactly that many calls, one per listed case and in the listed order." **Pass:** "You are auditing tool calls produced for a user request. Compare the CANDIDATE CALLS against the USER REQUEST and output a corrected call list [...] every invocation the request asks for must appear exactly once, including repeated calls to the same function with different arguments; remove duplicated calls the request does not ask for; do not invent calls the request does not require; keep correct calls unchanged, including their argument values. If the candidate list is already correct, output it unchanged." **Checklist:** the monolithic prompt plus those three rules. Verbatim texts are in the supplementary material.

B THE EARLIER PASS, ITS DEFECTS, AND THE TIMELINE OF THIS ONE

The first registered pass (four arms: MONO, MONO+PASS, SPLIT, SPLIT+PASS; three models; two benchmarks) reported recovery of 64 to 103% of the gap and specificity over the mono-pass control, and passed every registered criterion at 27B. A post-hoc audit found four harness defects: a strict JSON parser that turned 0 to 12.8% of sealed-test planner outputs into empty plans (moving the headline 27B/Live gap from 17.9 to 7.6 points when excluded); planner truncation at a 256-token ceiling; silent fail-open of unparseable pass outputs (up to 3.3% on Llama); and cached transport errors. The aggregation-rule artifact of §5 was found afterwards and affects v1 and v2 alike; it is

why the merge control exists. The rebuilt harness fixed the parser, the ceiling and the error caching, retained the pass's fail-open as registered behaviour but reports its rate (0 to 1.9% of pass outputs per test cell, Table 16), recorded serving provenance, and was pre-declared, with the new arms and their train pilots, in a second registration update before the pass reported here. All v1 numbers remain in the released record.

Deviations. Eleven, each stated as registered and as done. (1) Hypothesis G3 (audited-space gating) was dropped in the second update as belonging to the companion study and is not reported here. (2) The registration adjusted G1 and G2 jointly across models (four tests, six once Update 1 added 27B); the second update, written after the first pass had shown Llama weak and after the v2 train pilots, but before any v2 execution on a sealed set, made per-model families of four, and §4 reports 12B's G1 under both. (3) G1's instrument changed from a bootstrap interval on the recovery share to the linear contrast, which agree here (the 12B share interval is 53 to 95%). (4) The contract's attribution rule was registered as a roster holding "correctly typed argument values for $\geq 90\%$ of ground-truth calls". What we compute is a proxy: the share of string-valued ground-truth arguments, any accepted variant, found by substring anywhere in the concatenated rosters and subtasks of the plan, not per function, after stripping spaces and punctuation, so short values match liberally. Its unit is an argument, not a call; it ignores the 48% of arguments that are not strings and is undefined on 46 tasks. (5) Update 6 named `required` the primary variant of the native anchor. It read refuted on a run the provider corrupted, which we discard, so H12 has no usable registered reading; the reading we discuss is on `auto`, with the ladder extended to three unregistered cells and a difference in differences that was not registered (Table 31). (6) The merge control was run at the padded catalogue sizes (431 new calls), outside Update 5. (7) So was the full-schema planner at 27B, as Update 6 itself records. (8) Update 5's reporting rule put the character-count token ratio beside accuracy; Table 25 prints the measured ratio there and Table 23 keeps the model. (9) Update 6 registered the fourth model under the registered scorer; we report it under two post-hoc readers as well (Table 29). (10) On 21 September, after the last simulated review round had shown that two of our readings rested on a merge control never run, we ran it: for the full-schema planner of deviation 7 (198 new worker calls; Table 27) and with native workers in the four cells of the native anchor (266; Table 32). Both apply the registered merge rule to plans already in the record, neither is registered, and both were run with the sealed set long open. (11) Update 2 registered a mechanism reading for G5: if the upper bound is below zero, "the loss is the split itself under the registered one-worker-per-function, first-item-wins aggregation rule", and it expected the paper's mechanism claim to be "the split, not the information loss". That branch was triggered at 12B and on Llama. Table 2 reports the registered reading, but §4 declines the mechanism claim on the strength of two post-hoc filters, whose 12B result does not replicate on the other three sets. These eleven are departures from the registered design and analysis. The registration named exact McNemar on majority-binarized outcomes as its sensitivity analysis; Update 2 replaced it with a sign-flip test without comment, and we report only the latter (it agrees with the t-test on all 24 registered tests); we do not count this among the eleven because the replacement was itself registered before the pass. The post-hoc per-item pilot (train split), the 431 catalogue merge calls of deviation 6 and the full-schema planner run of deviation 7 were written into the released caches, not into overlays as the 21 September runs were: no test record was touched and no replay is affected, but those cache files are not the byte state that produced the registered artifacts. Two further things happened outside it and are disclosed where they occur: one harness change between the frozen commit and the artifacts (the cache client aborts on an authentication or billing error instead of storing it; Timeline below), and a 45-call probe of the fourth model, run twice (Table 29).

Unmet expectations. Ten pre-declared expectations went partly unmet. The second update's plan sentence, that one full-context call "with or without candidates" restores parity, failed for the with-candidates half at 12B (SPLIT+PASS minus MONO -3.8, CI -7.2 to -0.5). The second update declared a positive G0 lower bound on the Gemma models, which the fresh set did not confirm, and an inconclusive G0 on Llama, which came out negative in a cell where the contrast is fixed by construction (§4). It gave G1 point expectations of 95 to 105% recovery on the Gemma models; 12B recovers 71% (interval 53 to 95%). Update 6 expected H12 confirmed; its registered variant read refuted on the discarded run (deviation 5). The fourth declared the native split clearly below MONO, which the sealed set left inconclusive, though its other half held (Table 31). The second's residual worker prior toward one call per function (94%/71% on train) is 2.9% sealed (§5). The second expected G2 on all three models, and Llama's does not survive Holm (0.12). The sixth expected the fourth model's gap to be small, near Llama's 3.9, where its raw split loses 38.0 under the registered

parser and 9.4 under the post-hoc tolerant one, and only its merge control 2.9 (tolerant; Table 29). And Update 5's reading (the rule stands) held, but the mechanism it named for the growth, over-selection, did not: the growth is missed functions, in most of which (65 of 72 at 64) the plan names a distractor instead (§6).

Corrections made before submission. This manuscript went through simulated pre-submission reviews (AI use statement), each checked against the artifacts, and several of our own follow-up analyses were wrong when first written. For the record, and so that captions can describe only what a table shows: a first cost table priced prompt plus completion tokens as completions and concluded that the split's saving does not survive prompt caching (1.02× at 64 functions at a quarter-price read, 1.97× everywhere under free reuse; Table 24 replaces it); catalogue token ratios were character-count estimates and the per-entry costs first printed (26.8 and 108.3) had no committed generator (both now measured); the parser sensitivity table first replayed four arms (now eleven) and the parser audit three caches (now all sixteen); the dropped-item class was first defended by a directional test with an empty denominator (Table 35); the F_1 bars of Figure 2 used a looser class rule than the table beside them; the contract conditional was read at complete roster coverage and called the registered rule; the native anchor's difference in differences was computed for one cell of four, and its discard was first explained by a one-call cap that does not exist; the fourth model was reported without its registered controls and intervals, and a probe of it had to be run again because its responses had not been stored; and the registration update quoted 0.740 for the union control of §5, where the committed replay gives 0.732.

Timeline of the confirmatory pass. The update was posted and the flag that unlocks the driver was written on the evening of day 0; the twelve cells started immediately with pinned providers. About five hours in, the router began answering HTTP 402 (monthly credits exhausted); the drivers were stopped, and the one code change between the frozen commit and the artifacts, namely that the cache client aborts on 401/402/403 instead of storing the error record, was committed the next morning. The pass resumed on day 7 after credits were purchased, replaying every cached record and re-asking the 3,609 error records (a count from our run log, which is not part of the release: the re-asked records were overwritten, so it cannot be regenerated); all twelve artifacts were written that evening, each with zero transport-error records. The caches still hold 1,055 orphaned error records that no artifact references (HTTP 402 responses, plus the 48 DNS failures of a pilot incident), and carry no per-record timestamp, so the units executed before and after the interruption cannot be separated. All six registered caches are continuations of the pilot caches and also hold unpinned train-split records. What we can count without a replay is records that carry no provider field: 6,416, 6,403 and 6,371 in the curated caches and 152, 200 and 2,023 in the live ones (`og2v2_v23_addenda.py`). Those include every error record (183/184/184 and 152/200/152), which lack the field because the call failed; the rest, 6,233/6,219/6,187 curated and 1,871 in Llama's live cache, were written before provenance was recorded; the number of train-split records is larger. No test artifact references any of them. Because a (task, sample) unit executes all eleven arms in one thread, at most the units in flight at the stop (eight per cell) straddle the interruption; paired contrasts are protected against provider drift by construction, absolute arm levels are not.

Update 7, and a payload rule that was fixed before its cell. The wording check of Appendix A and the catalogue cell of Table 28 were run on the two models that lacked them after the design, the arms, the contrasts and the reading rules had been written down and committed, 86 seconds before the first call and a day before the text was posted, including the equivalence margin δ for the catalogue cell and the payload criterion for a native probe: a cell is read for accuracy only if fewer than 5% of its tool calls carry an empty function name and fewer than 5% of its responses return no call. The probe, Gemma-3-12B on 18 tasks, the equal-per-category rule of the update's `--smoke 20` over three categories, with the two `tool_choice` conditions counted apart, fails under both: under `required`, the condition the registered 27B cell used, 85 of its 202 tool calls carry an empty function name (42.1%, against 49% at 27B), and under `auto` 10 of 90 responses return no call (11.1%). Both are filed with the corrupted native cells, and the same provider breaks the same way on both Gemma models, rather than read (`og2v2_native_probe_gemma12b.py`). This is the rule the paper's own checklist asks for and the registered 27B cell did not have.

Update 7's expectation for 12B. It expected the merge-controlled loss to stay positive under both rewordings on Gemma-3-12B. It does not: the loss is 7.1, −0.8 and 3.0 points under the three wordings, so under A the merge control sits 0.8 points above MONO (CI −4.0 to +5.6) and the sign is not wording-robust on that model (Appendix A).

Update 7's expectation for the catalogue. It expected the merge-controlled loss at 64 functions to be larger at 12B than at 27B. It is not: MERGED minus MONO is $+1.1$ at 12B (CI -2.6 to $+4.8$) against -2.1 at 27B (CI -4.4 to $+0.1$), so the 12B cell shows no loss at all there and the two are not ordered as expected (Table 28).

A function-calling-tuned model, attempted after the last review round. Reviewers asked whether a model post-trained for function calling groups repeated invocations where the three general models did not. The only provider on our router that serves such a family (Salesforce xLAM-2) returns degenerate completions on a large share of requests: on the 32B model, under the registered text-prompted arms on BFCL fresh at k=5, 611 of 990 monolithic responses parse to no call at all (runs of a repeated character, non-ASCII text spliced into the JSON, truncated keys), 23% of all 4,518 responses carry such a run, and 313 of 990 plans are empty; the 70B sibling's single probe returned a completion made of one repeated character. The cell's means (MONO 0.352, SPLIT 0.221) are therefore the provider's, not the model's, and are filed with the corrupted native cells rather than reported; the run, its cache and the payload forensics are released (`og2v2_tuned_model.py`), and the question stays open for a clean endpoint.

C PER-CATEGORY RESULTS ON THE FRESH SET

Arm	Gemma-12B			Gemma-27B			Llama-70B		
	multiple	parallel	par.-mult.	multiple	parallel	par.-mult.	multiple	parallel	par.-mult.
mono	0.912	0.915	0.909	0.918	0.948	0.888	0.933	0.945	0.848
split	0.836	0.906	0.591	0.867	0.945	0.618	0.894	0.964	0.752
merged	0.836	0.906	0.782	0.867	0.945	0.824	0.894	0.964	0.752
full-ctx	0.839	0.930	0.197	0.903	0.921	0.676	0.888	0.936	0.739
contract	0.882	0.836	0.688	0.867	0.909	0.776	0.842	0.891	0.682
contract+rc	0.882	0.827	0.676	0.867	0.909	0.776	0.842	0.891	0.685
contract+ctx	0.858	0.855	0.745	0.870	0.921	0.852	0.836	0.885	0.697
split+pass	0.882	0.912	0.827	0.933	0.948	0.882	0.918	0.948	0.812
one call	0.927	0.909	0.924	0.921	0.970	0.930	0.903	0.933	0.852
checklist	0.906	0.903	0.912	0.903	0.942	0.900	0.927	0.933	0.855
mono+pass	0.927	0.918	0.927	0.936	0.948	0.903	0.924	0.948	0.842

Table 6: Per-category expected accuracy on BFCL fresh (n=66 per category and model); rows are arms.

D ALL SETS: ARMS, CONTRASTS, SHARES, COST AND DIAGNOSTICS

Table 7: The eleven arms: what the emitting call sees and how its output is post-processed. f selected functions; r count re-asks; †the same cached planner completion; ^{s}a single-function plan hands the worker the full request; ^{a}assignment shown; ^{r}plus the pass's three coverage rules.

