LEARNING WHAT TO CHECK: PROCESS SUPERVISION WITH ARTIFACT-GROUNDED VERIFICATION FOR SCIENTIFIC AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents can search the literature, write analysis code, and operate scientific software. A prerequisite for improving such agents from experience is scalable feedback. Coding provides an instructive contrast: compilers and tests score attempts automatically and localize failures. In experimental science, decisive labels often arrive only after costly wet-lab work and reveal little about which intermediate decision failed. We ask whether the structured artifacts produced during scientific work can provide an analogous source of feedback. We focus on one capability within the broader scientific process: checking computational outputs before they guide experiments. Scientific pipelines often encode the same underlying quantities in multiple files, creating consistency relations that can be recomputed without access to experimental outcomes. We formalize review as three abilities (selecting what to check, executing the check, and interpreting its result) and construct a benchmark in which the relevant checks and their programmatic semantics are controlled. The resulting diagnosis is a check-elicitation gap: the agents we evaluate can execute a specified check but often fail to identify which check is relevant. This diagnosis motivates a model–verifier architecture. The model handles open-ended check selection and computation; missing check knowledge can be supplied at run time or distilled into its weights. An independent verification layer then recomputes claims governed by exact programmatic rules, without consulting task labels. Model-side interventions raise performance on withheld defects from 12.8% to above 88%, while verification restores accuracy on defect-free inputs to 91.7% without sacrificing those gains. The missing-check pattern also appears on an external biological protocol benchmark. More broadly, our results suggest a division of labor for artifact-producing scientific workflows: train flexible selection and computation into the model, and place machine-checkable claims behind an independent verification boundary.

1 INTRODUCTION

Language-model agents can now search the literature, write analysis code, operate scientific software, and inspect experimental data (Huang et al., 2025). Recent work aims to improve such agents through repeated cycles of interaction, feedback, and training (Xue et al., 2026). Regardless of the optimization algorithm, every self-improvement loop requires an evaluator: the system must determine whether an attempt succeeded and, for a long trajectory, which intermediate decision should change.

Coding tasks provide this signal unusually well. A compiler, unit test, or execution trace can evaluate an attempt automatically and often localize the failure. Scientific work rarely comes with an equivalent oracle. In biology, decisive evidence may arrive only after a candidate has been synthesized, an assay has been run, or an experiment has been repeated. Such feedback is expensive and delayed, and a final success or failure provides little credit assignment across the file operations, calculations, tool calls, and scientific choices that preceded it. Outcome-driven training (Dong et al., 2023; Shao et al., 2024) inherits this limitation: on an earlier version of the task studied here, five rounds of it produced no reliable gain (Appendix A). This paper asks whether the structured artifacts produced by a scientific workflow can provide useful feedback before the experimental outcome arrives.

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.

Scope and setting. Scientific work includes forming hypotheses, choosing experiments, conducting analyses, inspecting their outputs, and repairing failures. We isolate the inspection stage: given the completed outputs of a computational pipeline, can an agent decide what evidence should be checked, perform those checks, and interpret the results correctly? We do not evaluate hypothesis generation, experimental design, repair of the pipeline, or wet-lab execution. Our setting is a neoantigen-selection workflow. A neoantigen is a short peptide, created by a tumor-specific mutation, that the immune system may recognize. Computational pipelines start from tumor sequencing, identify mutations, match them to the patient's immune-system (HLA) alleles, and rank candidate peptides for experimental testing. Testing those candidates is costly, so inconsistent computational outputs should be caught first. The workflow's files describe the same underlying quantities in different forms: read counts imply allele fractions; variant tables should agree with VCF records; ranked candidates should remain traceable to the mutations that produced them; and the tumor purity reported by quality control should agree with the allele fractions, since heterozygous somatic variants in a sample of purity p sit, on median, near $p/2$. Each such relation can be recomputed by a program. We call it a *check*, and the set of potentially relevant checks the *check space*.

Three abilities in scientific review. A correct review requires three distinct abilities. The agent must first propose the relevant check; it must then compute that check over the pipeline files; finally, it must decide whether the computed result indicates a defect. We call the first ability *check selection* and measure it by reported coverage; the other two are *execution* and *interpretation*. A single accuracy number fuses them. To separate them we built a task (§2) in which each instance is a set of pipeline output files (a *bundle*) with at most one planted inconsistency, governed by 17 checks whose truth a program can decide. The task statement names 7 of the checks and withholds 10, and we choose which.

From missing checks to verified claims. When the defect's check is named, the agent scores 100.0%; when it is one of the ten withheld checks, 12.8% (Figure 1a). The agent's answers show why: it reports running a median of 2 of the ten withheld checks, and 41% of its runs report none. On 152 dev instances with a withheld defect, listing all 17 checks with the weights unchanged raises the score from 17.1% to 98.8%. This reveals a *check-elicitation gap*: once a check is named, the model can execute it, but the task alone rarely elicits the relevant withheld check. A similar high-precision, low-recall profile appears on BioProBench, a benchmark we did not build (§3).

Listing every check cannot be the method. A short script with no model in it then scores 100%, so the benchmark would stop measuring anything, and a real catalog of checks would not fit in a prompt. We test two ways of supplying the missing check specifications without placing the full catalog in the initial task statement. *Tool-triggered retrieval*, a form of retrieval augmentation (Lewis et al., 2020), appends the checks relevant to the files an agent touches to that turn's tool result (a *knowledge layer*, §4). *Context distillation* (Askell et al., 2021; Snell et al., 2022) moves them into the weights: a copy of the model with access to all 17 checks generates successful trajectories, which are then used to train the model under the 7-check prompt (§5). Both increase reported coverage, with 97% of runs reporting all ten withheld checks, and raise the withheld-defect score to 88.1% and 92.2%. Both also reduce selectivity: accuracy on *clean* bundles (no planted inconsistency) falls from 98.8% to 41.7% and 67.9%. In a 20-question probe on BixBench, where the supplied knowledge consisted of analysis conventions the data cannot resolve, the knowledge layer made answers worse.

Most false alarms come from a single check, and the analysis in §6 points to why. The distilled model learned that a purity check exists, but judges it on individual outlier variants rather than on the median it is defined on, a decision rule no prompt states. This motivates an independent verification boundary between the probabilistic agent and the scientific artifacts. Following the LLM-Modulo pattern (Kambhampati et al., 2024), the agent proposes and computes candidate checks, while a deterministic *claim verifier* (§7) re-derives claims whose semantics are programmatically specified (15 of the 17 checks) from the agent's own files, without labels. One implementation serves both as a tool the agent may call and as a mandatory gate on its final answer. When it accepts a run, the answer key agrees 99.6% of the time. Used as an optional tool, it raises clean-bundle accuracy from 67.9% to 91.7% without losing withheld-defect performance (Figure 1b); used as a gate, it raises it to 100% and changes no correct answer. The two stages use different signals: the first is process supervision of the model with key-selected teacher turns, the second is key-free verification at run time, not yet used as a training reward.