Arm	Calls	Planner	The emitting call sees					Candidates	Post-processing
			request	own schema	all schemas	subtask	count+roster		
MONO	1	n/a	✓	n/a	✓	n/a	n/a	n/a	n/a
CHECKLIST	1	n/a	✓^{r}	n/a	✓	n/a	n/a	n/a	n/a
ONE CALL	1	n/a	✓	n/a	✓	n/a	n/a	empty	n/a
MONO+PASS	2	n/a	✓	n/a	✓	n/a	n/a	MONO's	fail-open
SPLIT	$1+f$	split	n/a^{s}	✓	n/a	✓	n/a	n/a	first item wins
MERGED	$1+f$	split†	n/a^{s}	✓	n/a	all items	n/a	n/a	repeats merged
FULL-CTX	$1+f$	split†	✓	n/a	✓^{a}	n/a	n/a	n/a	none
CONTRACT	$1+f$	manifest	n/a^{s}	✓	n/a	✓	✓	n/a	none
CONTRACT+RC	$1+f+r$	manifest†	n/a^{s}	✓	n/a	✓	✓	n/a	count re-ask
CONTRACT+CTX	$1+f$	manifest†	✓	n/a	✓^{a}	n/a	✓	n/a	leak filter
SPLIT+PASS	$2+f$	split†	✓	n/a	✓	n/a	n/a	SPLIT's	fail-open

Arm	BFCL test			Live test			Live fresh		
	12B	27B	70B	12B	27B	70B	12B	27B	70B
gap (pts)	11.0	8.6	3.5	6.9	8.2	-1.3	15.8	15.0	3.2
mono	0.931	0.930	0.935	0.697	0.759	0.692	0.762	0.785	0.753
split	0.822	0.845	0.900	0.628	0.677	0.705	0.603	0.635	0.722
merged	0.880	0.918	0.910	0.628	0.677	0.705	0.603	0.635	0.722
full-ctx	0.673	0.865	0.917	0.533	0.679	0.697	0.572	0.700	0.703
contract	0.812	0.880	0.860	0.692	0.644	0.664	0.680	0.675	0.692
contract+rc	0.810	0.877	0.860	0.692	0.646	0.659	0.680	0.675	0.700
contract+ctx	0.857	0.882	0.900	0.631	0.674	0.628	0.642	0.690	0.660
split+pass	0.911	0.941	0.927	0.697	0.751	0.700	0.673	0.752	0.767
one call	0.923	0.937	0.942	0.672	0.744	0.746	0.708	0.785	0.743
checklist	0.931	0.937	0.931	0.721	0.741	0.690	0.762	0.745	0.765
mono+pass	0.933	0.930	0.939	0.703	0.764	0.715	0.762	0.808	0.770

Table 8: Expected accuracy per arm on the replication sets (BFCL test n=166; Live test n=78; Live fresh n=120); rows are arms, columns are set $\times$ model (12B, 27B: Gemma-3; 70B: Llama-3.3).

Set	Model	G1 pass	G2 DiD	G5 full-ctx	G6 pass−one call
BFCL fresh	Gemma-12B	$+2.9$ ($+0.2$, $+5.6$)*	$+8.4$ ($+4.4$, $+12.3$)*	-18.9 (-24.7, -13.2)*	-4.6 (-8.3, -1.0)*
BFCL fresh	Gemma-27B	$+5.7$ ($+3.0$, $+8.5$)*	$+10.0$ ($+5.5$, $+14.5$)*	-3.1 (-7.2, $+1.0$)	-1.9 (-4.8, $+1.0$)
BFCL fresh	Llama-70B	$+0.4$ (-1.4, $+2.2$)	$+2.7$ ($+0.1$, $+5.3$)*	-3.5 (-6.0, -1.0)*	-0.3 (-2.8, $+2.2$)
BFCL test	Gemma-12B	$+3.4$ ($+0.8$, $+6.1$)*	$+8.8$ ($+4.7$, $+12.9$)*	-20.3 (-26.4, -14.2)*	-1.2 (-4.9, $+2.5$)
BFCL test	Gemma-27B	$+5.4$ ($+2.5$, $+8.2$)*	$+9.6$ ($+5.2$, $+14.0$)*	-2.2 (-6.1, $+1.7$)	$+0.4$ (-2.2, $+2.9$)
BFCL test	Llama-70B	$+0.9$ (-1.2, $+3.0$)	$+2.3$ ($+0.0$, $+4.6$)*	-0.1 (-1.8, $+1.7$)	-1.6 (-4.1, $+0.9$)
Live test	Gemma-12B	$+3.5$ (-1.3, $+8.2$)	$+6.4$ ($+1.2$, $+11.6$)*	-12.9 (-20.0, -5.9)*	$+2.6$ (-4.3, $+9.4$)
Live test	Gemma-27B	$+3.3$ (-0.2, $+6.9$)	$+6.9$ ($+1.7$, $+12.1$)*	-3.8 (-10.1, $+2.4$)	$+0.8$ (-5.5, $+7.1$)
Live test	Llama-70B^d	$+0.1$ (-3.4, $+3.6$)	-2.8 (-8.8, $+3.1$)	-0.1 (-5.2, $+5.0$)	-4.6 (-10.2, $+1.0$)
Live fresh	Gemma-12B	-0.9 (-4.6, $+2.8$)	$+7.0$ ($+3.2$, $+10.8$)*	-11.1 (-15.9, -6.3)*	-3.5 (-10.6, $+3.6$)
Live fresh	Gemma-27B	$+4.2$ ($+0.6$, $+7.7$)*	$+9.3$ ($+4.9$, $+13.8$)*	-1.0 (-5.5, $+3.5$)	-3.3 (-7.4, $+0.8$)
Live fresh	Llama-70B	$+2.9$ (-0.4, $+6.2$)	$+2.8$ (-2.0, $+7.7$)	-3.4 (-6.8, -0.1)*	$+2.3$ (-2.2, $+6.8$)

Table 9: Family-A contrasts on all sets (points, paired t 95% CI; * excludes 0, *unadjusted*). ^{d}gap below 3 points, descriptive by the registered rule.

Set	Model	G0 merged	G7 contract	G8 contract+ctx	G9 checklist
BFCL fresh	Gemma-12B	-0.4 (-2.9, $+2.2$)	-4.3 (-8.9, $+0.3$)	-2.6 (-7.3, $+2.2$)	-0.5 (-1.3, $+0.3$)
BFCL fresh	Gemma-27B	$+1.5$ (-1.2, $+4.2$)	-1.4 (-5.3, $+2.5$)	$+1.7$ (-1.9, $+5.2$)	-0.3 (-1.7, $+1.1$)
BFCL fresh	Llama-70B	-2.0 (-3.5, -0.5)$^{*\circ}$	-8.4 (-12.2, -4.7)*	-8.3 (-12.2, -4.5)*	-0.4 (-1.4, $+0.6$)
BFCL test	Gemma-12B	$+0.3$ (-2.2, $+2.8$)	-6.4 (-10.9, -2.0)*	-2.0 (-6.5, $+2.5$)	-0.0 (-0.5, $+0.5$)
BFCL test	Gemma-27B	$+3.1$ ($+0.1$, $+6.1$)*	-0.8 (-4.8, $+3.3$)	-0.5 (-4.8, $+3.7$)	$+0.7$ (-1.0, $+2.4$)
BFCL test	Llama-70B	-0.8 (-2.2, $+0.6$)	-5.7 (-9.4, -2.0)*	-1.7 (-4.5, $+1.0$)	-0.4 (-1.5, $+0.7$)
Live test	Gemma-12B	-3.5 (-7.9, $+1.0$)$^\circ$	$+2.9$ (-3.3, $+9.2$)	-3.2 (-10.8, $+4.4$)	$+2.3$ (-2.2, $+6.8$)
Live test	Gemma-27B	-4.1 (-7.1, -1.1)$^{*\circ}$	-7.4 (-14.0, -0.9)*	-4.4 (-9.9, $+1.2$)	-1.8 (-4.7, $+1.1$)
Live test	Llama-70B^d	$+0.6$ (-3.0, $+4.3$)$^\circ$	-3.5 (-10.9, $+3.9$)	-7.1 (-15.3, $+1.2$)	-0.3 (-3.5, $+3.0$)
Live fresh	Gemma-12B	-7.9 (-11.6, -4.2)$^{*\circ}$	-0.3 (-5.6, $+5.1$)	-4.1 (-9.6, $+1.4$)	-0.0 (-1.1, $+1.1$)
Live fresh	Gemma-27B	-7.5 (-10.6, -4.4)$^{*\circ}$	-3.5 (-8.2, $+1.2$)	-2.0 (-6.8, $+2.8$)	-4.0 (-6.9, -1.1)*
Live fresh	Llama-70B	-1.6 (-4.9, $+1.7$)$^\circ$	-4.6 (-8.6, -0.6)*	-7.8 (-12.3, -3.2)*	$+1.2$ (-1.9, $+4.2$)

Table 10: Family-B contrasts on all sets (points, paired t 95% CI; * excludes 0, *unadjusted*). ^{d}gap below 3 points, descriptive by the registered rule. $^\circ$the merge control is a byte-level no-op on this cell (identical outcome on $\geq 99.9\%$ of executions; byte-identical output on $\geq 99.7\%$), so G0 equals $-\frac{1}{2}$gap by construction.

Set	Model	gap	merged	full-ctx	contract	contract +ctx	pass	same outcome	counts = GT	compliant
BFCL fresh	Gemma-12B	13.4	47%	-91%	18%	31%	71%	94%	91%	98%
BFCL fresh	Gemma-27B	10.8	64%	21%	37%	65%	103%	93%	94%	100%
BFCL fresh	Llama-70B	3.9	0%	-38%	-164%	-162%	59%	100%	94%	99%
BFCL test	Gemma-12B	11.0	53%	-135%	-9%	32%	81%	94%	93%	99%
BFCL test	Gemma-27B	8.6	86%	24%	41%	44%	113%	93%	94%	99%
BFCL test	Llama-70B	3.5	28%	48%	-114%	0%	76%	99%	94%	98%
Live test	Gemma-12B	6.9	0%	-137%	93%	4%	100%	100%	91%	100%
Live test	Gemma-27B	8.2	0%	3%	-41%	-3%	91%	100%	88%	99%
Live test	Llama-70B	-1.3	n/a	n/a	n/a	n/a	n/a	100%	92%	99%
Live fresh	Gemma-12B	15.8	0%	-20%	48%	24%	44%	100%	83%	100%
Live fresh	Gemma-27B	15.0	0%	43%	27%	37%	78%	100%	81%	100%
Live fresh	Llama-70B	3.2	0%	-58%	-95%	-195%	142%	100%	88%	99%

Table 11: Recovery shares (point estimates), the share of executions on which the merge control's outcome is identical to the split's, and contract diagnostics (declared counts equal to the ground truth per function; worker count compliance), all sets.

Table 12: Share of the gap recovered on BFCL fresh, with bootstrap 95% intervals over tasks (descriptive; the registered tests are in Table 2).

Model	gap	merged	contract	contract+ctx	full-ctx	split+pass	one call
Gemma-12B	13.4	47% [28, 67]	18% [-27, 52]	31% [-18, 63]	-91% [-184, -38]	71% [53, 95]	106% [88, 131]
Gemma-27B	10.8	64% [40, 98]	37% [-14, 71]	65% [27, 97]	21% [-31, 58]	103% [79, 142]	121% [102, 158]
Llama-70B	3.9	0% [0, 0]	-164% [-652, -51]	-162% [-717, -44]	-38% [-279, 23]	59% [-21, 100]	67% [-72, 131]

Arm	BFCL fresh			BFCL test			Live test			Live fresh		
	12B	27B	70B	12B	27B	70B	12B	27B	70B	12B	27B	70B
mono	430	458	429	439	465	434	859	872	829	905	920	872
split	828	871	851	815	860	837	909	922	874	901	944	852
merged	836	879	851	820	867	837	909	922	874	901	944	852
full-ctx	1229	1221	1196	1184	1189	1158	1703	1639	1498	1667	1714	1458
contract	1148	1193	1147	1128	1179	1123	1139	1175	1133	1133	1177	1120
contract+rc	1157	1194	1152	1137	1184	1134	1139	1181	1138	1133	1178	1126
contract+ctx	1563	1602	1554	1525	1559	1499	1830	1871	1789	1849	1893	1763
split+pass	1381	1424	1424	1376	1423	1405	1887	1890	1825	1912	1954	1837
one call	508	511	571	512	519	567	926	934	985	966	979	1018
checklist	478	501	475	486	507	480	905	918	874	949	963	920
mono+pass	985	1013	998	1002	1030	1000	1825	1833	1778	1904	1918	1855

Table 13: Mean tokens per (task, sample), prompt + completion, all calls of the arm, all sets; rows are arms, columns are set × model. Registered G4, what the pass buys per extra token, reads 17.4/20.1/4.1 points per 1k tokens over SPLIT on BFCL fresh.

Arm	Gemma-12B			Gemma-27B			Llama-70B		
	prompt	compl.	calls	prompt	compl.	calls	prompt	compl.	calls
mono	353	77	1.00	353	104	1.00	363	66	1.00
split	660	168	2.54	658	213	2.53	701	150	2.52
merged	663	172	2.54	661	218	2.53	701	150	2.52
full-ctx	1031	198	2.54	1026	195	2.53	1046	150	2.52
contract	957	191	2.50	957	236	2.51	983	164	2.52
contract+rc	965	192	2.53	958	237	2.51	987	164	2.53
contract+ctx	1374	189	2.50	1386	216	2.51	1389	164	2.52
split+pass	1143	238	3.54	1141	283	3.53	1190	234	3.52
one call	423	84	1.00	423	87	1.00	432	138	1.00
checklist	401	77	1.00	401	100	1.00	410	65	1.00
mono+pass	838	147	2.00	839	175	2.00	852	146	2.00
G4: points recovered per 1k extra tokens (split+pass)		17.4			20.1			4.1	

Table 14: Prompt and completion tokens and API calls per (task, sample) on BFCL fresh, by arm (post-hoc replay of the cached usage; prompt plus completion equals the BFCL-fresh rows of the table above), and the registered descriptive G4 statistic.

Set	Model	repeated-name plans	fan-out≥2	lenient or retried plans	empty manifests	reconcile triggered
BFCL fresh	Gemma-12B	41%	35%	0.0%	0.5%	1.8%
BFCL fresh	Gemma-27B	40%	35%	1.4%	0.0%	0.2%
BFCL fresh	Llama-70B	18%	33%	3.0%	0.0%	0.9%
BFCL test	Gemma-12B	42%	30%	0.0%	0.0%	1.4%
BFCL test	Gemma-27B	42%	31%	2.2%	0.0%	0.8%
BFCL test	Llama-70B	12%	29%	3.6%	0.0%	2.4%
Live test	Gemma-12B	14%	29%	0.0%	0.0%	0.0%
Live test	Gemma-27B	13%	27%	11.0%	0.0%	1.0%
Live test	Llama-70B	5%	19%	2.3%	0.0%	0.8%
Live fresh	Gemma-12B	0%	24%	0.0%	0.0%	0.0%
Live fresh	Gemma-27B	1%	28%	9.7%	0.0%	0.2%
Live fresh	Llama-70B	0%	10%	0.0%	0.0%	1.3%

Table 15: Planner diagnostics per cell: share of plans with a repeated function name, fan-out ≥ 2, plans needing the lenient parser or the retry, manifests that failed closed, and contract reconciliations triggered (the descriptive CONTRACT+RC arm).

Set	Model	fail-open split+pass	fail-open mono+pass	truncated passes	full-ctx workers with foreign calls	foreign calls per worker
BFCL fresh	Gemma-12B	0.5%	0.5%	0.0%	21.4%	0.37
BFCL fresh	Gemma-27B	0.0%	0.5%	0.0%	0.4%	0.00
BFCL fresh	Llama-70B	1.6%	1.6%	0.2%	0.8%	0.02
BFCL test	Gemma-12B	0.2%	0.2%	0.0%	17.9%	0.31
BFCL test	Gemma-27B	0.0%	0.0%	0.0%	0.0%	0.00
BFCL test	Llama-70B	1.7%	1.9%	0.0%	0.1%	0.00
Live test	Gemma-12B	0.3%	0.0%	0.0%	20.5%	0.25
Live test	Gemma-27B	0.0%	0.0%	0.0%	0.6%	0.01
Live test	Llama-70B	1.0%	1.3%	0.0%	0.2%	0.00
Live fresh	Gemma-12B	0.0%	0.0%	0.0%	5.8%	0.06
Live fresh	Gemma-27B	0.0%	0.0%	0.0%	0.6%	0.01
Live fresh	Llama-70B	0.2%	0.8%	0.0%	0.9%	0.01

Table 16: Pass and full-context diagnostics per cell: fail-open rates of the pass on split and monolithic candidates, truncated passes, and full-context workers that emitted a call to a non-assigned function.

E POST-HOC DIAGNOSTICS ON THE FRESH SET

What the full-context arm changes, and what that costs. At 12B the arm is 12.2 points below the split overall and 39 below on parallel_multiple (0.197). It moves five boundaries at once, adding the request and every schema while dropping the subtask, and its shared preamble asks for "every call requested by the user" before the arm narrows that to the assigned function, which alone licenses the foreign calls we attribute to the model. Even filtered it stays below MONO (-2.8, -3.8, -4.5 points).

The one-call symptom, by whether the rule had anything to drop. Where the planner repeated a function's name, so first-item-wins had cases to discard, split workers emit exactly one call in $134/144$, $128/136$ and $3/8$ slots; where it did not, in $5/26$, $5/34$ and $5/161$. The first cut includes the 10/8/5 slots whose plan has fan-out 1, where the worker receives the whole request and nothing is discarded; excluding them the counts are $134/134$, $128/128$ and $3/3$. After the merge 97%/97%/96% of slots receive exactly the ground-truth count (`og2v2_v26_addenda.py`).

What the pass does, and what the candidates do to it. On Gemma the pass mostly adds, re-adding dropped invocations (142/145 added against 25/35 removed; Appendix E); on Llama it rewrites

arguments. Its harm is omission: at 12B it leaves 97 of 220 wrong split outputs unchanged and the empty-candidate call is right on 54 of those. The candidates anchor it, though our prompt also tells it to leave a correct list unchanged (Appendix A), so the anchoring is partly instructed. Deterministic merges cannot substitute for it: union and intersection are both worse than the monolithic list. A post-hoc train pilot of one worker per planner item, no merge, sits at or below it (Appendix F).

The manifest arm loses where its planner miscounts, then through transcription. A miscount is fatal: contract accuracy is 0.88/0.91/0.86 when the declared counts are right and 0.07/0.00/0.00 when they are not, and 9.5%/6.4%/6.5% of executions carry a wrong declared count vector, 29/20/20 of them because the planner selected the wrong functions (Appendix E). Wrong counts explain 57%/50%/56% of the contract's failures on *parallel*, where the planner declares too few (22/13/14 failures against 9/2/6), and 45%/41%/27% on parallel_multiple, where it declares too many (35/24/28 against 6/1/0), as on *multiple*. The count instruction is never isolated from the planner that supplies it: CONTRACT changes planner prompt, worker prompt and emission constraint at once, and that planner selects differently, so this describes the arm, not the count clause; the separating arm, a manifest plan with uncontracted workers, was not run. The rest are argument errors that persist when the worker also sees the full request: among CONTRACT+CTX failures with correct counts, compliant workers, no leakage and a right monolithic call, $35/42$, $11/17$ and $33/56$ carry a bad string argument, matching the roster's phrasing almost as often as the request's own (Appendix E). The failure is verbatim transcription where MONO normalizes to the schema's value. Because the roster quotes the request and the worker sees both, those two counts nearly coincide and this design cannot say which of them induces the copying; a roster-free count contract would.

Selection followed by one call, at a padded catalogue. On the 64% of 64-function executions where the planner names one function, so the operator is selection followed by one call on the full request, the names-only split scores 0.907 against MONO's 0.934 (-2.7, CI -5.8 to $+0.5$; -0.5 without the 17 substitution tasks) at a third of the tokens, and where it fans out, 0.437 against 0.908. That subset is the planner's choice, not an arm.

Argument fidelity of the hand-off. The split's subtasks jointly carry, as a mean over executions, 87%/87%/96% of the ground-truth string values on parallel_multiple, against 95%/96%/96% for merged items; paraphrase loss is the majority mechanism only at 12B, and elsewhere the worker had the value and transcribed, omitted or normalized it wrongly (Appendix E).

Contrast	Gemma-12B	Gemma-27B	Llama-70B
split+pass − mono	-3.8 (-7.2, -0.5)*	$+0.3$ (-2.6, $+3.2$)	-1.6 (-3.4, $+0.1$)
one call − mono	$+0.8$ (-1.8, $+3.4$)	$+2.2$ ($+0.1$, $+4.4$)*	-1.3 (-3.8, $+1.2$)
mono+pass − mono	$+1.2$ (-0.2, $+2.6$)	$+1.1$ (-0.3, $+2.5$)	-0.4 (-1.4, $+0.6$)
checklist − one call	-1.3 (-3.8, $+1.2$)	-2.5 (-5.0, -0.1)*	$+0.9$ (-1.7, $+3.5$)
contract − one call	-11.8 (-16.5, -7.1)*	-9.0 (-12.7, -5.3)*	-9.1 (-13.7, -4.5)*
contract − merged	-3.9 (-8.4, $+0.5$)	-2.8 (-6.6, $+0.9$)	-6.5 (-10.3, -2.6)*
contract+ctx − full-ctx	$+16.4$ ($+9.7$, $+23.1$)*	$+4.7$ (-0.2, $+9.7$)	-4.8 (-8.7, -1.0)*
contract+rc − contract	-0.7 (-1.7, $+0.3$)	$+0.0$ ($+0.0$, $+0.0$)	$+0.1$ (-0.1, $+0.3$)

Table 17: Registered descriptive contrasts on BFCL fresh (points, paired t 95% CI; * excludes 0, unadjusted).

Model	loss events	dropped item (merge right)	dropped-item signature	wrong function set	superset, extra emits	substi-tution	subset	worker-side	residual	argument	under	over
Gemma-12B	145	63	75	32	25	5	0	2	50	35	12	3
Gemma-27B	128	63	68	28	23	4	1	0	37	31	5	1
Llama-70B	53	0	0	23	14	5	0	4	30	29	0	1

Table 18: Loss events (mono right, split wrong) sub-classified: dropped item by the conservative rule (merge control right) and by signature (repeated-name plan and an under-emitting split); wrong emitted function set by planner selection; residual by the split's failure class.

Model	split	merged	full-ctx	+leak filter	+dedup	mono	par.-mult. full-ctx	+leak filter	+dedup
Gemma-12B	0.778	0.841	0.656	0.736	0.884	0.912	0.197	0.427	0.848
Gemma-27B	0.810	0.879	0.833	0.833	0.880	0.918	0.676	0.676	0.815
Llama-70B	0.870	0.870	0.855	0.864	0.864	0.909	0.739	0.767	0.767

Table 19: The full-context arm under two zero-LLM filters (post hoc, re-scored from the cached worker outputs): calls to non-assigned functions dropped (leak filter), then exact duplicate calls removed (dedup); overall and parallel_multiple accuracy. The leak filter alone removes the 30% of 12B's parallel_multiple calls that belong to another worker, which leaves the arm at 0.736, still below the merge control; with dedup on top it is 4.2 points above that control at 12B (CI $+0.4$ to $+8.1$) and not separated from it at 27B.

Model	exact set	superset / wrong	fan-out$\geq$2	split / mono at fan-out$\geq$2	split / mono at fan-out 1	failures with wrong set
Gemma-12B	89%	31/36	31/330	0.00 / 0.65	0.92 / 0.94	36/54
Gemma-27B	93%	20/24	20/330	0.10 / 0.90	0.92 / 0.92	22/44
Llama-70B	97%	5/10	5/330	0.00 / 1.00	0.91 / 0.93	10/35

Table 20: Planner selection on *multiple*, BFCL fresh: exact-set accuracy; supersets among wrong sets; fan-out-≥ 2 executions with split and monolithic accuracy there and at fan-out 1; split failures with a wrong emitted set.

Model	split items	merged items	roster + subtask	slots needing $\geq$2	one call (split)	one call (merged)	exact count (merged)
Gemma-12B	0.87	0.95	0.91	170	82%	3%	97%
Gemma-27B	0.87	0.96	0.94	170	78%	3%	97%
Llama-70B	0.96	0.96	0.94	169	5%	3%	96%

Table 21: Hand-off coverage on parallel_multiple, BFCL fresh: the share of ground-truth string values contained in the plan's subtasks taken together, so a value counts if any worker received it (split items / merged items / roster plus subtask), computed per execution and averaged over executions, not pooled over values; executions with repeated invocations hold more values and are where first-item-wins discards them, so the split-against-merged contrast in this column is mostly the dropped items and not paraphrase; and the share of slots needing ≥ 2 calls that received exactly one (split, merged) or exactly the right number (merged). Containment is measured after normalization, so it has a floor: on the same tasks a randomly paired request covers 0.114 of a task's values and the task's own request covers 0.967 (`og2v2_v20_addenda.py`).

	fan-out 1		fan-out 2		fan-out 3		fan-out ≥ 4	
Model	execs	ratio	execs	ratio	execs	ratio	execs	ratio
12B	639	2.13×	201	1.78×	115	1.75×	35	1.60×
27B	648	1.89×	192	1.80×	115	1.99×	35	1.80×
Llama	661	2.19×	184	2.09×	110	1.98×	35	2.14×

Table 22: Counterfactual latency of SPLIT relative to MONO, BFCL fresh, conditional on the planner's fan-out: the median over executions of the per-execution ratio, not the ratio of two medians. The counterfactual is the planner call plus the slowest worker, from per-call latencies recorded in the cache. It is a lower bound on the real penalty: it assumes perfect worker parallelism, no orchestration overhead and no contention, while the harness executes workers sequentially. Absolute medians are provider artifacts, so only within-model ratios are reported in this table. The aggregate figure would be dominated by fan-out 1, which is two thirds of executions; the split is slower at every fan-out separately.

functions per task	MONO	SPLIT	ratio
as shipped (2.15)	386	804	2.08×
8	1059	983	0.93×
16	1987	1229	0.62×
32	3832	1718	0.45×
64	7535	2702	0.36×
128	14928	4667	0.31×

Table 23: Prompt tokens for MONO and SPLIT on the same 198 tasks as the tool catalogue grows, with distractor schemas drawn from the rest of the benchmark. This table is a model, not a measurement: a token is taken to be four characters of prompt, the planner's recorded fan-out (1.54) is applied to a mean full-request worker, and no model is called, which is what lets it run to 128 functions. Against the provider's own token counts for the executions we did run (Table 24) it overstates the split's ratio by about 8% at the four padded sizes and 12% as shipped (2.08× against 1.86× as shipped, 0.36× against 0.33× at 64); every ratio quoted in the text is the measured one. The monolithic call pays for every schema and a worker pays for one, so the ratio falls below 1 just under eight functions per task. What happens to accuracy at those sizes is measured in Table 25.

			priced, output at 3× input, cache read at			
functions per task	prompt only	prompt + completion	1 (no cache)	0.25	0.10	0 (free)
as shipped (2.15)	1.86×	1.90×	1.95×	1.98×	2.00×	2.01×
8	0.86×	0.98×	1.16×	1.59×	1.83×	2.09×
16	0.57×	0.66×	0.80×	1.25×	1.59×	2.10×
32	0.42×	0.47×	0.56×	0.90×	1.27×	2.07×
64	0.33×	0.36×	0.41×	0.64×	0.94×	2.03×
32, worker prefixes billed fresh			0.56×	1.11×	1.70×	3.00×
64, worker prefixes billed fresh			0.41×	0.76×	1.23×	2.91×
8, full-schema planner	1.49×	1.56×	1.66×	1.87×	1.99×	2.12×
64, full-schema planner	1.07×	1.09×	1.12×	1.25×	1.42×	2.06×

Table 24: What the split costs relative to MONO, in the provider's own token counts for the executions of Tables 25 and 27 (`og2v2_cached_cost.py`; 27B, read-only replay, no model calls). *Prompt only* and *prompt + completion* are unpriced token ratios. The priced columns put output at 3× input and bill the reusable prefix of every prompt, its instruction and catalogue, at a fraction of the fresh rate: a deployment's tool set is fixed and both arms put the catalogue before the request, so a served model reads it from cache. A prompt's tokens are divided into prefix and remainder by the character share before the request. This is a pricing model applied to measured tokens, not a measurement: in the experiment itself no two tasks share a catalogue, and cache writes and expiry are not priced.