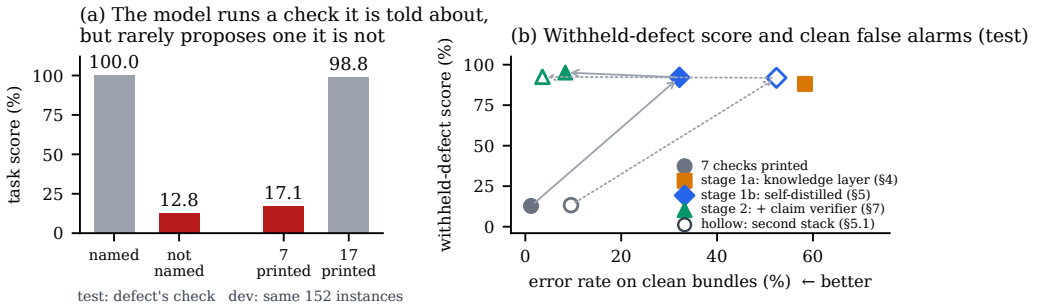
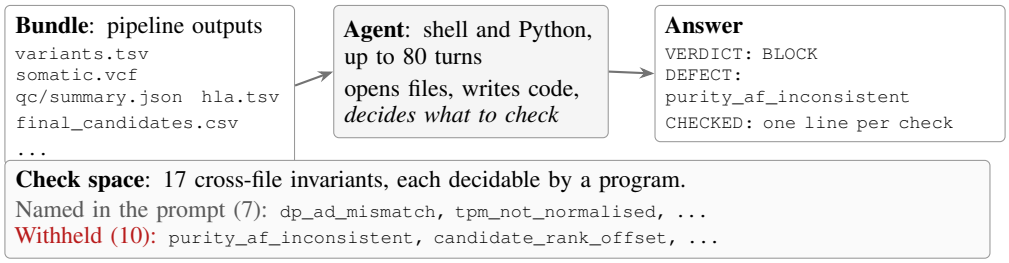


Figure 1: **(a)** Execution versus check selection. Left pair: test instances whose defect's check is named in the prompt (n=84) versus withheld (n=160). Right pair: the same 152 dev instances with 7 versus all 17 checks listed, with no weight change. **(b)** The two stages on the held-out test split: score on withheld-defect instances against the error rate on clean instances. Stage 1 moves up but right; stage 2 (verifier as a tool) moves left. Filled marks: one NVFP4 process; hollow: the second stack (§5.1).



Stage 1, raise coverage of the withheld checks: append them to tool results (§4) or distill them into the weights (§5). (Printing all 17 is only an oracle: a script then scores 100%.) **Stage 2, interpret:** verify each check the agent claims, against its own files (§7).

Figure 2: The task and the two stages. The prompt names a chosen subset of the check space; bundle, tools, turn budget and grader are held fixed.

Contributions. First, we formulate scientific review as three measurable abilities (check selection, execution, and interpretation) and introduce a benchmark in which checks can be selectively withheld while their truth remains programmatically decidable. Second, we identify a check-elicitation gap and compare two ways of narrowing it, tool-triggered retrieval and context distillation, each reported on defective, named-check and clean inputs, with no loss on an unrelated protocol benchmark. Third, we derive a model–verifier architecture from the observed failure mode: the model retains responsibility for open-ended check selection and computation, while an independent verification boundary adjudicates claims with exact programmatic semantics. The same verifier can provide interactive feedback to the agent or enforce a platform-level gate. Together, these components turn redundant scientific artifacts into intermediate feedback under sparse outcome labels.

2 A TASK WITH A CONTROLLABLE CHECK SPACE

Instances. Each instance is a *bundle*: the output files of a simulated neoantigen-calling pipeline built from public somatic variant calls of four SEQC2 reference samples (Fang et al., 2021) and one publicly released tumor–normal exome pair (BioProject PRJNA298330) (Figure 2). The files are too large to read by eye. The agent works in a sandbox with shell and file tools and must answer with a verdict: `CONTINUE` (usable), `NORMALIZE` (a scaling or coverage issue must be reconciled first) or `BLOCK` (the files contradict each other). It must also name the defect and list, one line each, the checks it ran. At most one defect is planted per bundle.

The check space. Seventeen cross-file invariants span nine kinds of reasoning: lookups, arithmetic identities (`af` = `ad_alt`/(`ad_ref`+`ad_alt`)), subset and join relations, orderings (ranks are exactly

$1..N$), intervals, a sum (TPM sums to 10^6), uniqueness, and a distributional check (allele fractions agree with tumor purity); Appendix B lists them. Two properties matter. First, *every invariant is decidable by a program*: a reference implementation computes the true state of all 17 on any bundle, so we can grade what the agent checked, not only its verdict. Second, *the prompt names a controllable subset*: 7 in the deployment condition, all 17 in the oracle condition. Nothing else changes.

Splits and strata. We iterate on a 200-instance dev split and report on a 600-instance test split that was never used for iteration. A separate 600-instance training split is used only to collect teacher trajectories. Bundle contents are disjoint across splits (SHA-256 over every file). The test split has three strata, evaluated separately: the defect's check is withheld (474 instances, 79%), the defect's check is named (84, 14%), or the bundle is clean (42, 7%). We run 160 withheld-defect instances and all instances of the other two strata. We never report a single "test accuracy": the methods below trade performance between strata, and a weighted average hides exactly that trade.

Metrics. We measure the steps of a review separately. *Reported coverage* is the number of the ten withheld checks that a rollout's answer claims to have run. It is self-reported: it counts claims, not executions traced in the tool calls. An item listed as skipped, or listed with a finding that says it was not run ("not in the 7-item checklist"), does not count. Whether a reported finding is *true of the artifact* is a separate question; §7 checks it by re-deriving each claim, which shows the claimed number is right but not that the agent computed it before writing it. The downstream *task score* awards one point for the verdict and one for the defect name, and averages score/2 over $k{=}2$ rollouts per instance. On clean bundles the correct answer is `CONTINUE`, so $1 -$ clean score is the *clean error rate*, our false-alarm measure. Differences are paired by instance, with 95% bootstrap intervals.

Models and the evaluation contract. The main model is Qwen3.8-27B (Qwen Team, 2026) served with vLLM (Kwon et al., 2023). MiniMax-M3, available only through an API whose version cannot be pinned (Appendix D), is a second model for replication. Any two numbers we compare share split, k and one serving process. This matters: moving the same checkpoint from NVFP4 to BF16 serving shifts next-token log-probabilities 0.53 times as far as LoRA training does (median), and one trained adapter's clean accuracy is 67.9% under one configuration and 42.9% under the other (§5.1). All answers are read by one extraction module with a regression suite, and every trained adapter must pass a log-probability check before evaluation; both rules came from silent scoring defects in our own harness (Appendix G).

3 Diagnosis: the agent executes checks but does not propose them

Execution is not the bottleneck. On test instances whose defect's check is named in the prompt, Qwen3.8-27B scores 100.0% ($n{=}84$) and MiniMax-M3 scores 97.7% ($n{=}57$). Both compute an identity over hundreds of rows and compare it to a column without difficulty.

Coverage is. When the defect's check is withheld, with the same model, harness, tools and grader, the score is 12.8% ($n{=}160$). The answers say why. On these rollouts the agent claims to have run a mean of 3.7 of the ten withheld checks (median 2). 130 of 320 rollouts (41%) claim none, and only 10.3% claim all ten. Most withheld checks are never proposed.

A weight-frozen control. The numbers above concern different defects, so we also hold the instances fixed and change only the prompt. On the 152 dev instances with a withheld defect, printing all 17 checks instead of 7 raises the score from 17.1% to 98.8% ($k{=}2$ in both arms). No parameter changed, so the 82-point gap is attributable to text such as "also check whether ranks are exactly 1..N". Once named, the checks are executed.

Why printing the checks cannot be the method. Once all 17 checks are printed, a 38-line program that reads the checklist and applies each invariant scores 100%. A benchmark that prints its rubric therefore measures instruction following, and our benchmark must keep withholding the check space even as the methods below supply it. The supply channel cannot be the task statement.

The same deficit on an external benchmark. BioProBench (Liu et al., 2025) asks, in its ERR task, whether a protocol step contains an error. With the official prompts, Qwen3.8-27B and MiniMax-M3

both reach 62.3% accuracy, and both are short in the same place: precision 68.9% and 75.0%, recall only 46.7% and 40.4%. This high-precision, low-recall profile is consistent with missed error classes, although a single-turn benchmark does not expose which checks the model actually considered. Splicing 17 error classes, derived from the *training* split only, into the official prompt raised Qwen's recall from 46.7% to 57.2% (+10.5) and accuracy from 62.3% to 67.0%, with precision not falling (68.9% $\rightarrow$ 71.9%) and zero gradient steps; MiniMax moved the same way, less (Appendix E).

4 STAGE 1A: TOOL-TRIGGERED RETRIEVAL OF CHECKS

Mechanism. A *catalog* is a YAML file of entries. Each entry names the files it applies to, one sentence stating an invariant, and a consequence line such as `No -> BLOCK, defect purity_af_inconsistent`. After each turn, the harness appends the entries that match the files the turn touched to that turn's tool result: at most 5 per turn and 12 per episode, each served once. The prompt, tools, turn budget and grader are untouched, so the benchmark still withholds the check space while the agent receives it. A whole rollout receives 1,351 characters of catalog text.

What made it work. What mattered was not delivery but what each entry said. Without the consequence line, most rollouts performed the computation but did not use the result in their answer. With it, an imperative header and a per-turn budget, dev reached 89.1% (the design history is in Appendix C).

Held-out result. On the 160 test instances with a withheld defect, coverage rises from a mean of 3.7 to 9.9 withheld checks per rollout (97.2% of rollouts claim all ten). The score rises from 12.8% to 88.1%, a paired difference of +75.3 (95% CI [+70.0, +80.2]) and 1.0 point below dev. All ten withheld defect kinds improve, each with an interval excluding zero (Table 6).

The cost: false alarms. Every instance above contains a defect. On the 42 clean test bundles, the ordering inverts: clean accuracy falls from 98.8% with 7 printed checks to 41.7% with the knowledge layer, and to 39.3% with all 17 printed. MiniMax-M3 shows the same direction (90.6% $\rightarrow$ 65.4% with all 17 printed). We had predicted the knowledge layer would be gentler than printing everything, since it surfaces a few checks at a time; it was not. Given checks it had not considered, the model finds one that seems to fit. A method evaluated only on defective inputs is measuring how eager it is.

The boundary: knowledge the data cannot settle. BixBench (Mitchener et al., 2025) asks open analysis questions. We wrote a catalog of 15 analysis conventions for it (which normalization, which threshold, which test) and ran the same arms on 20 questions with three rollouts each. Every rollout received entries, yet accuracy fell from 28.3% to 25.0% and best-of-3 from 35.0% to 30.0%, while questions with three identical answers rose from 4 to 7 (exact matching; e.g. 165/107/165 became 107/107/107). Supplied knowledge appears to concentrate answers: toward the truth when the artifact can settle an entry, toward a bias when nothing in the data resolves it. With 20 questions this is a probe, but it suggests a boundary condition: the check space should be *verifiable against the artifact*, not merely enumerable.

5 STAGE 1B: CONTEXT DISTILLATION

Design. This is context distillation (Askell et al., 2021; Snell et al., 2022), and it can be viewed as learning with privileged information (Vapnik & Vashist, 2009): the check space is available at training time and withheld at test time. The teacher is *the same checkpoint* as the student, run with all 17 checks printed; the student sees the 7-check prompt. What can transfer is therefore knowledge of which checks exist and how to run them. A stronger teacher model would confound this with ordinary capability transfer. We keep teacher trajectories on the training split whose verdict and defect are both correct.

The target is each turn, not the final answer. What we want to transfer lives in *which files the agent opens and what code it writes* in the middle turns. A final answer contains specific computed values ("median af 0.048 over 400 rows"), and a model trained to emit it without doing the computation learns to invent numbers. So each training example is one turn. The input is the

Table 1: Self-distillation on the test strata (task score, k=2). Each column group shares one serving process. Δ is the paired per-instance difference to the base model, with a 95% bootstrap CI.

		NVFP4 stack (§5)			second stack, BF16 (§5.1)		
stratum	n	base	distilled	Δ [95% CI]	base	distilled	Δ [95% CI]
withheld defect	160	12.8	**92.2**	$+79.4$ [$+74.4$, $+84.1$]	13.3	**91.9**	$+78.6$ [$+73.6$, $+83.3$]
named check	84	100.0	99.7	-0.3 [-0.9, $+0.0$]	100.0	98.2	-1.8 [-3.9, -0.3]
clean	42	98.8	**67.9**	-31.0 [-40.5, -21.4]	90.5	**47.6**	-42.9 [-54.8, -31.0]

conversation up to that turn, with the teacher's prompt replaced by the student's; the target is the teacher's next turn. We checked that a rewritten training prompt and a real evaluation prompt agree byte for byte after chat templating. Turn-level targets make context length critical. The final turn carries the verdict and has a median length of 27,928 tokens, so at a 16k limit only 7% of final turns survive truncation, and at 32k, 65%.

Clean trajectories. The teacher is 99.4% correct on defective bundles but only 34.2% on clean ones, so rejection sampling alone yields too few clean examples. We first tried STaR-style rationalization (Zelikman et al., 2022): we handed the model the correct verdict and kept trajectories whose every claim verified. All 109 collected trajectories appealed to the handed-over answer somewhere; one noticed a contradiction and wrote "But the ground truth is that everything passes". Claim-level checks cannot catch this, because the conclusion is right and the damage is in the path. We discarded them and used 88 unaided correct rollouts on 69 clean instances. The corpus has 4,087 turn-level examples from 573 trajectories, 12% of them clean. Two withheld defect kinds, `candidate_rank_offset` and `af_ad_mismatch`, contribute *no* training instance. Their names still appear in the teacher's final answers, so the hold-out tests transfer to a kind without a worked example, not the invention of an unmentioned check.

Training. LoRA (Hu et al., 2022) with r=16 on seven projection types, the base loaded in NF4 (Dettmers et al., 2023), a 32k context, 15% of trajectories held out for validation, and two epochs (846 steps on one RTX PRO 6000, 96 GB). The BF16 adapter is served on an NVFP4 checkpoint of the same model, in the same vLLM process as the base model (Table 7).