Reading the cost table. The upper block also discounts each worker's prefix, its instruction and one schema, about 330 tokens per execution; that needs every function's worker prefix kept warm and is shorter than the minimum cacheable prefix of the metered caches we know of, so the middle block bills those prefixes fresh. The same minimum would also deny the names-only planner's own prefix a cache read at 16 and 32 functions, where it falls under that minimum and MONO's does not; we do not apply it there, so those two rows flatter the split, and the 64-function row, where both prefixes are long, is unaffected; at 8 functions and as shipped MONO's prefix is under that minimum too, so there the discounted columns would revert to the no-cache column for both arms. The discounts and the output ratio are those of metered closed APIs, applied to open models' token counts. At 64 functions the names-only split costs 0.41× fresh and 0.64 to 0.76× at a quarter-price read; at a tenth it is 0.94× or 1.23× (1.13× or 1.37× with output at 5×), and 2 to 3× where prefix reuse is free, as on self-hosted weights. Priced fresh it crosses 1 between 8 and 16 functions. The full-schema planner of Table 27 costs more than MONO in every column. The split is also slower at every size: the counterfactual latency ratio (planner plus slowest worker over MONO, median over executions) is 1.9, 1.9, 2.2, 2.0 and 1.7 as shipped and at 8, 16, 32 and 64 functions, because the planner decodes about as many tokens (92 to 97) as the whole monolithic answer (101 to 106).

functions per task	MONO	SPLIT	SPLIT − MONO (pts, 95% CI)	fan-out	prompt-token ratio
as shipped (2.15)	0.918	0.810	−10.8 [−15.6, −6.0]	n/a	1.86×
8	0.917	0.788	−12.9 [−18.1, −7.8]	1.55	0.86×
16	0.917	0.769	−14.8 [−20.3, −9.4]	1.56	0.57×
32	0.918	0.767	−15.2 [−20.6, −9.7]	1.56	0.42×
64	0.924	0.737	−18.7 [−24.2, −13.2]	1.56	0.33×

Table 25: Accuracy against catalogue size, one pass on BFCL fresh at 27B ($k = 5$, $T = 0.7$, 7,920 executions), with the measured prompt-token ratio of the same executions beside it (Table 24). Registered in Update 5 as descriptive, with the padding rule, the sizes and this reading fixed before the padded condition was run; the set itself had been analysed by then, so this is a further contact with BFCL fresh and not a clean one, as is every follow-up from Update 3 on. Each task's catalogue is padded with distractor schemas from the pool's 675 distinct functions, the task's own excluded, deterministically and identically for both arms, and *appended*: the task's own functions keep positions 1 to 4 of the 64, the most favourable place for a reader of a long catalogue, so "MONO is flat under distractors" is measured there and nowhere else. The distractors are other tasks' functions and some do a similar job to a task's own (the distractor `hotel.book` beside a task's own `hotel_booking`, `history.get_event_date` beside `get_event_date`); Table 26 shows how much of the growth they carry; MONO and the planner see all of them, each worker still sees only its own. The unpadded row is the registered arm itself, read from the confirmatory artifact rather than re-executed. The pre-declared reading was that the practical rule is scoped to small catalogues if the contrast rises toward zero and some upper bound crosses it, and that the rule stands if the contrast at 64 functions is no closer to zero than unpadded. It is farther: the loss grows monotonically and no interval covers zero, while MONO is flat under distractors and the split is not. Fan-out is unchanged (1.55 to 1.56), so the manipulation did not turn the operator into something else. This was the expectation stated in the update before the run.

functions	raw	merge control	both corrections	wrong set	planner exact	misses one
as shipped (2.15)	10.8	3.9	2.6	n/a	n/a	n/a
8	12.9	6.1	3.5	4.4	0.941	0.010
16	14.8	7.4	5.1	4.8	0.932	0.023
32	15.2	8.4	5.6	5.6	0.919	0.043
64	18.7	12.8	8.2	9.8	0.873	0.073

Table 26: Which channel widens when the catalogue grows (post hoc, replay of the Update 5 executions with no new calls except the merge control's 431; `og2v2_catalogue_mechanism.py`, `og2v2_catalogue_merge.py`). Losses in points against MONO: *raw* is the split, *merge control* undoes our first-item-wins rule, *both corrections* adds per-call F_1. All three grow under the registered names-only planner, so the small doubly corrected number of Table 38 belongs to the shipped catalogue under that planner; with a full-schema planner the merge-controlled loss does not detectably grow (Table 27). *wrong set* is the loss carried by executions whose emitted function set differs from the ground truth's; among gross loss events it gains 5.4 points from 8 to 64 functions while everything else loses 0.3, together +5.1 of the +5.8 by which the net gap grows, the rest being fewer gains. Table 27 shows the growth is ours: it is what the planner's names-only catalogue costs. *planner exact* is the share of plans selecting exactly the needed functions and *misses one* the share missing at least one; the planner's failure at scale is missing a needed function, not over-selection, which holds at 0.037 to 0.055. Of the 72 misses at 64 functions 65 name a distractor instead, concentrated in 17 tasks (10, 22 and 43 such misses on 2, 5 and 10 tasks at 8, 16 and 32): 62 of the 65, on 15 of the 17 tasks, name a distractor whose name shares a content word with the function missed, and the scorer credits only the task's own. This is disambiguation among near-duplicate tools from names and descriptions, which a full-schema planner mostly resolves (misses 0.011), not retrieval at scale. Without those 17 tasks the loss is 12.2, 13.7, 12.6 and 13.9 and the 8-to-64 growth +1.8 (CI −1.8 to +5.4) instead of +5.8 (CI +1.6 to +9.9); on them alone it is +48.2. The 17 are the tasks with such a miss at 64; four more have one only at a smaller size, and without all 21 the losses are 11.3, 12.9, 11.2 and 13.2 and the growth +1.9 (CI −1.6 to +5.4). Excluding tasks on a planner outcome conditions on the treatment, so what a deployment would see lies between the two (`og2v2_v24_addenda.py`). The unpadded row is the registered arm.

functions	planner catalogue	loss (pts)	planner exact	misses one	prompt-token ratio
8	names + descriptions	12.9	0.941	0.010	0.86×
8	full schemas	9.4	0.975	0.001	1.49×
64	names + descriptions	18.7	0.873	0.073	0.33×
64	full schemas	10.2	0.963	0.011	1.07×

Table 27: What the planner's catalogue representation costs (unregistered follow-up; executions and accuracy columns from `og2v2_catalogue_planner_info.py`, the table and its measured ratio column written by `og2v2_cached_cost.py`). The registered planner is shown {name, description} per function; MONO is shown the full schema, which is 24.8 against 101.2 prompt tokens per additional catalogue entry, measured from the provider's token counts between 8 and 64 functions. The follow-up changes only that: the planner's catalogue carries the same schemas MONO reads; workers and aggregation are untouched. Most of the growth in Table 25 is that one choice: the split is cheaper at a large catalogue only while its planner is under-informed, and the information that repairs the planner costs about what it saves. The losses in the table are raw-split losses. Of the full-schema split's 116 loss events at 64 functions, 85 (8.6 of 11.7 gross points, the net 10.2 being those less 15 gains; 81 of 116, 8.2 of 11.7, at 8) carry the dropped-item signature of §5. The merge control for this planner was run on 21 September, after the last review round (`og2v2_catalogue_merge_schemas.py`; unregistered; the planner calls replayed, 198 new worker calls kept in a cache file of their own): MONO, split and merged score 0.924, 0.822 and 0.903 at 64 functions and 0.917, 0.823 and 0.901 at 8, so the merge-controlled loss is 2.1 points (CI −0.1 to +4.4) and 1.6 (CI −1.1 to +4.3), with no detectable growth between the two sizes (+0.5, CI −2.2 to +3.2), where the names-only planner's merge-controlled loss is 12.8 and 6.1. Merging returns 8.1 points (CI +4.3 to +11.8) of this planner's 10.2 at 64 functions and 7.8 (CI +4.1 to +11.5) of 9.4 at 8. No margin was registered for this contrast, and against the $\delta = 2$ used elsewhere neither size shows the merged split within it. Before the run, the executions whose merged prompts happened to be cached had bounded the 64-function figure between 0.5 and 8.5 (`og2v2_v24_addenda.py`); the measured value lies inside. The registered Update 5 reading is unaffected, since it compares the registered split against MONO, and neither configuration beats one call on accuracy. The text of Update 6 quotes 26.8 and 108.3 tokens per entry and ratios of 0.35 and 1.07×; the ratios were character-count estimates and the per-entry figures had no committed generator, and the measured values are the ones above.

Model	functions	names-only SPLIT − MONO	full-schema SPLIT − MONO	full-schema MERGED − MONO	reading at δ=2
Gemma-12B	8	−14.0 (−19.2, −8.9)*	−9.3 (−13.7, −4.9)*	−1.9 (−4.9, +1.1)	inconclusive
	64	−14.2 (−20.3, −8.2)*	−5.9 (−10.9, −0.9)*	+1.1 (−2.6, +4.8)	inconclusive
Gemma-27B	8	−12.9 (−18.1, −7.8)*	−9.4 (−13.9, −4.9)*	−1.6 (−4.3, +1.1)	inconclusive
	64	−18.7 (−24.2, −13.2)*	−10.2 (−14.5, −5.9)*	−2.1 (−4.4, +0.1)	inconclusive
Llama-70B	8	−4.0 (−7.4, −0.7)*	−2.0 (−4.5, +0.5)	−0.1 (−2.0, +1.8)	within δ
	64	−6.7 (−11.1, −2.2)*	−4.3 (−8.3, −0.4)*	−0.3 (−3.5, +2.9)	inconclusive

Table 28: The catalogue cell on every model that has one (27B from Update 5, the others under Update 7 on 24 and 25 September, its text committed before their first call): the names-only planner's split and the full-schema planner's split and merge control against MONO at 8 and 64 padded functions, paired over tasks, with the reading of the merge-controlled loss against the pre-declared equivalence margin δ (within δ if the interval lies inside $(-\delta, \delta)$, a loss if its upper bound is below $-\delta$, otherwise inconclusive; `og2v2_update7_tables.py`).

catalogue	planner	MONO	SPLIT	loss (pts)	planner misses one
as shipped	names + descriptions	0.939	0.560	38.0	0.000
as shipped	full schemas	0.939	0.535	40.4	0.006
8	names + descriptions	0.935	0.536	39.9	0.003
8	full schemas	0.935	0.529	40.6	0.008
64	names + descriptions	0.919	0.499	42.0	0.065
64	full schemas	0.919	0.525	39.4	0.008

as shipped, names planner, every arm under each parser; contrasts against MONO *with paired* 95% *CI*

parser	MONO	SPLIT	MERGED	ONE CALL
frozen	0.939	0.560; −38.0 [−43.2, −32.8]	0.619; −32.0 [−36.9, −27.2]	0.941; +0.2 [−1.1, +1.5]
tolerant	0.939	0.845; −9.4 [−13.3, −5.5]	0.910; −2.9 [−5.1, −0.7]	0.941; +0.2 [−1.1, +1.5]
tolerant + Python-style	0.939	0.858; −8.2 [−12.2, −4.2]	0.921; −1.8 [−4.0, +0.3]	0.941; +0.2 [−1.1, +1.5]

Table 29: The fourth model (Update 6 Part B; `og2v2_qwen_cell.py`, `og2v2_qwen_controls.py`). Qwen2.5-72B-Instruct on BFCL fresh, the registered protocol throughout, served unpinned as the update named it (11 of 13,602 records came from a second provider; `og2v2_v21_addenda.py`). *Loss* is MONO minus SPLIT in points. The upper block is under the frozen parser, so most of each loss in it is the delimiter artifact described below. The lower block is H11's registered arms: every arm of the unpadded cell with its paired interval against MONO, including the merge control and the empty-candidate call, under the frozen parser (the registered reading) and, post hoc, under the same text with the call delimiter normalised and with a reader for Python-style calls added, each applied to all arms alike.

The fourth model, read. The gap between the two parsers is ours: our protocol sends a single user message and no system message, and without one this model's workers emit correct JSON inside `[calls]` where every prompt asks for `<calls>`, on 432 of the 463 worker slots (of 1,494, 31%) that produced no parseable call; one more wraps the instructed tag in brackets, 13 are explicit empty lists and 17 are Python-style calls, `[name({...})]`, with no delimiter and no JSON list. Normalising the delimiter recovers 433 slots, 29% of all; a reader for the Python style recovers the 17. A probe on 15 of the failing prompts resent each three ways (45 live calls, responses stored and replayed offline; `og2v2_qwen_system_probe.py`): as the harness sends it, 3 of 15 used the instructed delimiter, 8 used `[calls]` and 4 sent a bare JSON list; with a system message, 15 of 15; the monolithic prompt on the same tasks, 15 of 15. The monolithic arm uses the instructed delimiter on all 990 executions (989 yield a parseable call), so this is the worker prompt and the missing system message, not a broken chat template. Under the tolerant parser split minus MONO is −9.4 points (CI −13.3 to −5.5) and merged minus MONO −2.9 (CI −5.1 to −0.7): this planner enumerates repeated invocations as the Gemma planners do (358 of 990 executions carry a repeated name), so two thirds of its raw loss is our first-item-wins rule, and against the merge control the four models lose 7.1, 3.9, 3.9 and 2.9. That last interval does not survive reading the Python-style calls too: with them split minus MONO is −8.2 (CI −12.2 to −4.2) and merged minus MONO −1.8 (CI −4.0 to +0.3), not separated from zero, so this model's merge-controlled loss is 1.8 to 2.9 points depending on how much of its output is read. The update expected this model's raw gap to be small, near Llama's 3.9; it is not. H10 holds here under either parser as registered, a point comparison: the planner's miss rate at 64 functions, which does not depend on the call parser, falls from 0.065 to 0.008 with a full-schema catalogue, and the 8-to-64 growth of the loss is +2.1 (CI −1.0 to +5.2) with names against −1.2 (CI −4.1 to +1.7) with schemas under the frozen parser, and +2.8 (CI −0.2 to +5.8) against −1.9 (CI −4.5 to +0.6) under the tolerant one. Neither growth is separated from zero; their difference is, +3.3 (CI +0.4 to +6.3) and +4.7 (CI +1.9 to +7.6) (`og2v2_qwen_h10_tolerant.py`). Both readers are post hoc and specific to this model: applied to the 160,189 completions in the other fifteen caches, the tolerant parser changes the parse of none and the Python-style reader of two.

		gap		ONE CALL − MONO		G6	
Model	Set	reg.	delim.	reg.	delim.	reg.	delim.
12B	BFCL fresh	13.43	13.43	+0.81	+0.81	−4.65	−4.65
12B	BFCL test	10.96	10.96	−0.84	−0.84	−1.20	−1.20
12B	Live test	6.92	6.92	−2.56	−2.56	+2.56	+2.56
12B	Live fresh	15.83	15.83	−5.33	−5.33	−3.50	−3.50
27B	BFCL fresh	10.81	10.81	+2.22	+2.22	−1.92	−1.92
27B	BFCL test	8.55	8.55	+0.72	+0.72	+0.36	+0.36
27B	Live test	8.21	8.21	−1.54	−1.54	+0.77	+0.77
27B	Live fresh	15.00	15.00	+0.00	+0.00	−3.33	−3.33
Llama	BFCL fresh	3.94	3.94	−1.31	+0.71	−0.30	−2.22
Llama	BFCL test	3.49	3.37	+0.72	+1.81	−1.57	−2.53
Llama	Live test	−1.28	−1.28	+5.38	+5.38	−4.62	−4.62
Llama	Live fresh	3.17	3.17	−1.00	−0.83	+2.33	+2.17

Table 30: Three derived columns (the gap, ONE CALL minus MONO, and G6) from a re-scoring of all eleven arms in all twelve cells under a delimiter-aware parser; the 13 arm-cells that move are listed in `results/og2v2_parser_sensitivity.json` (post hoc; `og2v2_parser_sensitivity.py`; replay from cache, no new calls, and no step needed the registered parse as a fallback). The registered parser takes the span from the first `[` to the last `]` of a completion and ignores the `<calls>` tags every prompt in this study asks for, so a prose preamble containing a bracket, or a corrected second block, is read as an empty call list.