Results. Table 1 (left). Coverage moves as intended: the student claims a mean of 9.8 of the ten withheld checks, and 97.2% of its rollouts claim all ten, against 10.3% for the base model. The withheld-defect score reaches 92.2%, above the knowledge layer's 88.1%, and clean accuracy, 67.9%, is also above the knowledge layer's 41.7%. So moving the check space from the scaffold into the weights improved both measures. The comparison is descriptive, not causal: the teacher was the 17-check oracle, not the knowledge layer. The student barely loses on checks the prompt already names (-0.3). Clean inputs are the weak point: 67.9% is far below the base model's 98.8%, and 26 of the 27 false alarms claim the same invariant, `purity_af_inconsistent`. The model often knows the right answer, since best-of-2 clean accuracy is 88.1%.

5.1 Replication in a second training-and-serving stack

The run above made choices to fit a single 96 GB GPU: a quantized base, a 32k context, and NVFP4 serving. We retrained in a different stack: a BF16 base, a 49,152-token context (which keeps 93% of final turns), data-parallel training on four H200s (910 steps, 5.8 hours), and BF16 serving on a different vLLM version and hardware. The same 573 trajectories yield more rows at 49k (4,404 against 4,087), and 15% of trajectories are again held out for validation (Table 7). Base model, new adapter and adapter plus verifier (§7) were served by one process. Several things change at once, so this stack tests whether the findings survive a change of stack. It cannot attribute any difference to precision.

Table 1 (right). *The withheld result replicates*: 13.3% $\rightarrow$ 91.9%, and so does coverage. On withheld instances, the share of rollouts claiming all ten withheld checks rises from 9.4% to 97.8%. The eight trained kinds move by $+81.9$ points ($[+76.5, +86.6]$) and the two held-out kinds by $+68.9$ ($[+57.3, +79.9]$), less than the trained kinds, consistent with transfer that is real but incomplete without a worked example. *The cost on named checks is small but no longer zero*: -1.8, with an

interval that excludes zero. *Clean accuracy is again the weak point* (47.6%, against 67.9% in the first stack), and the evidence points to serving rather than to a detectable training-recipe effect. We served the first stack's adapter with the second stack's serving configuration. Its clean accuracy falls from 67.9% to 42.9%, statistically indistinguishable from the new adapter's (-4.8, $[-20.2, +10.7]$). On the eight trained withheld kinds the two adapters are also indistinguishable (-2.1, $[-5.9, +1.5]$). We detected no difference attributable to the training recipe at this sample size, whereas switching the serving configuration was associated with a 25-point change in clean accuracy. Both adapters also leave the linear-attention layers of this hybrid architecture unadapted (Appendix D). Distillation did not reduce performance outside the task: on BioProBench ERR, served from the same process (k=2), the adapter scores 61.5% against the base model's 59.3% (paired $+2.2$, $[-0.7, +5.2]$) and more often finishes within the token budget (7 and 10 unparsed answers in two runs, against 17 and 15; Appendix E). The training data share nothing with this benchmark, so this is evidence that distillation did not cost general capability.

6 WHY COVERAGE ALONE CREATES FALSE ALARMS

Stage 1 raised coverage close to its ceiling, yet clean inputs got worse. In the second stack, 98.8% of the student's rollouts on *clean* bundles claim all ten withheld checks, yet 52.4% of those bundles are flagged. The checks are being proposed; they are being misjudged. The false alarms themselves show how. 40 of the student's 44 clean false alarms again claim `purity_af_inconsistent`. That invariant compares $|\mathrm{median}(af) - p/2|$ with a tolerance of 0.10. We recomputed this gap on each of the 40 bundles. The largest is 0.0024 and the median 0.0014, about forty times below the tolerance. These are not borderline cases, so a slightly miscalibrated threshold cannot explain them.

Instead, 31 of the 40 rollouts point at *individual* variants whose allele fraction is far above $p/2$, and call that the violation. Across the variants of one tumor, such scatter is ordinary biology (subclonal mutations, copy-number changes). The invariant is defined on the median precisely so that scatter does not trigger it. The base model makes the same move less often (4 of its 6 purity false alarms).

Our hypothesis is an underspecified check definition. The student sees only the check's name. The teacher's one-line definition reads "a heterozygous somatic variant at purity p sits near af $= p/2$". It is phrased per variant, and no prompt mentions the median or the tolerance. Self-distillation transferred the *proposal* of the check, and the model appears to fill the missing definition with the most salient evidence in the file, the outliers. A check that is only named transfers only its name, and more coverage does not repair the resulting interpretation failure. Because this diagnosis was made post hoc on the test split, we did not tune on it and instead tested it on new data.

A prospective, interventional test. We then tested the hypothesis on a clean split sealed after this analysis (228 clean bundles, new seed; Appendix B), all arms in one BF16 process. In this interim paired analysis (the distilled model's run covered 156 of the 228 bundles at submission), it again false-alarms on 55.1%, 79 of 86 times on purity, with gaps of at most 0.0024. Adding one paragraph to its prompt that states the statistic and the tolerance lowers the rate to 0.9%, with no purity false alarm (paired -54.5, $[-62.2, -46.2]$), the base model's level. Training on teacher trajectories that state the definition lowers it to 24.1% (paired -27.6, $[-37.8, -17.3]$). These results are consistent with an underspecified definition being a major contributor to the false alarms, and on the completed pairs supplying the definition at the moment of judgment performed better than training it in. Because the added paragraph names a withheld check, it serves here as a diagnostic, not as a method.

7 STAGE 2: A DETERMINISTIC CLAIM VERIFIER

Following the LLM-Modulo pattern (Kambhampati et al., 2024), we pair the model with a deterministic external verifier. It re-derives the claims covered by its registry, of the form "I checked X and found Y", from the files the agent itself had, without the label, the reference implementation or the generator; one implementation serves as the agent's tool and as the platform's gate, so the two cannot disagree.

Only claimed checks are verified. The verifier exposes `verify(claim, artifact)` and deliberately not `verify_all(artifact)`. Running all 17 checks would *be* the reference solver, which

Table 2: The claim verifier as a tool (budget 4), paired by instance against the same model without it, in the same serving process. Task score, k=2; 95% bootstrap CIs. The NVFP4 stack's tool arm was not run on the named-check stratum (n.r.).

	self-distilled, NVFP4 stack			self-distilled, second stack		
stratum	no tool	tool	Δ [95% CI]	no tool	tool	Δ [95% CI]
clean (n=42)	67.9	**91.7**	+23.8 [+11.9, +36.9]	47.6	**96.4**	+48.8 [+35.7, +61.9]
withheld (n=160)	92.2	95.2	+3.0 [+0.0, +6.1]	91.9	92.5	+0.6 [−2.0, +3.4]
named (n=84)	n.r.	n.r.	n.r.	98.2	98.8	+0.6 [−1.8, +3.0]

scores 100% and measures nothing. The verifier can therefore say "the violation you claim is not supported by your own numbers", but never "you forgot to check the ranks". It acts on interpretation, not on coverage. It covers 15 of the 17 invariants and abstains on the other two rather than guess.

Agreement with the answer key. On the 572 test rollouts of the self-distilled model, we treat "every covered claim re-derives" as a prediction of correctness. When the verifier accepts a rollout, the key agrees on 99.6% of them (502 of 504). It rejects 96.5% of the rollouts the key marks wrong, and abstains on 1.9%. All 66 failures it found were interpretation errors, with the numbers right and the conclusion wrong; none was a fabricated number. Run in reverse over the labels, the verifiers find that 799 of 800 dev and test bundles violate exactly the invariant their label names, and no clean bundle violates any.

As a tool the agent may call. The agent's workspace contains only a stub that knows the request protocol and no check names; the verifier runs in a host process, so the list of valid check names cannot be read from the workspace; agents did try (Appendix F). A per-rollout budget of 4 distinct checks, fixed on dev before any test run and enforced on the host side, closes the remaining channel of querying every name.

Results. Table 2. Clean accuracy rises by 23.8 points in the NVFP4 stack and by 48.8 in the second stack, and in the second stack all 40 purity false alarms disappear. The verifier registry was frozen (identical file hash) before the second-stack runs and before the analysis in §6; its purity check simply implements the invariant's definition. Asked about purity, it recomputes the median statistic and replies with the gap and the tolerance, exactly the definition the prompt left out. The first stack's purity false alarms were known when the registry was written, so only the second stack's result is prospective. The NVFP4 figure, 91.7%, also exceeds that model's best-of-2 clean accuracy (88.1%), so the tool changes answers rather than merely helping pick the better of two samples. Against the untrained base in the second stack's process, the self-distilled model with the tool is $+84.7$ points on trained withheld kinds and $+6.0$ on clean ($[-2.4, +14.3]$). Used this way, high coverage no longer costs selectivity. Used instead as a mandatory gate after the agent answers, the verifier raises NVFP4 clean accuracy to 100% and changes no correct answer on the defect strata.

Does the tool leak the withheld checks? Before running, we wrote down that a rise in withheld-defect recall with the tool would indicate leakage. Defect identification did rise, from 91.2% to 95.0%, so the criterion fired. Leakage does not appear to explain it: (i) without the tool the distilled model already claims all ten withheld checks in 97.2% of rollouts (97.8% in the second stack); (ii) in all 83 cases where the verifier rejected a claim the model abandoned it, and in 74 then named the correct defect; and (iii) removing the budget let 6.9% of rollouts query every check but changed identification by only $+0.6$ ($[-1.9, +3.1]$). On the two held-out kinds the second stack scores slightly lower with the tool (-5.5, $[-11.6, 0.0]$).

8 Related work

Outcome and process supervision. Outcome-driven post-training, such as rejection sampling (Dong et al., 2023), GRPO (Shao et al., 2024) and sequence-level variants (Zheng et al., 2025), assumes the terminal signal is informative enough to credit the trajectory. Process supervision rewards intermediate steps instead (Uesato et al., 2022; Lightman et al., 2024), and Math-Shepherd

(Wang et al., 2024) obtains step labels without humans. Our contribution is upstream of the algorithm. On this task class the process units worth supervising are *which checks get proposed* and *how they are interpreted*, and both can be measured before any training. Our verifier yields per-claim signals from the artifact rather than from a learned reward model.

Distilling privileged context. Context distillation moves the effect of a prompt into the weights (Askell et al., 2021; Snell et al., 2022), building on knowledge distillation (Hinton et al., 2015); learning using privileged information (Vapnik & Vashist, 2009) formalizes a teacher that sees information the student will not have at test time. STaR (Zelikman et al., 2022) trains on the model's own successful rationales. We use the same mechanism, with one difference: the privileged context is a subset of an enumerable, verifiable check space, so what is distilled can be measured directly. Our results add two cautions. Handing the teacher the verdict contaminates the path. And distilling a check's name does not distill its definition.

Tools, retrieval and self-correction. The knowledge layer resembles retrieval augmentation (Lewis et al., 2020) delivered through the tool loop of ReAct-style agents (Yao et al., 2023; Schick et al., 2023). It differs in the unit delivered, an invariant *and its consequence*, where the consequence was worth more than the fact. It also comes with a failure condition: unverifiable knowledge hurts. Intrinsic self-correction is unreliable (Huang et al., 2024), and self-refinement relies on model feedback (Madaan et al., 2023). Our verifier follows the LLM-Modulo pattern (Kambhampati et al., 2024) for scientific review; it verifies only the checks the model claimed, which limits the most direct channel for leaking a withheld check space.

Benchmarks for biology agents. BixBench (Mitchener et al., 2025), LAB-Bench (Laurent et al., 2024), BioLP-bench (Ivanov, 2024) and BioProBench (Liu et al., 2025) evaluate models and agents on biological analysis and protocols. Agents such as Biomni (Huang et al., 2025) target broad biomedical tasks, and ScienceBuddy (Xue et al., 2026) couples harness evolution with model reinforcement learning in a recursive improvement loop. We do not claim such a loop; we study one of its prerequisites, automatic feedback attached to intermediate decisions. None of these benchmarks exposes a controllable check space, so the named-versus-withheld contrast cannot be built on them. We use BioProBench as an external anchor because it has a training split and programmatic grading. On the fragility of agent evaluation, see Kapoor et al. (2025).

9 Discussion and limitations

What to take away. Measure coverage, execution and interpretation separately before post-training; they call for different treatment. Check selection and execution can be trained into the model, but a check transferred by name transfers only the name. Claims governed by deterministic, explicitly specified rules are suitable for external verification; scientific interpretation as a whole is not. The architecture therefore targets artifact-producing workflows (tables, variant calls, rankings, QC files, executable tests) and does not cover the novelty of a hypothesis, the plausibility of a mechanism, or the choice of a threshold without a canonical answer, which is the boundary the BixBench probe illustrates. Report the false-alarm rate beside every coverage gain: here, every coverage method made clean inputs worse until claims were verified.

Limitations. *Scientific scope.* We study review of completed computational artifacts, not hypothesis formation, experiment selection, pipeline repair, or wet-lab execution. Verified diagnoses may support those stages but do not establish competence in them. *One primary benchmark, which we built.* External evidence is a single-turn replication of the phenomenon (BioProBench) and a negative boundary case (BixBench). *Tolerances.* Our one tolerance is where both trained models fail (§6); a realistic catalog has many. *Serving dependence*: one adapter's clean accuracy differs by 25 points between two serving stacks. *The verifier is not yet a training signal.*

Conclusion. A general model executes named checks almost perfectly but rarely proposes withheld ones. Retrieval and distillation narrow this gap but reduce selectivity; a label-free verifier of the agent's claims restores it for checks with exact semantics. What to train into a model and what to verify externally can thus be decided by measurement.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and experiments. An LLM coding assistant, working under the authors' direction and review, wrote much of the benchmark harness, graders, verifiers, training and serving scripts, and analysis scripts. It also launched and monitored training and evaluation jobs, and diagnosed failures in them. Several of the scoring defects in Appendix G were found this way. We also used it to compute statistics from measured records, to draft and revise text, and to convert the manuscript to LaTeX. We did not use generative AI tools to choose the research question or to decide which claims to make. Reported numbers are produced by scripts from run records, and a consistency check compares each registered number in the text with its source; the few values transcribed from original records are listed by that check together with their source. The models under study (Qwen3.8-27B, MiniMax-M3) are the subject of the paper, not assistance with it. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The benchmark is built from public, de-identified somatic variant calls: four SEQC2 reference cell-line samples and one publicly released tumor–normal exome pair (BioProject PRJNA298330). Only derived per-variant features are used and redistributed; no reads and no identifying metadata. No new human data were collected. Our results show that an automated reviewer can raise false alarms at rates that depend on its serving configuration. A reviewer deployed on clinical pipelines should be evaluated on clean inputs, under its production configuration.