What the parser changes. We found the parser's behaviour by diagnosing it on a fourth model and then auditing our own three. Of the 132 arm-cells, 13 move, every one of them on Llama: ONE CALL by $+1.1$, $+2.0$ and $+0.2$ points and ten other arms by 0.1 to 0.2. Two of those ten are SPLIT and MERGED on BFCL test, which moves that cell's gap from 3.49 to 3.37; no other gap in this paper moves at all. The registered reading that moves is G6 on Llama: on the registered set from -0.3 (CI -2.8 to $+2.2$, inconclusive, the upper bound above δ=2) to -2.2 (CI -4.3 to -0.2, adds at most δ), and on BFCL test from -1.6 to -2.5. The exact figure depends on how a delimiter-aware parser treats a second block, at the 0.1-point level. We report both and claim neither as a correction: the registered reading stands as registered, and a reader should know it rests on a parser that does not read its own delimiter. At the completion level (`og2v2_parser_audit.py`, all sixteen released caches, comparing parsed content) the two parsers differ on 167 completions: 119 and 46 in Llama's two caches, 2 in the fourth model's, and none on either Gemma model. The caches hold 173,791 completions, but most are planner outputs or carry no call tag and cannot differ; of the 55,213 that contain a `<calls>` tag the rate is 0.3%, and on Llama 165 of 39,246, 0.4%.

model	variant	native MONO	native split	split − mono [95% CI]	DiD against text [95% CI]	native − text MONO	ladder (δ=2)
27B	`required`	0.323	0.489	+16.6 [+10.7, +22.4]	−27.4 [−33.9, −20.8]	−59.5	refuted[†]
27B	`auto`	0.866	0.767	−9.9 [−14.7, −5.1]	−0.9 [−3.9, +2.1]	−5.3	no verdict[‡]
Llama	`required`	0.903	0.889	−1.4 [−4.7, +1.8]	−2.5 [−5.9, +0.8]	−0.6	inconclusive[*]
Llama	`auto`	0.757	0.788	+3.1 [−1.3, +7.5]	−7.1 [−11.8, −2.3]	−15.3	refuted[*]

Table 31: The native anchor in all four (model, variant) cells, by one script (`og2v2_native_within_gemma.py`; read-only replay). *DiD* is the native gap minus the text gap on the same 198 tasks, negative when the split loses less inside the interface; *native − text* MONO says what the interface does to the baseline itself. Only the monolithic call and the split's workers are behind the API: the planner and first-item-wins are replayed from the text cache, so the aggregation artifact is common to both interfaces by construction, and these cells can speak only to the rest (Table 32 runs the merge control with native workers). H12 was registered for 27B under `required` only; the ladder is applied to the other three cells as our extension ([*]), and the DiD is not part of the registered reading. [‡]The 27B `auto` row receives no ladder verdict (on the raw numbers it would read *confirmed*): it is a raw reading of a cell whose payloads are corrupted too (60 monolithic responses and 48 split executions return no call, 43 of them the same executions). Its sign is stable, −11.7 points (CI −16.7 to −6.7) on the 925 executions where both arms return a call and −11.3 (CI −16.0 to −6.6) with no-call responses credited with the text outcome in both arms (`og2v2_native_merged.py`); its DiD is the difference of two corrupted numbers, and it does not measure the operator. [†]The registered cell, discarded as corrupted: half its monolithic tool calls come back with an empty function name (see below).

The native anchor, cell by cell. *27B, `required`* (registered): the split is +16.6 points above the native monolithic call, so H12 reads *refuted*. We discard the cell because the run is corrupted, and the evidence is the payloads: this provider returns 1070 of the monolithic call's 2168 tool calls with an empty function name under this setting, on 618 of 990 responses (311/330 *parallel*, 306/330 *parallel_multiple*, 1/330 *multiple*), and 502 of the workers' 2057 on 342 of 1517; under `auto` none of 2040 and none of 1914. The harness scored those as calls to no function, which is why MONO collapses while the split, whose calls are spread over several responses, is flattered. The cell cannot be repaired: only 160 of the 1070 carry any arguments, and 62 of those have an argument key set that fits exactly one function in the task's catalogue. There is no cap on calls per response: the longest 27B response carries six calls under `required` and five under `auto`. H12 therefore has no usable registered reading. *27B, `auto`*: the split loses 9.9 points, DiD −0.9 (CI −3.9 to +2.1), with a native baseline 5.3 points below the text one and 60 native monolithic and 48 native-split executions of 990 returning no call. 7.4 of those 9.9 points (7.1 of the 10.8 in text) sit on the 80 executions where first-item-wins bites (a repeated name at fan-out ≥ 2): there the text split and the native split score 0.000 on every execution against a native MONO of 0.91. We ran the native merge control (Table 32) expecting it to return most of those points, as it does in text. It does not: in text, merging lifts those executions from 0.000 to 0.85 and the arm by 6.9 points (CI +3.4 to +10.3); natively, to 0.30 and by 2.4 (CI +0.4 to +4.5), a difference of 4.4 (CI +1.3 to +7.6), and the native merged arm stays 7.5 points below native MONO (CI +3.1 to +11.8) against 3.9 in text, DiD +3.5 (CI −0.6 to +7.7). We first read that as native workers dropping items, and the payloads say otherwise. The 130 merged workers on those executions carry 265 plan items; as text they return 265 calls, natively 171, and 89 of them return exactly one. But those 89 responses are billed completion tokens that match every item carried: paired with the unmerged worker for the same function on the same execution, which was handed one item and returned one call, the ratio of billed tokens has median 2.00 (minimum 1.88) for the 84 two-item workers and 3.11 for the 5 three-item workers, every ratio is above 1.7, and the one call returned is identical to the unmerged worker's in 88 of 89; all 89 finish with `tool_calls` and none adds prose. The 41 workers that return every item bill the same (1.93×), and under `required`, where a merged worker that returns one call bills 1.00×, 62 of the 130 carry an arguments string that does not parse as one object. The raw generations are not in the payloads, so this is inference from the provider's own token counts: tokens for every item were billed and one call was returned. Whether the provider's parser alone loses the rest, or the worker emits its calls in a different form on a newline-joined subtask and the parser returns the first, the payloads cannot say; either way the loss is tied to the form of subtask our merge rule builds. On the 25 of the 80 executions where every merged worker's items all come back, the native merged arm scores 0.96, as native MONO does (text merged 0.76); on the other 55 it scores 0.00 against 0.89 and 0.89. Native workers do return repeated calls when one plan item asks for them (62 of the

71 unmerged workers whose single item covers a function the ground truth needs twice or more), so what is lost is lost on the newline-joined subtask the merge rule builds, between generation and the returned `tool_calls`. The baseline has the same defect: all 60 native monolithic responses that return no call have empty content and were billed 20 to 97 completion tokens (median 53), which is 6.1 points of executions under a baseline 5.3 below the text one. So under `auto` as under `required` calls that are billed at this provider's 27B endpoint do not come back, and the 7.5 and the DiD of +3.5 are raw readings of a corrupted cell, not the operator's cost inside the interface. On the other 910 executions, where the rule is inert and merging changes nothing, the native gap is 2.5 (CI −1.1 to +6.1) against 4.0 (CI +0.3 to +7.7) in text, DiD −1.5 (CI −4.6 to +1.6); on the 598 whose plan repeats no name at all, 4.1 (CI −0.5 to +8.7) against 4.8, DiD −0.7 (CI −4.5 to +3.2). There the anchor neither shows nor excludes an interface effect: the DiD half-widths, 3.1 where the rule is inert and 3.9 on plans that repeat no name, are of the order of the merge-controlled gap of 3.9 itself. *Llama,* `auto`: the native baseline collapses by 15.3 points and the gap with it (DiD −7.1, CI −11.8 to −2.3), which says the interface hurts the single call, not that it explains the split. *Llama,* `required`: the only cell whose native baseline is not separated from the text one (−0.6, CI −3.9 to +2.7) and whose payloads are clean; split minus MONO is −1.4 (CI −4.7 to +1.8) against −3.9 in text, DiD −2.5 (CI −5.9 to +0.8), and −3.9 (CI −7.5 to −0.3) on plans that repeat no name, one post-hoc cut among several. Both native arms carry a system message the text protocol never sends ("Call every tool needed to satisfy the user's request …" for the monolithic call, its one-tool analogue for a worker), so what is compared throughout is the interface plus that message. On Llama the merge control has almost nothing to act on: 3 of 990 executions carry a repeated name at fan-out ≥ 2, and the merged arm differs from the split by one execution of 990 under `auto` and by none under `required` (Table 32).

model	variant	native MONO	native split	native merged	merged − MONO	text merged − MONO	DiD
27B	`auto`	0.866	0.767	0.791	−7.5 [−11.8, −3.1]	−3.9 [−7.7, −0.2]	+3.5 [−0.6, +7.7]
27B	`required`†	0.323	0.489	0.494	+17.1 [+11.2, +22.9]	−3.9 [−7.7, −0.2]	−21.0 [−27.2, −14.8]
Llama	`auto`	0.757	0.788	0.789	+3.2 [−1.2, +7.6]	−3.9 [−7.0, −0.9]	−7.2 [−11.9, −2.4]
Llama	`required`	0.903	0.889	0.889	−1.4 [−4.7, +1.8]	−3.9 [−7.0, −0.9]	−2.5 [−5.9, +0.8]

Table 32: The merge control inside the tool-calling interface (unregistered; `og2v2_native_merged.py`, run on 21 September 2026 after the last review round: 266 new native worker calls, kept in cache files of their own so the released native caches are untouched; read-only replay thereafter). The registered text plan is replayed, the registered merge rule is applied to it, and the workers are native. *Text merged* − MONO is the registered merge control on the same tasks. *DiD* is the native merged gap minus the text merged gap, negative when the merge-controlled split loses less inside the interface. Neither 27B row measures the operator. Under `auto`, 89 of the 130 merged workers come back with one call while billed the completion tokens of all their items (2.0× the one-item worker's), and 60 monolithic responses come back empty after 20 to 97 billed tokens; where every item is returned the merged arm scores 0.96, as native MONO does. †The discarded cell; its +17.1 is the merged arm against native MONO, a different contrast from the split's +16.6 in Table 31, both raw readings of the same cell: 530 of its 2168 merged-arm calls carry an empty function name and 62 of its 130 merged workers an arguments string that does not parse. On Llama 3 of 990 executions carry a repeated name at fan-out ≥ 2, so the control has nothing to act on.

Model	dropped-item share	95% CI (task-clustered)	tasks with a loss event	tasks with a dropped item
12B	43.4%	[26.1, 60.4]	37	13
27B	49.2%	[30.5, 66.9]	36	14
Llama	0.0%	[0.0, 0.0]	21	0

Table 33: The dropped-item share of §5 with a task-clustered bootstrap (4,000 resamples of the tasks that carry a loss event; `og2v2_v21_addenda.py`). The decomposition is computed over executions, but the unit of independence everywhere else in this paper is the task, and few tasks carry the classes: 13 and 14 distinct tasks account for every dropped item on the Gemma models. The intervals are correspondingly wide and no committed test compares the dropped-item class with the residual or the wrong-set class, so "the largest class is our own aggregation rule" is a point estimate on both Gemma models; what is separated from zero is what merging returns (§4).

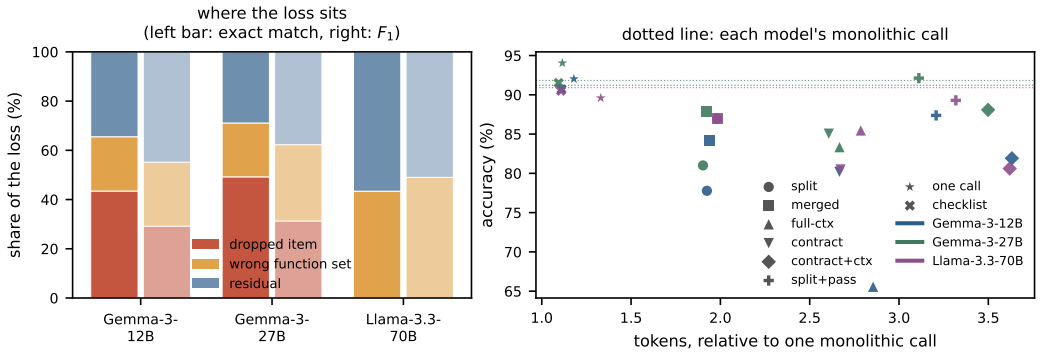

Figure 2: Left: where the split's loss sits, as a share of the loss, under the registered binary metric (solid) and per-call F_1 (pale), both under the class assignment of Table 35 (MONO right, SPLIT wrong; dropped item if MERGED is right), so the two bars decompose the same events. The ordering does *not* survive the change of metric on the Gemma models: the residual becomes the largest class (45% against the dropped item's 29% at 12B, 38% against 31% at 27B), because F_1 gives partial credit for the calls a dropped item leaves behind while a wrong argument still costs the whole call. On Llama the ordering holds. Table 40 also gives the shares under a looser, F_1-threshold class rule. Right: the plane of Figure 1, repeated here for reference, every arm as accuracy against token cost relative to that model's monolithic call (dotted lines), at the catalogue the benchmark ships; CHECKLIST is the cheapest arm other than MONO and reads no material effect against it (G9), one full-context call with an empty candidate list is next ($+0.8$, $+2.2$ and -1.3 points from MONO), and every split arm lies to the right of both and below the empty-candidate call.