REPRODUCIBILITY STATEMENT

The anonymized supplementary material contains the code, the evaluated splits and the variant pool they were generated from, the graded record of every reported rollout (with final answers for the first-stack runs), and the results file from which the paper's numbers are taken. From these, `python -m checkspace reproduce` recomputes the record-based results (813 values, all identical to the shipped ones) and `check-numbers` checks each registered number in the text against them (335 values), including that paired numbers share dataset, split and sampling budget; a few values transcribed from original records are listed with their source. `seal` re-verifies split disjointness and labels, and regenerating the sealed split from its recorded seed reproduces it byte for byte. Groups that need full agent sessions (the verifier audit, gate and budget) are reported but not recomputable from the shipped records. Serving commands, training configurations and hyperparameters are in the supplementary README, §5, §5.1 and Appendix D. MiniMax-M3 is served only through an API, so its results cannot be reproduced exactly.

REFERENCES

Amanda Askell et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023.

Li Tai Fang, Bin Zhu, Yongmei Zhao, Wanqiu Chen, Zhaowei Yang, Liz Kerrigan, Kurt Langenbach, Maryellen de Mars, Charles Lu, Kenneth Idler, et al. Establishing community reference samples, data and call sets for benchmarking cancer mutation detection using whole-genome sequencing. *Nature Biotechnology*, 39(9):1151–1160, 2021. doi: 10.1038/s41587-021-00993-6.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations*, 2024.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Gavin Li, Junze Zhang, et al. Biomni: A general-purpose biomedical AI agent. *bioRxiv*, 2025. doi: 10.1101/2025.05.30.656746.

Igor Ivanov. BioLP-bench: Measuring understanding of biological lab protocols by large language models. *bioRxiv*, 2024. doi: 10.1101/2024.08.21.608694.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22895–22907. PMLR, 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Zy4uFzMviZ.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques. LAB-Bench: Measuring capabilities of language models for biology research. *arXiv preprint arXiv:2407.10362*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Yuyang Liu, Liuzhenghao Lv, Xiancheng Zhang, Jingya Wang, Li Yuan, and Yonghong Tian. BioProBench: A corpus and benchmark for biological protocol reasoning in autonomous science. *arXiv preprint arXiv:2505.07889v4*, 2025.

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

Ludovico Mitchener, Jon M. Laurent, Alex Andonian, Benjamin Tenmann, Siddharth Narayanan, Geemi P. Wellawatte, Andrew White, Lorenzo Sani, and Samuel G. Rodriques. BixBench: a comprehensive benchmark for LLM-based agents in computational biology. *arXiv preprint arXiv:2503.00096*, 2025.

Qwen Team. Qwen3.8 model card. https://huggingface.co/Qwen/Qwen3.8-27B, 2026.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009. doi: 10.1016/j.neunet.2009.06.042.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024. doi: 10.18653/v1/2024.acl-long.510.

Shuhan Xue, Jianyuan Zhong, Ziyuan Nan, Wenbin Li, Zhaochen Yu, Jinchao Ding, Qiang Gao, Pengyu Zhan, Yuntong Zhang, Tian Cheng, Zhenfei Yin, Yingcheng Wu, and Ling Yang. Science-Buddy: Recursive-in-recursive self-improvement for interactive scientific agents. *arXiv preprint arXiv:2609.17523*, 2026.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A OUTCOME-DRIVEN POST-TRAINING ON THIS TASK

Before the work reported here, we tried outcome-driven post-training on an earlier version of the task, which used nine defect classes and shorter, rendered inputs. The reward was the final verdict. Two rounds of RAFT (Dong et al., 2023) left paired accuracy unchanged (20 instances fixed, 20 broken, p=1.0). A third round with seven times more data gained 0.4 points. GSPO (Zheng et al., 2025) gained 8 points under greedy decoding, which was not significant (p=0.28). Scores on three other tasks did not change. These runs predate the current dataset and are cited only to motivate the search for a denser signal.

B THE DATASET

This appendix explains what an instance is for readers without a genomics background, where the underlying data come from, and how instances are built and labeled.

B.1 BACKGROUND: WHAT THE PIPELINE DOES

Cancer cells carry mutations that normal cells of the same person do not. A *somatic variant* is such a mutation: a position in the genome (`chrom`, `pos`) where the tumor's DNA reads show a base (`alt`) different from the reference base (`ref`). Sequencing produces many short reads covering each position. At a variant, `ad_ref` reads support the reference base and `ad_alt` reads support the alternative; their sum is the *depth* (`dp`), and `ad_alt`/`dp` is the *allele fraction* (`af`). A tumor sample is never pure: a fraction p of its cells are tumor cells (the *purity*). A heterozygous somatic variant is present on one of two chromosome copies in tumor cells only, so its expected allele fraction is about $p/2$. Individual variants scatter around this value for biological reasons (subclonal mutations, copy-number changes), which is why the corresponding check compares the median.

A neoantigen pipeline turns such variants into candidates for a personalized cancer vaccine. It (1) calls somatic variants by comparing tumor and normal sequencing data, usually with several *variant callers* whose votes are combined; (2) determines the patient's HLA alleles, the immune-system genes that decide which peptides can be presented to T cells; (3) translates each variant into short mutant peptides (8–11 amino acids); (4) predicts how strongly each peptide binds each HLA allele; and (5) ranks the candidates for synthesis and testing. Each stage writes files, and later stages copy or derive values from earlier ones. Table 3 lists the files in a bundle. The same quantity therefore appears in several places, which is what makes cross-file checks possible.

file	content	origin
`variants.tsv`	400 somatic variants: position, alleles, read counts, `af`, `dp`, one column per caller, number of supporting callers	real (recomputed)
`somatic.vcf`	the first 200 variants in VCF format with tumor and normal genotypes	derived from the table
`reference.fai`	chromosome lengths of the GRCh38 reference genome	real
`manifest.json`	sample names, reference build, minimum validated depth	fixed
`hla.tsv`	the four HLA alleles the sample was typed for	synthetic
`predictor.json`	the binding predictor's supported alleles and peptide lengths	fixed
`peptides.csv`	12 mutant peptides, each tied to a real somatic variant	synthetic
`binding.csv`	a binding score for each (peptide, HLA) pair	synthetic
`final_candidates.csv`	the 6 top-ranked candidates with position and rank	derived
`expression.tsv`	expression of 300 genes in TPM units (sums to 10^6 by definition)	synthetic
`qc/summary.json`	mean sequencing coverage and tumor purity	derived / synthetic

Table 3: Files in a bundle. *Real*: taken from variant calls on public sequencing data. *Derived*: computed from other files so that a clean bundle is consistent. *Synthetic*: generated, since the task tests consistency checking and not biological prediction.

B.2 WHERE THE DATA COME FROM

The variant rows come from a pool of 37,919 somatic single-nucleotide variants in five tumor samples. Four are sequencing replicates of the SEQC2 reference tumor–normal cell-line pair HCC1395/HCC1395BL (Fang et al., 2021), sequenced at four sites (labeled FD, IL, LL and NC; 2,507–4,121 variants each). The fifth is a publicly released tumor–normal exome pair from BioProject PRJNA298330 (sample 3948; 23,385 variants). For each sample we ran a somatic calling pipeline with several callers (Mutect2, Strelka2, VarScan2 and DeepSomatic, the first and last also in GPU implementations) and kept, per variant, the read counts and each caller's call. For the SEQC2 samples a flag marks membership in the SEQC2 high-confidence truth set; the generator uses it only to prefer high-confidence variants when attaching peptides. No sequencing reads or personal metadata are stored.

B.3 HOW A BUNDLE IS GENERATED

Generation is seeded and deterministic. For one instance the generator:

1. picks one of the five samples and draws 400 of its variants;
2. recomputes the derived columns from the read counts (`dp` $=$ `ad_ref` $+$ `ad_alt`, `af` $=$ `ad_alt`/`dp`, the number of supporting callers), so that a clean bundle satisfies every invariant even where the raw caller output did not;
3. draws four HLA alleles from the predictor's supported list, attaches a random 8–11-residue peptide to each of 12 somatic variants, draws a binding score per peptide, and ranks the six best-scoring peptides as final candidates;
4. generates 300 expression values and rescales them to sum to 10^6;
5. sets QC coverage at random between 60 and 140 and sets purity to twice the median allele fraction, so purity agrees with the table;
6. writes the files of Table 3; and
7. for a defective instance, applies exactly one of the 17 injections of Table 4, for example changing one `af` value so that it no longer equals `ad_alt`/`dp`, or setting purity far from twice the median allele fraction.

Each injection is written next to the check that detects it and the one-line description a prompt would print, in one registry, so that the three cannot drift apart.

Labels and validity. The label is the verdict the defect implies (`BLOCK` or `NORMALIZE`; `CONTINUE` for a clean bundle) and the defect's name. A reference implementation evaluates all 17 invariants on every generated bundle, and an instance is kept only if exactly the injected invariant is violated (none for a clean bundle). Running all verifiers over the dev and test splits afterwards finds one test instance whose injection also broke a second invariant; it is counted rather than repaired, so that every reported number refers to the same data.

Splits. Instance identifiers hash a split-specific tag, and bundle contents are compared by SHA-256 across all splits: no identifier and no bundle is shared. The test split mixes three strata that the paper reports separately: the defect's invariant is *withheld* from the prompt, the defect's invariant is *named* in the prompt, or the bundle is *clean*. Sampling weights, set per split and recorded in its metadata, oversample withheld defects and clean bundles relative to a uniform draw. The sealed split of §6 was generated after the analysis there, with a new seed; regenerating it from its recorded seed reproduces it byte for byte.

C KNOWLEDGE LAYER DETAILS

A catalog entry names the files it applies to, states an invariant in one sentence, and gives its consequence, for example:

```
match:  [qc/summary.json, variants.tsv]
text:  qc tumor_purity should agree with the allele
frequencies:  a heterozygous somatic variant at purity p
sits near af = p/2.
action:  No -> BLOCK, defect purity_af_inconsistent.
```

invariant	kind	prompt	question the check answers
`pos_beyond_contig`	lookup	named	Is every variant position inside its chromosome, per `reference.fai`?
`dp_ad_mismatch`	arithmetic	named	Does `dp` equal `ad_ref + ad_alt` on every row?
`tumor_normal_swap`	lookup	named	Does the sample the manifest calls the tumor carry the alt reads?
`hla_out_of_domain`	subset	named	Is every HLA allele used listed in the predictor's supported alleles?
`peptide_len_out_of_range`	interval	named	Is every peptide within the predictor's declared length range?
`tpm_not_normalised`	sum	named	The `tpm` column is declared TPM. Does it sum to 10^6? (`NORMALIZE`)
`depth_below_validated`	lookup	named	Is QC mean coverage at or above the validated minimum? (`NORMALIZE`)
`af_ad_mismatch`	arithmetic	withheld†	Does `af` equal `ad_alt/(ad_ref+ad_alt)` on every row?
`caller_count_mismatch`	arithmetic	withheld	Does `n_callers_supporting` equal the number of caller columns set?
`untraceable_candidate`	join	withheld	Does every final candidate peptide appear in an upstream table?
`candidate_hla_unlisted`	subset	withheld	Is every candidate's HLA one the sample was typed for?
`candidate_pos_orphan`	join	withheld	Does every candidate `(chrom, pos)` appear in `variants.tsv`?
`duplicate_candidate`	uniqueness	withheld	Does any (peptide, HLA) pair take two ranked slots?
`vcf_record_orphan`	subset	withheld	Does every VCF record have a row in `variants.tsv`?
`candidate_rank_disorder`	ordering	withheld	Does rank order agree with binding scores?
`purity_af_inconsistent`	distribution	withheld	Is $\lvert\mathrm{median}(\texttt{af}) - p/2\rvert \leq 0.10$ for QC purity p?
`candidate_rank_offset`	ordering	withheld†	Are ranks exactly $1..N$ with no gaps or repeats?

Table 4: The check space. Every defect implies `BLOCK` unless marked. *Named*: printed in the deployment prompt. *Withheld*: not printed. †No training instance in §5 (the two held-out kinds).

The first version, which delivered the invariant sentence only, closed 44% of the gap on dev (16.0% → 53.5%). Every rollout received its entries, starting in the first turn, so delivery was not the limiting factor. On the 64 purity-defect rollouts of that version, 94% performed the computation but only 34% scored; for comparison, the 7-check baseline reached 10% and 4%, and the prompt with all 17 checks 100% and 98%. Adding the consequence line, and replacing a header that called the entries "reference material, not an instruction" with an imperative one, added about 50 points. Limiting entries per turn rather than per episode added 12 more (Table 5).

dev, 152 withheld-defect instances	mean	majority (k=2)	best-of-2
knowledge layer v1 (invariant only)	53.5	33.6	65.1
knowledge layer v2 (with consequence, imperative, per-turn limit)	89.1	75.0	96.1
all 17 printed (oracle)	98.8	96.1	100.0

Table 5: Iteration on dev. With k=2, a majority requires both rollouts to be fully correct.

Two further changes did not help. Serving first the entries least likely to be delivered, replayed over 302 trajectories, changed no outcome, because entries tied to the same file are delivered or missed together. On clean inputs, an explicit burden-of-proof instruction ("quote the row and the two numbers that disagree; if you cannot, the check passed") raised clean accuracy from 34.2% only to 45.0%.

withheld defect	n	7 printed	knowledge layer	Δ
`purity_af_inconsistent`	39	3	82	+78.8
`candidate_rank_disorder`	34	9	93	+84.6
`candidate_rank_offset`	31	0	82	+82.3
`af_ad_mismatch`	10	40	95	+55.0
`duplicate_candidate`	9	8	97	+88.9
`untraceable_candidate`	9	17	94	+77.8
`vcf_record_orphan`	9	17	83	+66.7
`candidate_hla_unlisted`	7	50	100	+50.0
`caller_count_mismatch`	6	42	92	+50.0
`candidate_pos_orphan`	6	42	83	+41.7

Table 6: Held-out test, per withheld defect (task score). Every paired interval excludes zero.