Model	Set	executions	parse to an empty list
12B	BFCL fresh	990	5 (0.5%)
12B	BFCL test	830	2 (0.2%)
12B	Live test	390	16 (4.1%)
12B	Live fresh	600	51 (8.5%)
27B	BFCL fresh	990	6 (0.6%)
27B	BFCL test	830	1 (0.1%)
27B	Live test	390	6 (1.5%)
27B	Live fresh	600	4 (0.7%)
Llama	BFCL fresh	990	42 (4.2%)
Llama	BFCL test	830	17 (2.0%)
Llama	Live test	390	18 (4.6%)
Llama	Live fresh	600	14 (2.3%)

Table 34: The ONE CALL arm's empty-parse rate (`og2v2_v21_addenda.py`). §3 reports a fail-open rate of 0 to 1.9% for the pass arms; this arm is not in that table and its convention is different. The pass arms fail open to the base arm's call list, so an unreadable completion costs nothing; ONE CALL has no fallback, so an unreadable completion is an empty list and the exact-match metric zeroes it. Six of twelve cells exceed the 1.9% ceiling, the worst at 8.5%. This bounds the arm the practical rule recommends, and it is the same parser described in Table 30.

Table 35: Anatomy on BFCL fresh (990 executions per model; post hoc, committed scripts). Executions with MONO right and SPLIT wrong, classified as *dropped item* (the merge control is right on the same execution), *wrong function set* (emitted functions differ from the ground truth) or *residual* (right functions, wrong counts or arguments); entries are count (share of loss events; points). *Gains*: executions where only the split is right; the net gap is loss events less gains, in points.

Model	exec.	dropped item	wrong function set	residual	all loss events	gains	net gap
Gemma-12B	990	63 (43%; 6.4)	32 (22%; 3.2)	50 (34%; 5.1)	145 (14.6)	12 (1.2)	13.4
Gemma-27B	990	63 (49%; 6.4)	28 (22%; 2.8)	37 (29%; 3.7)	128 (12.9)	21 (2.1)	10.8
Llama-70B	990	0 (0%; 0.0)	23 (43%; 2.3)	30 (57%; 3.0)	53 (5.4)	14 (1.4)	3.9

Is the dropped-item class real, and what nets out. The dropped-item rule is defined by another arm's outcome, so it could in principle catch a lucky re-sample. A directional test cannot settle that: across the 79, 80 and 3 executions where merging changes the output the split is right on *none*, so a right-to-wrong flip has no chance to occur. What settles it uses only the split arm's own plan and output: a loss event is a dropped item if the emitted function set is right, the split under-emits, and the deficit is no larger than the number of plan items first-item-wins discarded. That rule gives 75/68/0, a strict superset of the merge-defined 63/63/0 with no merge-only event; and every merge-defined event is an under-emission with the right function set and a repeated-name plan, none of them at the correct call count, so none can be a merged draw that happened to fix an argument. The three classes also never co-occur (dropped $\wedge$ wrong-set $= 0$ on every model), though the cascade had at most 2/0/4 opportunities to make them, since merging cannot change which functions the planner selected (`og2v2_v20_addenda.py`). The table's events are gross: both arms err at $T = 0.7$, and the gains are mostly the monolithic call's own argument errors (9 of 12, 12 of 21, 11 of 14), so a class the two arms fail in equally often fills the table and adds nothing to the gap. Netted (failures of SPLIT minus failures of MONO on the same executions, classified wrong set, then wrong count, then wrong argument; task-clustered bootstrap, `og2v2_v23_addenda.py`) the gap of 133/107/39 executions is count $+74$ (CI 38 to 113), $+77$ (42 to 115), $+2$ (-8 to 13); wrong set $+41$ (18 to 68), $+15$ (-11 to 42), $+22$ (4 to 44); argument $+18$ (-9 to 46), $+15$ (-10 to 41), $+15$ (-9 to 40), intervals from 100,000 task resamples. In this netting of the raw split the argument channel is separated from zero on no model, but the classification order books every dropped-item execution as a count failure and so hides its argument errors. Against the merge control (net 70/39/39) the channels are wrong set $+41/+15/+22$, count $-1/-1/-1$ and argument $+30$ (CI 0 to 62), $+25$ (0 to 52) and $+18$ (-5 to 42): the largest channel of the 27B residual, its lower bound at zero as at 12B. The wrong function set is the only Llama channel separated from zero; it is $+22$ or $+23$ in the split, the merge control, the full-context arm and its filtered version alike because all four reuse one plan, and it is not separated from the argument channel ($+7$, CI -24 to $+40$). Planner selection and hand-off coverage are in the tables above.

			all fan-out ≥ 2		parallel_multiple only		
Model	Set	share fan-out ≥ 2	against SPLIT	against MERGED	against SPLIT	against MERGED	overall
12B	BFCL fresh	0.35	37.0 [25.3, 48.6]	19.7 [10.1, 29.4]	32.8 [20.6, 45.0]	13.1 [4.2, 22.0]	13.4
12B	BFCL test	0.30	36.4 [23.2, 49.6]	18.5 [7.8, 29.1]	32.4 [18.5, 46.4]	10.2 [1.5, 19.0]	11.0
12B	Live test	0.29	36.7 [15.4, 57.9]	36.7 [15.4, 57.9]	n/a	n/a	6.9
12B	Live fresh	0.24	67.5 [52.5, 82.6]	67.5 [52.5, 82.6]	n/a	n/a	15.8
27B	BFCL fresh	0.35	32.2 [20.2, 44.3]	13.3 [3.0, 23.7]	28.3 [15.8, 40.8]	7.4 [−2.4, 17.2]	10.8
27B	BFCL test	0.31	28.8 [14.5, 43.2]	5.4 [−6.2, 17.0]	24.4 [9.3, 39.5]	−2.7 [−12.7, 7.4]	8.6
27B	Live test	0.27	38.0 [20.6, 55.4]	38.0 [20.6, 55.4]	n/a	n/a	8.2
27B	Live fresh	0.28	53.3 [39.2, 67.5]	53.3 [39.2, 67.5]	n/a	n/a	15.0
Llama	BFCL fresh	0.33	11.5 [3.8, 19.3]	11.5 [3.8, 19.3]	10.2 [2.8, 17.5]	10.2 [2.8, 17.5]	3.9
Llama	BFCL test	0.29	9.3 [1.0, 17.6]	6.1 [−1.0, 13.2]	7.8 [−0.1, 15.7]	4.4 [−2.0, 10.7]	3.5
Llama	Live test	0.19	12.9 [−12.0, 37.9]	12.9 [−12.0, 37.9]	n/a	n/a	-1.3
Llama	Live fresh	0.10	72.5 [48.9, 96.1]	72.5 [48.9, 96.1]	n/a	n/a	3.2

Table 36: The loss conditional on the planner's fan-out (post hoc; `og2v2_fanout_loss.py`), task-weighted, with the paired 95% CI over tasks that contribute at least one qualifying execution. The artifact carries both weightings and §4 quotes the execution-weighted ones for Live fresh, where they differ most (66.7/55.4/62.7 against the 67.5/53.3/72.5 tabulated here); on the registered set the two agree to within 1.6 points against SPLIT and 2.6 against MERGED (execution-weighted 17.7/10.8/11.2). *Against* MERGED is the same conditional with the merge control in place of the split, which is how the paper says the operator should be costed: 19.7, 13.3 and 11.5 on the registered set, where the three models are no longer separated, and with *multiple* excluded, whose fan-out is itself the planner's over-selection error, 13.1, 7.4 (CI −2.4 to +17.2) and 10.2; the column repeats the split's digit for digit in the seven cells where merging changes no execution's outcome (Llama on BFCL fresh and every Live cell). The operator is a genuine split only where the planner fans out, so the conditional is the loss of the split as such; the *parallel_multiple* columns are the fan-out ≥ 2 executions within that category, not the whole category, and the two conditioning sets differ because on the curated sets almost all fan-out sits in *parallel_multiple*. Both condition on an output of the split arm's own planner, so neither is the counterfactual accuracy of a policy that routes on predicted fan-out.

Live fresh, conditional on splitting. Conditional on splitting, this design does not separate the three models (+66.7/+55.4/+62.7 points, execution-weighted; Llama minus 12B −4.0, CI −25.6 to +18.7, joint task bootstrap, `og2v2_v23_addenda.py`), so what differs on that set is the planner's fan-out rate: standardizing Llama's rate to each Gemma model's gives 12.9 and 14.9 against their observed 15.8 and 15.0. At fan-out 1 the split is not separated from MONO (0.792 against 0.786; 0.832 against 0.827; 0.800 against 0.767); where it fans out, the extra worker emits a call rather than the empty list it was told to return in 97%/86%/100% of those executions, and an oracle that removes those calls leaves 21, 8 and 7 points, none separated from zero over tasks.

		pass		full-ctx		contract		contract+ctx	
Model	Set	band	power	band	power	band	power	band	power
12B	BFCL fresh	30 to 70%	1.00	7 to 93%	0.63	16 to 84%	0.82	15 to 85%	0.80
12B	BFCL test	26 to 74%	0.98	-5 to 105%	0.43	10 to 90%	0.68	9 to 91%	0.67
12B	Live test	-18 to 118%	0.30	-52 to 152%	0.16	-41 to 141%	0.19	-60 to 160%	0.15
12B	Live fresh	27 to 73%	0.99	20 to 80%	0.90	16 to 84%	0.83	15 to 85%	0.81
27B	BFCL fresh	25 to 75%	0.97	12 to 88%	0.73	14 to 86%	0.78	17 to 83%	0.85
27B	BFCL test	17 to 83%	0.84	4 to 96%	0.57	3 to 97%	0.54	1 to 99%	0.51
27B	Live test	7 to 93%	0.62	-26 to 126%	0.25	-30 to 130%	0.23	-18 to 118%	0.30
27B	Live fresh	27 to 73%	0.99	20 to 80%	0.90	19 to 81%	0.88	18 to 82%	0.86
Llama	BFCL fresh	4 to 96%	0.57	-14 to 114%	0.34	-45 to 145%	0.18	-48 to 148%	0.17
Llama	BFCL test	-10 to 110%	0.37	-1 to 101%	0.48	-57 to 157%	0.15	-29 to 129%	0.24
Llama	Live test[d]	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Llama	Live fresh	-55 to 155%	0.15	-56 to 156%	0.15	-77 to 177%	0.12	-94 to 194%	0.11

Table 37: What the registered ladder could resolve (post hoc; `og2v2_power_ladder.py`). *band*: the true recovery shares whose readings the ladder cannot separate from each other, $\frac{1}{2}\pm$ half the interval over the cell's gap; *power*: the probability the ladder would have read "recovers the majority" against an arm that recovered the entire gap (noncentral t, two-sided, $\alpha = 0.05$, observed per-task SD). [d]excluded by the registered 3-point rule. The rule was fixed by hand and is too permissive: at these SDs a majority-recovery test needs a gap of 5 to 16 points (registered cells; median 11.8 over all cells) before it reaches 80% power against complete recovery, so Llama's fresh cell, gap 3.9, was admitted to hypotheses it could not confirm and its negatives are readings the design could not have overturned. Over the 44 admitted readings the ladder returns *recovers* 5 times, *does not* 13 and *inconclusive* 26; over the three repair arms alone (G5, G7, G8) it is 0, 13 and 20, of which the nine readings inside the registered families, on BFCL fresh, are 0, 4 and 5 and the other 24 are replication cells read by the same ladder without Holm protection.

Model	Set	exact match	per-call F_1	F_1 95% CI	MERGED − MONO, F_1 [95% CI]
12B	BFCL fresh	13.4	5.4	[2.8, 8.0]	−3.20 [−5.59, −0.80]
12B	BFCL test	11.0	4.1	[1.9, 6.3]	−2.80 [−4.87, −0.73]
12B	Live test	6.9	3.4	[−2.6, 9.4]	−3.38 [−9.38, +2.61]
12B	Live fresh	15.8	9.8	[3.5, 16.1]	−9.82 [−16.13, −3.52]
27B	BFCL fresh	10.8	4.6	[2.0, 7.2]	−2.57 [−5.02, −0.12]
27B	BFCL test	8.6	2.2	[−0.4, 4.8]	−0.35 [−2.78, +2.08]
27B	Live test	8.2	3.9	[0.1, 7.7]	−3.90 [−7.70, −0.11]
27B	Live fresh	15.0	8.0	[3.2, 12.7]	−7.96 [−12.67, −3.25]
Llama	BFCL fresh	3.9	3.0	[0.9, 5.1]	−2.91 [−5.01, −0.82]
Llama	BFCL test	3.5	1.8	[0.2, 3.4]	−1.61 [−3.19, −0.04]
Llama	Live test	−1.3	−4.1	[−10.1, 2.0]	+4.08 [−1.98, +10.13]
Llama	Live fresh	3.2	0.5	[−5.4, 6.4]	−0.50 [−6.37, +5.37]

Table 38: The gap under both metrics on all four sets (post hoc, replay, no new calls; `og2v2_graded_all_sets.py`). The graded scorer reproduces every registered per-execution outcome in all twelve cells, so the two columns differ only in how a partly-right list is credited. The registered metric is binary per execution, so one surplus or missing call zeroes a task; on the real-user sets, whose loss is dominated by over-selection, that accounts for roughly half the reported gap, and Llama's Live-fresh loss is 0.5 points under F_1. The last column is the doubly corrected loss, the merge control against MONO under F_1: 3.2, 2.6 and 2.9 on the registered set, all excluding zero, but on BFCL test 2.8, 0.4 (CI −2.1 to +2.8) and 1.6 (lower bound +0.04), so the 27B figure does not replicate and Llama's barely does; on the real-user sets merging changes nothing and the column repeats the F_1 gap.