D TRAINING DETAILS

	first stack (§5)	second stack (§5.1)
training hardware	1× RTX PRO 6000 Blackwell (96 GB)	4× H200 (141 GB), data-parallel
base weights during training	BF16 checkpoint loaded in NF4 (QLoRA)	BF16
corpus	573 trajectories, 4,087 rows (32k)	same 573 trajectories, 4,404 rows (49k)
max length	32,768 (65% of final turns kept)	49,152 (93%)
training / validation rows	3,381 / 706	3,651 / 753 (96 trajectories)
rows per optimizer step	8	8 (2 per GPU × 4)
steps / wall clock	846 / 20 h 50 m	910 / 5 h 46 m
LoRA	r=16, α=32, dropout 0.05; `q,k,v,o,gate,up,down_proj`	
optimization	2 epochs, learning rate 10^{-4}, loss on completion tokens only	
serving base	NVFP4 checkpoint, vLLM 0.28.0, same GPU	BF16 checkpoint, vLLM 0.27.1, H200

Table 7: Training and serving setup. The final turn, which carries the verdict, has a median length of 27,928 tokens.

Why the row counts differ. Both runs use the same 573 teacher trajectories, split into one training example per turn. A turn is dropped when its conversation prefix plus target exceeds the maximum length, so the 32k corpus keeps 4,087 turns and the 49k corpus 4,404. Both runs then hold out 15% of trajectories, chosen by trajectory so that no conversation contributes to both sides, as a validation set, which leaves 3,381 and 3,651 training rows. The main text's corpus size (4,087) counts all rows of the first corpus.

Quantization in the first stack. The first stack had to fit a 27B model with 32k-token sequences on one 96 GB GPU. Training therefore loaded the BF16 checkpoint with 4-bit NF4 quantization (QLoRA (Dettmers et al., 2023)); the LoRA weights themselves are trained and stored in BF16. For serving we used a separately released NVFP4 checkpoint of the same model (`unsloth/Qwen3.8-27B-NVFP4`), a 4-bit floating-point format that the GPU executes natively, and loaded the BF16 adapter on top of it. No conversion of the adapter is needed beyond renaming its keys to the serving model's module names; base model and adapter were served by the same process. The quantization seen in training (NF4) and in serving (NVFP4) therefore differ. This is one of the differences between the two stacks, and §5.1 shows that serving the first stack's adapter on the BF16 stack reproduces the second stack's clean-input behavior.

Adapter coverage. LoRA targets are matched by module name. In the hybrid Qwen3.8 architecture the projections of the 48 linear-attention layers are named `in_proj_qkv`, `in_proj_z` and `out_proj`, so both adapters modify attention only in the 16 full-attention layers, plus every layer's MLP. We noticed this after both runs.

Models. The BF16 Qwen3.8-27B weights were obtained from ModelScope (no Hugging Face revision); we record the SHA-256 of `model.safetensors.index.json` (`77042094...`) and `config.json` (`191e0af2...`). MiniMax-M3 was called through the vendor's API (`https://api.minimaxi.com`) in September 2026; its version cannot be pinned.

Strict correctness. The paper reports the benchmark's task score, which gives one point each for the verdict and the defect name. Requiring both to be correct gives the following. First stack: 90.6% on withheld defects for the distilled model. Second stack: base, distilled and distilled with the verifier reach 10.6, 89.4 and 92.5% on withheld defects, 100.0, 97.6 and 98.2% on named checks, and 90.5, 47.6 and 96.4% on clean bundles. The paired distilled-minus-base difference on named checks is -2.4 $[-4.8, -0.6]$. Undecided rollouts (3 in the verifier arm) count as failures.

E BioProBench details

ERR	n	accuracy	precision	recall	F1
Qwen3.8-27B, official prompt	300	62.3	68.9	46.7	55.7
+ 17 revealed error classes	300	67.0	71.9	57.2	63.7
MiniMax-M3, official prompt	273	62.3	75.0	40.4	52.5
+ 17 revealed error classes	294	64.0	74.7	43.6	–

Table 8: BioProBench ERR. The error classes were written from the training split only and cover 79.4% of its 46,286 corrupted rows. The MiniMax arms have different effective n because of API failures, so we do not pair them.

In the ERR task each item is one step of a laboratory protocol, possibly corrupted, and the model answers whether it contains an error.

Does distillation cost anything here? We also ran the base model and the second-stack adapter twice on the same 300 ERR items, requested by name from one BF16 server, and pooled the two runs (k=2). Unparsed answers count as wrong.

ERR, same server, n=300, k=2	accuracy	precision	recall	unparsed (run 1 / 2)
base model	59.3	70.7	42.8	17 / 15
distilled (second stack)	61.5	73.1	41.1	7 / 10

Table 9: Paired accuracy difference $+2.2$ points, 95% CI $[-0.7, +5.2]$. On the 268 items answered in every run it is $+0.7$ $[-2.2, +3.5]$; all but one unparsed answer ran out of the token budget before writing an answer. Single runs of the base model agree on only 262 of 300 items, so we pool two runs and read the result as no loss rather than a gain.

On the ORD task (reordering shuffled protocol steps) with official prompts, Qwen3.8-27B reaches 51.6% exact match (Kendall τ 0.771) and MiniMax-M3 44.6% (0.751).

F Verifier details

The agent's workspace contains only a stub that knows the request protocol and no check names. The registry answers from a host process, for the directory in which the request file was written, and ignores any path named in the request. In a probe, one rollout searched the stub for the string `choices`, looking for a list of valid check names, and then read the whole file; it found none. The call budget counts distinct checks rather than calls, costs the same for named and withheld checks (different prices would reveal which checks are withheld), and is enforced by the host, since the agent can edit the stub.

G SILENT SCORING DEFECTS

Table 10 lists defects we found in our own evaluation harness. Each produced a plausible number and raised no error. Two of them led to the evaluation rules in §2. The verdict parser (defect 1) mis-scored 7.2% of treatment rollouts but only 2.1% of baseline rollouts, because the treatment writes more prose. A LoRA adapter whose parameter names did not match the serving model's modules (defect 8) was skipped by the server without a warning, so the "trained" model reproduced its own baseline byte for byte; the loss curve and logs gave no sign of it.

#	defect	effect
1	verdict parser: unanchored, case-insensitive pattern where the last match wins; the prose "**The one defect:** in `in`…" was read as defect	7.2% of treatment rollouts (2.1% of baseline)
2	answer extractor: the same pattern matched an instruction in the model's own reasoning	0 of 234 rollouts changed
3	BioProBench extractor took the *first* answer span, which was a restated template	148 of 200 rows wrongly counted as unparsed
4	shard runner used a wrong output directory name, and `|| true` turned the error into a successful exit	summary covered 6 of 48 tasks
5	instance identifiers were hashed without the dataset tag	40 identifier collisions across splits
6	the comparison script read only the dev instance table	no per-defect breakdown on test
7	BioProBench clean rows have an empty text field, and the label is stored as the string `"True"`	measured 21–33% instead of 67%
8	LoRA parameter names did not match the served model; unmatched modules were skipped silently	adapter served as its own baseline

Table 10: Silent scoring defects. Defects 1–3 share one pattern; we replaced them with a single answer-extraction module whose regression tests are real responses that once produced wrong numbers.