What the multiplicity controls, and what it does not. Holm is applied within each (model, family) block of four on BFCL fresh, so familywise error is controlled per block, not across the six blocks; over all 24 registered tests at once the study-wide rate is higher, and there 27B's G1 reads 1.2×10^{-3} and 12B's 0.543. Two further caveats. G6 and G9 are read against pre-specified margins from *unadjusted* intervals, so those two verdicts carry no familywise protection. And 7 of the 12 G0 cells

are identities rather than tests: the merge control returns the split's outcome on every execution, so G0 equals $-\frac{1}{2}$gap by construction (Llama on three of its four sets, all but BFCL test, where merging changes eight executions; and both Gemma models on the two real-user sets). The reason is not that those planners never repeat a name: Llama's does on 183 of its 990 fresh executions, but 180 of those plans have fan-out 1, where the worker is handed the full request and the rule has nothing to drop (`og2v2_v23_addenda.py`). Apart from Llama's G2, marked in Table 2 as not met after Holm, no conclusive verdict in that table changes under Holm within the registered families (the alternative schemes were run for G1 only). The sensitivity analysis registered in Update 2 agrees throughout: across all 24 registered tests the sign-flip p never crosses 0.05 on the other side of the studentized p, the largest discrepancy being Llama's G9 (0.416 against 0.562). Computed by `og2v2_multiplicity.py`.

Can a split worker abstain? §4 reports that an over-selected split worker emits rather than returning the empty list in 97%/86%/100% of executions. That worker never sees the request: its only referent is a subtask the planner was told to write as a "self-contained request for only that function", which is an instruction to call. The full-context arm gives the same worker the request and the same escape, and on the same Live-fresh executions its surplus workers emit in 89%/47%/86% of slots, against 92%/84%/100% of slots for the split (the 97%/86%/100% above is per execution). At 27B, information nearly halves spurious emission. Spurious emission is most of the Live-fresh conditional loss and not all of it: removing every call to a function outside the ground truth by oracle lifts the split at fan-out ≥ 2 from 0.020/0.120/0.000 to 0.476/0.590/0.559 against MONO's 0.687/0.675/0.627, and what is left (21.1/8.4/6.8 points execution-weighted; paired over tasks 18.0, CI -0.1 to $+36.1$, 4.8, CI -7.7 to $+17.3$, and 10.0, CI -17.5 to $+37.5$, all three covering zero) is consistent with the needed worker answering a paraphrase instead of the request (`og2v2_v23_addenda.py`). So the split figure is a property of the arm's information partition, not a disposition of the worker, and it is the one place in this study where full context clearly repairs *selection* (`og2v2_guard_and_abstention.py`).

Model	Set	gap	(a) merge	(b) filters	(c) one call, with 95% CI
12B	BFCL fresh	13.4	+6.4	+10.6	+14.2 [+9.3, +19.2]
12B	BFCL test	11.0	+5.8	+4.9	+10.1 [+4.8, +15.4]
12B	Live test	6.9	+0.0	−3.3	+4.4 [−3.7, +12.4]
12B	Live fresh	15.8	+0.0	+0.7	+10.5 [+2.9, +18.1]
27B	BFCL fresh	10.8	+6.9	+7.0	+13.0 [+8.4, +17.6]
27B	BFCL test	8.6	+7.3	+6.4	+9.3 [+4.2, +14.4]
27B	Live test	8.2	+0.0	+5.4	+6.7 [−1.5, +14.8]
27B	Live fresh	15.0	+0.0	+7.2	+15.0 [+9.4, +20.6]
Llama	BFCL fresh	3.9	+0.0	−0.6	+2.6 [−1.2, +6.5]
Llama	BFCL test	3.5	+1.0	+1.8	+4.2 [+1.0, +7.5]
Llama	Live test	-1.3	+0.0	−0.8	+4.1 [−3.4, +11.6]
Llama	Live fresh	3.2	+0.0	−1.5	+2.2 [−3.7, +8.0]

Table 39: Each clause of the practical rule in §6, in points over SPLIT, in all twelve cells: (a) merging repeated-name items, (b) full context with the two zero-LLM filters, (c) one full-context call with no candidates. (a) is positive in 5 cells and exactly zero in the other 7, where merging changes no execution's outcome, (b) in 8 and negative in 4 while costing 1.7 to 2.9× the tokens (2.7 to 2.9× on BFCL fresh), (c) in all twelve, with the interval excluding zero in seven of them. Only (c) is positive everywhere; the bracketed interval is the paired 95% CI for (c).

	12B		27B		Llama	
Arm	exact	F1	exact	F1	exact	F1
MONO	0.912	0.942	0.918	0.951	0.909	0.949
SPLIT	0.778	0.888	0.810	0.905	0.870	0.919
MERGED	0.841	0.910	0.879	0.925	0.870	0.920
SPLIT+PASS	0.874	0.924	0.921	0.953	0.893	0.938
ONE CALL	0.920	0.948	0.940	0.962	0.896	0.920
FULL-CTX	0.656	0.852	0.833	0.927	0.855	0.924
CONTRACT	0.802	0.884	0.851	0.913	0.805	0.878
CONTRACT+CTX	0.819	0.888	0.881	0.929	0.806	0.887
gap, MONO minus SPLIT (pts)	13.4	5.4	10.8	4.6	3.9	3.0
share of the F1 deficit, registered classes: dropped / wrong set / residual	29/26/45%		31/31/38%		0/49/51%	
the same under the F1-threshold rule: dropped / wrong set / residual	38/25/37%		38/30/33%		1/42/56%	
MERGED minus MONO (pts)	-7.1	-3.2	-3.9	-2.6	-3.9	-2.9
SPLIT+PASS recovery contrast, G1 (pts)	$+2.9$	$+0.9$	$+5.7$	$+2.5$	$+0.4$	$+0.4$

Table 40: Eight of the eleven arms re-scored under a graded metric (post hoc, no new executions; CONTRACT+RC, CHECKLIST and MONO+PASS are omitted). Predicted and ground-truth calls are matched one-to-one by the same schema-aware pairwise checker the registered scorer uses, and $F_1 = 2PR/(P+R)$ with P over predicted and R over ground-truth calls; exact match is the special case $F_1 = 1$ with equal counts, and the graded scorer reproduces all 23,760 registered outcomes. The registered metric is binary per execution, so one surplus or missing call zeroes the task, which is exactly what the two largest loss classes produce; the comparison shows how much of the gap that costs. The two share rows use two different class assignments: the first is the assignment of Table 35 (MONO right, SPLIT wrong; dropped item if MERGED is right), re-credited by F_1 deficit, with 63/32/50, 63/28/37 and 0/23/30 events; the second books an event on any positive F_1 deficit and a dropped item on any positive F_1 gain from merging, giving 75/32/43, 73/28/36 and 3/23/43. The looser rule is what moves the dropped-item class above the residual on the Gemma models and gives Llama a class the registered metric puts at zero. The last two rows re-read the contrasts the paper leans on: the merge control against MONO halves but keeps intervals clear of zero (-3.20, CI -5.59 to -0.80; -2.57, CI -5.02 to -0.12; -2.91, CI -5.01 to -0.82), and the G1 recovery contrast keeps its sign and its asymmetry, holding at 27B (CI $+1.2$ to $+3.9$) and going null at 12B (CI -0.4 to $+2.2$) and on Llama (CI -0.9 to $+1.7$), which is the reading we claim.

Request (BFCL fresh, *parallel_multiple_31*): *"Find how many cases and the judge handling a specific lawsuit for Pacific Gas and Electric and Tesla Inc."*

Planner output (Gemma-3-12B, sample 0; one item per invocation although the prompt says "use each function name at most once and combine all repeated invocations"):

1. `lawsuit.fetch_details`: Fetch the number of cases for Pacific Gas and Electric. 2. `lawsuit.fetch_details`: Fetch the number of cases for Tesla Inc.

3. `lawsuit.judge`: Fetch the judge handling the lawsuit for Pacific Gas and Electric. 4. `lawsuit.judge`: Fetch the judge handling the lawsuit for Tesla Inc.

SPLIT (registered first-item-wins): items 2 and 4 are dropped; each worker receives one case and emits one call → 2 calls, wrong. **MERGED**: each worker receives both lines of its function and emits two calls → 4 calls, right. **MONO**: 4 calls, right.

Full context (*parallel_multiple_33*, Gemma-3-12B, sample 0): *"Locate all supermarkets in Los Angeles and find the most popular site seeing place in Miami."* The `sightseeing.popular_in_city` worker, holding the request and every schema, emits the identical call {city: Miami, state: Florida, kidsFriendly: true} twice, and the `supermarket.find_in_city` worker adds a third sightseeing call of its own: four calls for a two-call request, wrong. Dropping the leaked call leaves three, still wrong; removing the exact duplicate leaves the right two.

Figure 3: Upper: the dropped-item mechanism on one fresh-set execution; the worker's "one call" is a faithful answer to the one case it was handed. Lower: the verbatim-duplicate emission of a fully informed worker.

arm	overall	mult.	par.	par. mult.	vs. MONO (pts, CI)
native, `auto`	0.757	0.945	0.712	0.612	-15.3 [-20.1, -10.4]
native, `required`	0.903	0.891	0.955	0.864	-0.6 [-3.9, $+2.7$]
native split, `auto`	0.788	0.918	0.715	0.730	-12.1 [-16.7, -7.5]
native split, `required`	0.889	0.927	0.958	0.782	-2.0 [-5.4, $+1.4$]
MONO (text)	0.909	0.933	0.945	0.848	n/a
SPLIT (text)	0.870	0.894	0.964	0.752	n/a
SPLIT+PASS (text)	0.893	0.918	0.948	0.812	n/a

Table 41: Native tool-calling anchors on Llama-3.3-70B, BFCL fresh, $k = 5$ at $T = 0.7$, 990 executions per arm and variant, the same pinned build the registered pass was served (`Llama-3.3-70B-Instruct-Turbo`). *native* is the monolithic call through the provider's API; Within the interface the two native arms are not separated: the native split minus the native monolithic call is -1.4 (CI -4.7 to $+1.8$) forced and $+3.1$ (CI -1.3 to $+7.5$) left to choose. That within-interface contrast was computed after the update's declared reading came out inconclusive, on the same executions and with no new calls; its difference in differences against the text gap is -2.5 (CI -5.9 to $+0.8$) forced and -7.1 (CI -11.8 to -2.3) left to choose, the second because the native monolithic call collapses, not because the split improves (Table 31). Left to choose, the native monolithic call scores 0.757 (-15.3 against MONO, CI -20.1 to -10.4) and the native split 0.788 (8.2 points below our text split); forced to emit, 0.903 and 0.889. *native split* is the registered split with its workers behind that API, the planner replayed from the committed cache so the plan, its merge and the execution order are byte-identical to SPLIT. Unregistered and descriptive, declared with its train pilot in a third registration update before it was run. The schemas go to the provider's `tools` array and the calls are read from `tool_calls`; BFCL's type vocabulary is mapped to JSON Schema, and the 194 of 336 distinct function names that carry a dot (233 of the 425 tool entries built across the 198 tasks) are sanitized for the request and mapped back before scoring. Under `auto` the native monolithic call's largest failure class is a wrong emitted function set (101 of 241 failures), then argument errors (69) and too few calls (63); it answers in prose in 4 executions. Forcing the emission leaves argument errors (65) dominant, cuts the wrong-set class to 9, and trades them against over-emission (21).

Model	G5 registered	G5 leak-filtered	G5 filtered+dedup	recovery (filtered+dedup)
Gemma-12B	-18.9 (-24.7, -13.2)	-10.9 (-16.2, -5.5)	$+3.9$ ($+0.6$, $+7.2$)	79%
Gemma-27B	-3.1 (-7.2, $+1.0$)	-3.1 (-7.2, $+1.0$)	$+1.6$ (-1.7, $+4.9$)	64%
Llama-70B	-3.5 (-6.0, -1.0)	-2.6 (-4.9, -0.3)	-2.6 (-4.9, -0.3)	-15%

Table 42: Contrasts for the filtered full-context arm: the G5 contrast under the registered arm and under each filter, and the filtered arm's recovery share.

Model	filtered+dedup − mono	filtered+dedup − merged	dedup effect on mono / split / merged (pts)
Gemma-12B	-2.8 (-6.0, $+0.4$)	$+4.2$ ($+0.4$, $+8.1$)	+0.3 / +0.0 / +0.0
Gemma-27B	-3.8 (-7.5, -0.2)	$+0.1$ (-3.1, $+3.3$)	+0.1 / +0.0 / +0.0
Llama-70B	-4.5 (-7.3, -1.8)	-0.6 (-3.3, $+2.1$)	+0.0 / +0.1 / +0.1

Table 43: The filtered full-context arm against MONO and against the merge control, and the effect of deduplication on the other arms (a no-op check).

Model	failures	own duplicate	foreign superset	other wrong set	under / mixed	argument	with a foreign-emitting worker
Gemma-12B	265	247	8	5	0	5	182
Gemma-27B	107	87	0	14	0	6	6
Llama-70B	86	25	11	8	0	42	12

Table 44: Full-context failures on parallel_multiple by emission pattern (own duplicate: right function set, a function over-emitted; foreign superset: a call to a function outside the ground truth; other wrong set; under or mixed; argument), and failures in which some worker emitted a call to a non-assigned function.

Model	surplus slots	exact copies	distinct	failures	fixed by leak filter	fixed by filter+dedup
Gemma-12B	305	265	40	265	76	215
Gemma-27B	111	74	37	107	0	46
Llama-70B	31	9	22	86	9	9

Table 45: Surplus worker slots on parallel_multiple (own-function calls beyond the ground-truth count) split into exact copies and distinct calls, and full-context failures fixed by each filter.

How often an arm emits a byte-identical duplicate. 269, 63 and 21 full-context executions carry one, against at most 13, 15 and 9 in any other arm (the contract arms) and 3, 1 and 4 in MONO (`og2v2_v23_addenda.py`). **A one-factor version of the information arm.** A one-factor arm, own schema plus request plus subtask, run on train under the earlier harness, reads $+1.0$, $+1.7$ and -1.6 over the split (Appendix B): the direction survives a cleaner instrument.

Model	n (counts right)	contract+ctx	contract	merged	mono	contract+ctx − mono (paired over tasks)
Gemma-12B	896	0.895	0.878	0.858	0.919	-1.8 (-5.4, $+1.7$)
Gemma-27B	927	0.941	0.908	0.892	0.929	$+1.7$ (-1.1, $+4.4$)
Llama-70B	926	0.862	0.861	0.887	0.914	-5.2 (-8.5, -1.9)

Table 46: Contract arms conditional on the manifest planner's declared counts (post hoc): execution-level accuracy of CONTRACT+CTX, CONTRACT, MERGED and MONO on the same executions when the declared counts equal the ground truth, and the paired task-level contrast of CONTRACT+CTX − MONO on those executions. Splitting that subset again by the registered roster rule ($\geq 90\%$ coverage), computed as the string-argument proxy of Appendix B, gives $+3.8$ over MONO (CI $+0.4$ to $+7.3$) where the roster and subtask text held the ground-truth strings and -5.9 (CI -13.1 to $+1.2$) where they did not, at 27B (at 12B a small and on Llama a large residual remains in both halves); at complete coverage, a stricter rule we did not register, $+4.6$ (CI $+1.4$ to $+7.8$) and -7.9 (CI -15.9 to $+0.1$). The statistic is undefined on 230 executions per model; folding those into the upper stratum gives $+3.4$ (CI $+0.6$ to $+6.2$) under the registered rule and $+3.9$ (CI $+1.3$ to $+6.5$) under the complete one. These are unadjusted post-hoc contrasts, several per model, on subsets selected by the treatment arm's own planner, and coverage stratifies the roster-free merge control too (-3.0 against -7.0 points to MONO at 27B), so part of the swing is the task and not the roster (`og2v2_guard_and_abstention.py`).

Model	n par.-mult. (counts right)	contract+ctx	mono	n (counts wrong)	contract+ctx	contract	merged	mono
Gemma-12B	280	0.864	0.929	94	0.096	0.074	0.681	0.851
Gemma-27B	300	0.937	0.913	63	0.000	0.000	0.683	0.762
Llama-70B	302	0.762	0.854	64	0.000	0.000	0.625	0.844

Table 47: The same conditioning on parallel_multiple, and accuracy of the four arms on executions whose declared counts are wrong.

Model	executions	failing parameters	omitted	unexpected	wrong value	string	number	bool / other
Gemma-12B	47	61	13	5	43	25	7	29
Gemma-27B	36	62	10	3	49	33	1	28
Llama-70B	29	41	14	4	23	24	3	14

Table 48: Failing parameters in the merge control's argument-class failures with MONO right (post hoc; pinned scorer's value comparison, best assignment over same-name calls): kind of failure and value type.

Model	in request, not in subtask	in subtask	in neither	strings: lost	strings: present	strings: neither
Gemma-12B	31	12	18	16	4	5
Gemma-27B	15	35	12	5	24	4
Llama-70B	8	13	20	8	9	7

Table 49: Provenance of the same failing parameters: whether an accepted value appears in the request but not in the worker's subtask (paraphrase loss), in the subtask (the worker had it), or in neither (booleans, defaults, normalized forms); the last three columns restrict to string-valued parameters.

Model	mult.	par.	par.-mult.	compliant	acc. | counts right (n)	acc. | counts wrong (n)
Gemma-12B	97%	90%	85%	98%	0.878 (896)	0.074 (94)
Gemma-27B	95%	95%	91%	100%	0.908 (927)	0.000 (63)
Llama-70B	95%	94%	92%	99%	0.861 (926)	0.000 (64)

Table 50: Contract diagnostics: declared counts equal to the ground truth by category, worker count compliance, and accuracy conditional on the declared counts being right or wrong.

Model	mult.	par.	par.-mult.	roster coverage
Gemma-12B	10/39	31/54	46/103	0.883
Gemma-27B	18/44	15/30	30/74	0.883
Llama-70B	16/52	20/36	28/105	0.900

Table 51: Contract failures whose declared count was wrong (by category) and the mean roster-plus-subtask coverage of ground-truth string values (the "planner solved it" statistic).

Model	failures (counts right, no leak, mono right)	string value outside accepted	verbatim in roster	verbatim in request	merged right	contract right	split right
Gemma-12B	42	35	34	31	28	13	27
Gemma-27B	17	11	9	7	5	2	5
Llama-70B	56	33	32	33	29	11	29

Table 52: Transcription in CONTRACT+CTX: failures with correct declared counts, count-compliant workers, no leaked call and a right monolithic call; how many carry a string argument outside the accepted values; how many of those copy the roster's phrasing and the request's phrasing verbatim; and how many of them the merged, contract and split arms get right.

Model	added	removed	arg.-modified	wrong→right	right→wrong	wrong split outputs left unchanged	of which one call right
Gemma-12B	142	25	20	95	0	97/220	54
Gemma-27B	145	35	38	112	2	68/188	34
Llama-70B	23	32	133	39	16	46/129	11

Table 53: The pass on split outputs: edits, outcome flips, and wrong outputs it left unchanged (with how many of those the empty-candidate call gets right). "Unchanged" compares the two call lists as lists, while the scorer ignores order. Compared as multisets the counts are 105 (55), 68 (34) and 65 (26) against the tabulated 97 (54), 68 (34) and 46 (11): 8 and 19 wrong outputs at 12B and on Llama are returned reordered and otherwise untouched, so the table understates how often Llama's pass leaves a wrong list alone. Of the multiset-unchanged wrong outputs, 66/58/61 are argument errors (`og2v2_v23_addenda.py`).

Model	fail-open split+pass	fail-open mono+pass	mono (s)	split (s)	split+pass (s)	one call (s)
Gemma-12B	0.5%	0.5%	3.8	7.5	11.0	4.0
Gemma-27B	0.0%	0.5%	5.1	9.7	13.7	4.4
Llama-70B	1.6%	1.6%	1.6	3.4	5.1	2.6

Table 54: Pass fail-open rates and counterfactual median latency (planner plus slowest worker, plus the pass) from per-call timings.

E.1 Detail behind Section 4

The paragraphs below stood in Section 4 and were moved here, with only their lead-ins edited, so that the section states each reading once; every number is produced by the committed scripts named in the text.

Two more comparisons on the pass. Two comparisons the cache already contains say the same thing from the other side (post hoc, `og2v2_v26_addenda.py`): the pass over *monolithic* candidates is $+0.4$, -1.1 and $+0.9$ points from the same call with no candidates, all three intervals covering zero; and a pass over the *merge control's* candidates, scored where its prompt is already cached and credited with a success everywhere else, cannot exceed 0.890 and 0.924 on the Gemma models against ONE CALL's 0.920 and 0.940. Candidates from a better source would not rescue the pass.

The filtered full-context arm on the other sets. The 12B reading does not replicate: the same filters recover 45%, -48% and 4% on the other three sets, against 75%, 66% and 48% at 27B. The arm also moves five boundaries at once (Appendix E).

The count contract, conditioned on the planner's count. Conditioning on the planner (post hoc) locates much of the contract arms' loss. When the declared counts equal the ground truth (896/927/926 executions), CONTRACT+CTX scores 0.895/0.941/0.862 against MONO's 0.919/0.929/0.914 on the same executions (paired over tasks: -1.8, CI -5.4 to $+1.7$; $+1.7$, CI -1.1 to $+4.4$; -5.2, CI -8.5 to -1.9); when they do not (94/63/64), it scores 0.096/0.000/0.000 against 0.851/0.762/0.844. At 27B, on the executions where the planner counts right, the fully informed count-contracted split is inconclusive against MONO rather than at parity (no equivalence margin was pre-specified), and that subset is selected by the treatment arm's own output. Splitting it again by a proxy for our registered roster rule (unadjusted post-hoc cuts, Table 46) goes against us: the contract's gain sits where the planner had already written the answer down, and part of that swing is the task. The count re-ask changes nothing (-0.7, 0.0, $+0.1$ points).

Live fresh. On Live fresh (single-call routing tasks) the loss is large (15.8, 15.0, 3.2 points) and is *planner over-selection*: executions with fan-out 1 are not separated from MONO, while the 147/166/59 of 600 with fan-out ≥ 2 score 0.020/0.120/0.000 against MONO's 0.687/0.675/0.627, mostly because the extra worker emits a call rather than the empty list it was told to return. Conditional on splitting this design does not separate the three models, so what differs there is the planner's fan-out rate (Appendix E). The pass repairs by pruning (192 removals at 27B) and recovers 44%/78%/142%; neither single-call prompt is universally better there: the empty-candidate call is 5.3 points below MONO at 12B and CHECKLIST is 4.0 below that call at 27B.

E.2 Detail behind Section 5

As above: these paragraphs stood in Section 5 and were moved here with only their lead-ins edited. All are post hoc.

Sub-classes and the netting against MONO. Of the loss events of Table 35: *wrong emitted function set*, 22%/22%/43% (3.2/2.8/2.3 points), of which 25/23/14 are a planner superset whose extra worker emitted a call and the rest a substitution, a subset, or a worker that abstained or emitted a foreign name (Appendix E). *Residual*: 34%/29%/57% (5.1/3.7/3.0 points), of which argument errors in correctly counted calls are 35/31/29 events, under-emission 12/5/0 and over-emission 3/1/1. The split is right where MONO is wrong in 1.2 to 2.1 points of executions, mostly on MONO's argument errors, so these are gross shares: netted by failure class against MONO's own failures, only the count channel on the Gemma models (+74, +77 executions) and the wrong function set at 12B and on Llama (+41, +22) exclude zero, and the argument channel does on no model. In the raw split the classification order hides argument errors behind dropped items: against the merge control that channel is +30, +25 and +18 executions (CIs 0 to 62, 0 to 52, −5 to 42, lower bounds a resample can move by one execution), the largest part of the 27B residual at point estimate (Appendix E).

After the merge. No under-emission survives the merge (0 of 72/58/82 merged-arm failures on parallel_multiple); what remains is wrong function sets (21/13/17), argument errors in correctly counted calls (48/40/62) and over-emission (3/5/3). The slot counts split by whether the planner repeated that function's name are in Appendix E.

Over-selection by planner prompt. The planner prompt induces the over-selection: asked instead for cases and counts, the manifest planner over-selects on *multiple* on 0% and 2.4% of executions against the split planner's 9.4% and 6.1%, and on Llama neither moves (1.5% both; Appendix E).

Full-context surplus. On parallel_multiple at 12B, of 265 full-context failures 182 leak another worker's function and 265 of 305 surplus slots are byte-identical copies; at 27B the leak is gone and 74 of 111 surplus slots are copies; on Llama 9 of 31 are, and 42 of 86 full-context failures are argument errors, as are 41 of MONO's 50 in the same category.

F Post-hoc train pilot: one worker per planner item

A natural alternative is the operator that exploits the planner's per-invocation items instead of merging them: one worker per planner item, repeated names kept, no merge (the one-worker-per-call limit of the contract). We ran it after the test pass on the train split only (162 tasks, k=5, unpinned providers as in the pilots), so it is exploratory. Gemma-12B: 0.814 against 0.723 (split), 0.836 (merged) and 0.885 (MONO); 56% of the gap recovered; the contrast against the half-gap is +0.9 (CI −3.3 to +5.1) and against the merge control −2.2 (CI −5.2 to +0.8). Gemma-27B: 0.849 against 0.749, 0.852 and 0.891; 70% recovered; +2.9 (CI −1.4 to +7.2) and −0.2 (CI −3.3 to +2.8). Llama-70B: 0.800 against 0.854, 0.878 and 0.941; the per-item operator is 5.4 points *below* the split (−9.8 against the half-gap, CI −13.8 to −5.7; −7.8 against the merge control, CI −11.2 to −4.3). It loses on *parallel* (0.913 → 0.840 at 12B), where the registered split's single surviving item hands the worker the full request and the per-item operator instead hands each worker one paraphrased clause. The per-item operator is therefore another way of restoring the cases without restoring the arguments, and it does not exceed the merge control on any model. Its code was added to the harness after the test pass and touches no registered arm.

G Does the aggregation assumption exist outside this harness?

The merge control isolates a rule we wrote: an aggregation step that keeps one item per function name when the planner, told to write one item per function, writes one per invocation. A reviewer asked whether deployed frameworks carry the same assumption. We audited, read-only and with the decisive lines quoted so that a reader can check them, twelve open-source agent and tool-calling frameworks and the three major providers' tool-calling APIs (`paper/framework_audit.md` in the supplementary material lists each framework, its version, the structure that carries calls and the verdict). The question was whether a request that needs the same tool invoked several times with different arguments is representable and preserved through planning, dispatch and result merging.

The answer is that no surveyed stage keys tool *calls* by function name: every planner or dispatcher carries calls as a list, or a dictionary keyed by call id, step index, task id or evidence id, and every result merge matches on that id; dictionaries keyed by tool name appear only as registries that look up the callable. Multiplicity is lost in four places, all by keeping one item of a single-valued slot rather than by name: CrewAI's native tool path executes only the first of several parallel calls when the batch contains a tool marked `result_as_answer` or `max_usage_count`, the closest analogue to the collision we measured; the OpenAI Agents SDK and AutoGen execute the first of several hand-offs requested in one turn and discard the rest, with an explicit message; LlamaIndex's agent workflow holds the next agent in a single slot, so a later hand-off overwrites an earlier one; and the LangGraph supervisor disables parallel hand-offs by default, which prevents the case rather than dropping it. Two frameworks de-duplicate by the pair of name and arguments, never by name alone.

Reading: the first-item-wins collision is a defect of our harness and of the planner contract we wrote, not a pattern of the surveyed frameworks' tool paths; what they share with it is the hand-off path, where a turn that asks for several hand-offs keeps one. The merge control is therefore reported as the control that isolates a harness-level aggregation choice, and what transfers is the mechanism it exposed: a planner told to write one item per function writes one per invocation on two of five plans, and a worker emits exactly the cases it is handed.

H PROTOCOL DETAILS

Split manifests and sample rules (sha-stratified, prefix-stable), the paired t and sign-flip procedures, Holm families, the pinned scorer digests, the lenient parsers, provider pinning, and the full registration texts (including train-pilot disclosures and the pre-specified interpretation ladders) are reproduced in the supplementary material and regenerate from the committed artifacts. The registered readout of Table 2 is produced by evaluating the three BFCL-fresh artifacts alone; the twelve-cell report released alongside it adjusts its Holm block over all four sets per model and is not the registered readout.